Algorithmes multidimensionnels et multispectraux en Morphologie Mathématique : approche par méta-programmation.

Raffi Enficiaud

▶ **To cite this version:**

Raffi Enficiaud. Algorithmes multidimensionnels et multispectraux en Morphologie Mathématique : approche par méta-programmation.. Mathematics [math]. École Nationale Supérieure des Mines de Paris, 2007. English. NNT : . pastel-00003122

HAL Id: pastel-00003122
https://pastel.hal.science/pastel-00003122

Submitted on 4 Feb 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MINES PARIS
ParisTech

ED. n^o431 : Information, Communication, Modélisation et Simulation.

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

Docteur de l'École des Mines de Paris

Spécialité « Morphologie Mathématique »

Présentée et soutenue publiquement
par

Raffi ENFICIAUD

le 26 février 2007

Algorithmes multidimensionnels et multispectraux en Morphologie Mathématique : Approche par méta-programmation.

Directeur de thèse: Michel BILODEAU

Jury

M. Dominique JEULIN	*Président*
Mme. Isabelle BLOCH	*Rapporteur*
M. Christophe DUCOTTET	*Rapporteur*
Mme. Corinne VACHIER-MAMMAR	*Examinateur*
M. Christophe GRATIN	*Examinateur*
M. Michel BILODEAU	*Examinateur*

Abstract

Multi-dimensional and multi-spectral algorithms in the field of Mathematical Morphology : The meta-programming approach.

Raffi Enficiaud

This PhD focuses on algorithms in the field of Mathematical Morphology and Image Processing, from the point of view of modern programming techniques.

Computer science is often considered as a simple witness of hardware improvements dictated by Moore's law. However, software techniques evolve as well, and new programming tools such as meta-programming are now available for the scientific community. Programming is of paramount interest for image processing; for this task, meta-programming brings significant improvements both in scientific terms by providing a powerful mean of abstraction, and in terms of practicability by providing portability, development centralizing, error reduction, etc.

The work presented in this thesis is structured around the elaboration of a general algorithmic framework for - morphological - image processing. Furthermore, examples drawn from concrete industrial applications (in the fields of visual surveillance and car security) are described in order to illustrate the use of some of these developments.

Prior to any algorithmic development, the thesis first proposes a model identifying underlying mathematical notions and their connections: data structures - images, graphs, topology, orders, and priority queues - are revisited with the meta-programming paradigm. This model clearly distinguishes the different actors and reaches the targeted abstraction level. Using specific mechanisms, the automation of numerous tasks is enabled with the compiler. Algorithms are then closer in description to the original mathematical formulations. These developments open for a wide area of research, including N dimensional and hyperspectral images which we further investigate in the following.

The support of both N dimensional imaging and generic programming philosophy led us to define an exact distance transform algorithm. The hypotheses assumed regarding the underlying distance are few (homogeneity in space and convexity of the unit ball), which allows the use of a wide class of functions. Following a theoretical study, an algorithm is proposed based on ordered propagation. The error-proneness on many examples in 4D space (euclidian, ℓ^5, oriented, non-isometric) is also illustrated. A different approach uses morphological distance transforms, popular in the field of mathematical morphology. This approach has recently seen a significant extension from binary to grey-scale images: Beucher's « quasi-distance ». In this thesis, an algorithm is proposed which features lower complexity than the one proposed originally for this purpose.

The handling of colour, or more generally of hyperspectral images (i.e. vectorial data), is rather delicate in the field of Mathematical Morphology. This thesis tackles this issue, and proposes three different approaches: the use of colour metrics, then of local statistics and finally of algebraic lattices by means of lexicographical orders. The programming framework introduced in the document suits all the needs of such approaches. Colour metrics enable the definition of morphological gradients in colour spaces. However, this definition is computationally expensive and practically unusable for wide neighbourhoods. In this case, local statistics provide an interesting alternative solution. Focus was put on circular statistics in HLS colour space, which led to a chromatic gradient.

Lexicographical orders also provide an algebraic framework suitable for mathematical morphology in colour spaces. Using the framework proposed in this thesis, such orders do not require a fundamental

revisiting of the existing algorithms. Operators based on orders and geodesy (extrema, reconstructions, granulometries, ...) are extended with a very low cost in development.

These abstract considerations are illustrated within two concrete applications: skin colour characterization robust to illumination changes (automotive security context), and motion detection (visual surveillance).

Finally, this thesis deals with the issue of segmentation - more precisely, the watershed algorithm. An efficient implementation of the watershed uses hierarchically ordered queues. However the original algorithm using this method leads to some bias. The algorithm proposed in this thesis corrects these biases. Again, a great benefit comes from the proposed framework and, as a result, the algorithm is bound neither to space nor to relief data: watersheds in 4D, on real or colour images are now possible. Region construction is then modified in order to have a finer control on segmentations from a few numbers of markers. The first modification brings external information (e.g. colour or statistical consistency) using a cost function, computed either on the contour or the interior of the region. The second modification is based on contours' local geometrical configuration and simulates a viscous flooding.

Key words Mathematical morphology, meta-programming, C++, N dimensional images, multi-spectral images, exact distance transforms, morphological distances, quasi-distances, colour, circular statistics, lexicographical orders, robust skin detection, illumination changes, motion detection with colour, segmentation, watershed, hierarchically ordered queues, viscous floodings, constraint floodings.

Résumé

Algorithmes multidimensionnels et multispectraux en Morphologie Mathématique : Approche par méta-programmation.

Raffi Enficiaud

Au cours de ces travaux de thèse, nous nous sommes intéressés d'un point de vue global aux algorithmes en Traitement d'Image et plus particulièrement en Morphologie Mathématique, selon certaines techniques nouvelles de programmation.

L'évolution matérielle des moyens informatiques suit les prédictions de la loi de Moore. Cependant, une évolution parallèle, d'ordre logicielle, met à la disposition de la recherche scientifique des moyens de programmation nouveaux, dont la méta-programmation. Les avantages sont considérables, tant en terme scientifique par les possibilités offertes, qu'en termes simplement pratiques (portabilité, capitalisation des développements, réduction des erreurs, etc.).

La présentation des travaux est structurée autour de la conception d'une bibliothèque de traitement - morphologique - d'image. Les différents aspects sont illustrés en partie par des exemples pris dans les domaines de la vidéosurveillance et de la sécurité automobile, et issus de projets industriels.

Nous présentons dans un premier temps le cadre informatique utilisé pour l'écriture algorithmique. Afin de rendre efficace l'utilisation des nouvelles techniques de programmation, une étude préalable des notions mathématiques en Morphologie Mathématique - images, graphes, relations d'ordre, voisinages, éléments structurant, ... -, ainsi que des outils informatiques associés, est réalisée. La séparation correcte des rôles permet en outre l'écriture des structures indépendamment de la nature des données qu'ils contiennent, l'automatisation de nombreuses opérations par le compilateur, et une écriture algorithmique fidèle à une formulation mathématique. La conjonction de ces développements ouvre un grand champ d'exploration comme celui émanant des images nD et hyperspectrales, dont nous nous proposons d'explorer certains aspects.

Le support des images nD associé à la programmation générique a sollicité le développement d'un algorithme de transformée exacte en distance. Les hypothèses sur la fonction distance sont faibles (homogénéité dans l'espace et convexité de la boule unité associée) afin d'utiliser les mêmes développements pour une large classe de fonction. Suite à une étude théorique, nous proposons un algorithme de calcul basé sur des propagations. Le même algorithme est utilisé pour l'ensemble des illustrations (fonctions de distance - ℓ^2, ℓ^5, orienté, non isométrique, ... - sur des images $4D$). Les transformées morphologiques en distance sont d'approche totalement différente et d'usage courant en morphologie mathématique. Elles connaissent actuellement de nouveaux développements grâce à l'extension numérique proposée par Beucher: les « quasi-distances ». Nous proposons un algorithme de calcul rapide de ces distances.

La couleur et plus généralement les images multispectrales (données vectorielles) sont d'une manipulation délicate en morphologie mathématique. Nous présentons trois approches complémentaires: l'utilisation de métriques couleurs, des statistiques locales et enfin les relations d'ordre lexicographique. Notre cadre informatique et algorithmique est parfaitement adapté à ces trois types de traitement. Le cadre métrique permet d'étendre la définition du gradient morphologique aux espaces couleurs, et plusieurs métriques dans Lab et HLS sont envisagées. Cette formulation est cependant coûteuse en termes de calcul et devient impraticable lorsque le voisinage utilisé pour le gradient s'agrandit. L'usage de statistiques locales permet de contourner ce problème. Nous nous sommes particulièrement intéressés

à des statistiques circulaires dans HLS, ce qui nous a amené à la définition d'un gradient chromatique dans cet espace.

Enfin, l'utilisation de relation d'ordre lexicographique étend le cadre algébrique classique à la couleur, sans modification fondamentale des algorithmes. Dans cette optique, nous verrons quels sont les moyens à notre disposition pour étendre la plupart des opérateurs (extrema, reconstruction, granulométries, ...) en maintenant un coût de développement bas. Deux études illustrent ces développements : la caractérisation chromatique de la peau, robuste aux changements d'illumination (contexte automobile), et la détection des zones de mouvement (vidéosurveillance).

Le dernier sujet d'intérêt concerne la segmentation, et plus particulièrement l'algorithme de ligne de partage des eaux. L'implémentation de référence à l'aide de files d'attente hiérarchiques conduit à certains biais que nous corrigeons. L'algorithme proposé étant générique, nous l'appliquons sur des images de dimension 4, sur des reliefs en précision flottante ou couleur. Nous modifions ensuite la construction des bassins versants de manière à contourner certaines difficultés rencontrées lors de la segmentation avec un nombre faible de marqueurs. La première modification injecte dans le processus de propagation une information extérieure exprimée sous forme de fonction de coût. Cette fonction concerne aussi bien le contour que l'intérieur de la région en cours de construction. La seconde modification utilise une contrainte locale et rend le fluide *visqueux*. Des analogies sont établies entre ces nouvelles propagations et les équations d'évolution de courbe à l'aide de dérivées partielles.

Mots clefs Morphologie mathématique, méta-programmation, C++, images en N dimensions, images multispectrales, transformée en distance exacte, distances morphologiques, quasi-distances, couleur, statistiques circulaires, ordres lexicographiques, détection robuste de peau, changement d'illuminant, détection de mouvement par la couleur, segmentation, ligne de partage des eaux, files hiérarchiques d'attente, propagations visqueuses, propagations contraintes.

« L'impossible, nous ne l'atteignons pas, mais il nous sert de lanterne. »
René Char

« L'heure de la fin des découvertes ne sonne jamais. »
Colette

« La liberté, c'est de n'arriver jamais à l'heure. »
Ubu roi, Alfred Jarry

Je remercie infiniment Michel Bilodeau. Malgré son humilité sans fond, peu de gens égalent ses qualités humaines et ses compétences scientifiques. Son ouverture d'esprit, ses idées et ses critiques ont fourni à ce document sa forme quasi-définitive, et m'ont convaincu de l'intérêt du travail que j'avais fait au Centre de Morphologie Mathématique. Je pense lui avoir pris beaucoup d'énergie en motivation, tant mon inertie à changer d'avis est importante.

Je remercie Serge Beucher pour avoir conduit une partie de mes travaux. J'insiste particulièrement sur son tempérament d'un abord parfois difficile, mais montrant rapidement une sympathie extrême et une patience que de rares personnes seulement peuvent atteindre. Je le remercie également pour le soutien de ses collègues, sachant extraire l'essence de leurs travaux et les présentant avec une verve infatigable. J'apprécie également les critiques intarissables et le scepticisme sans retenue qu'il manifeste pour les travaux de nos « concurrents » (et je lui donne raison !).

Je remercie Dominique Jeulin, pour sa grande motivation, son altruisme, ses connaissances encyclopédiques et sa curiosité. Je le remercie particulièrement pour m'avoir accordé du temps malgré son emploi du temps toujours très chargé. Je remercie Fernand Meyer de m'avoir accueilli au centre de Morphologie Mathématique, et m'avoir permis de réaliser ce projet Ô combien houleux. Ces deux personnages, hauts dignitaires de la Morphologie Mathématique, créent une aura particulièrement fertile autour du thème du traitement d'image.

Je remercie Mme. Isabelle Bloch et M. Christophe Ducottet, qui ont accepté le rôle délicat de rapporteur, et ce malgré la petitesse de la police de caractère et le nombre de pages assez important. Leurs critiques constructives et leurs appréciations ont rendu ce travail moins indigeste. Je remercie également les autres membres du jury, Mme. Corinne Vachier-Mammar & M. Christophe Gratin avec qui j'ai eu l'occasion de collaborer sur le projet *Clovis*, qui ont bien voulu accorder de leur temps pour la relecture de certaines parties de ce manuscrit.

À la suite d'un séjour aussi long au Centre de Morphologie Mathématique, je ne peux qu'avoir beaucoup de monde à remercier. Galanterie oblige, je commencerai par les dames. Catherine Moysan a accompagné mes péripéties dans le fabuleux monde de l'administration. Heureusement l'ironie toute féminine qu'elle manifestait à mon ignorance était toujours accompagnée d'un indécrochable sourire radieux et d'une bonne humeur, et qui sont les plus beaux joyaux du centre. Laura Andriamasinoro, la petite force tranquille coincée entre deux écouteurs au fond de la bibliothèque, a toujours manifesté de la disponibilité, de la gentillesse et de la motivation pour les nombreux échanges musicaux.

Béatriz et Étienne ont égayé de nombreuses soirées bellifontaines, et ont initié le concept de repas international. J'apprécie notamment la sympathie et l'amabilité des échanges que j'ai eus avec eux, surtout le jamón espagnol et la « confiture de lait avec plus de sucre et encore plus de caramel » péruvien, et les nombreuses relectures des choses inachevées.

Je porte une grande reconnaissance à l'égard de Jean-Claude Klein pour les très très très nombreuses discussions administratives, et surtout pour ses qualités de médiation qui m'ont sauvé d'une voie sans issue.

J'adresse une mention particulière pour Romain Lerallut avec qui j'ai beaucoup travaillé. Sans lui, le support de cette thèse (anciennement Morphée) n'existerait pas. J'ai appris énormément de choses à son contact, dont le travail en équipe, certes en comité réduit, mais avançant dans une synergie particulièrement rare. J'espère que nous aurons l'occasion de collaborer encore à l'avenir, sur des projets toujours plus intéressants et enrichissants.

Gabriel Fricout, (ex)thésard et cascadeur, incarne autant « L'homme qui tombe à pic » que « Barracuda »; j'ai particulièrement apprécié ses humeurs, son scepticisme pour les choses qu'il n'aime pas faire (comme quoi faut pas se forcer) et sa motivation pour tous les sports ne nécessitant pas de souplesse ni de grâce. Tout cela ne retire en rien la pertinence des nombreuses discussions (utiles ou futiles) que j'ai eues avec lui. Il n'y a d'ailleurs que Laurence pour nourrir un animal pareil.

Nicolas Laveau pour ses gâteaux, ses thés aux parfums nombreux, sa bouilloire toujours prête et la porte de son bureau ouverte en permanence (qui trahit un peu une certaine volonté à être « dérangé »).

Jesús Angulo López, pour les nombreux échanges sur la couleur, sa grande sympathie et sa patience. Francis Bach, pour son aide, le Backgamon, les bonnes tranches de viande, sa patience infaillible à mes questions matinales, et sa capacité à tirer le niveau vers le haut. Jaromír Brambor, pour sa gentillesse et pour l'absolue nécessité de parler de tous les détails. Marc & Miguel & la sangria & le mouvement anarchiste espagnol.

Thomas Retornaz pour la bonne humeur qu'il a apportée au grenier et ses clopes savoureuses (« c'est toi qui les fume, et c'est toujours ça de gagné pour mes poumons »), et son attitude toute italienne. Je ne dois pas trop m'étendre sur les personnages des autres centres, concurrence oblige, mais leur présence a été très agréable dans le désert culturel bellifontain. Je pense notamment à Rosalie Vandromme, avec qui je me suis bien fendu la gueule (et le bide); la petite Julie Lions, Aurélie Jouve et les astuces administratives, Sunseare Gabalda, Hugo Bauer et ses jeux de mots désastreux, Caroline & François Prognon les amoureux de la faune domestique. Aux nouveaux et aux anciens dont je ne connais respectivement pas et plus les noms.

À mes amis Thomas Walter et Costin Alin Cacciu, qui ont accompagné mon séjour au centre. Thomas, grâce à sa patience et son pouvoir argumentateur, m'a convaincu de l'intérêt de parler avec un individu tel que Costin (que je regardais d'un œil assez farouche quand même). J'ai découvert beaucoup de choses sous son regard qui m'auraient très certainement échappé par impétuosité, arrogance, impatience, enfin bref, tous ces petits vices qui gâchent un peu les jugements. Quant à Costin, j'ai grâce à lui maîtrisé « l'argumentation par la mauvaise foi » et les breuvages « aux plantes » et « qui désinfectent » (malgré l'innocence du flacon). J'ai découvert en lui un esprit de camaraderie proche du mien et une capacité à encaisser les plaisanteries avec la juste froideur de l'Est.

À Emmanuel Baccelli, le myspacien rocker superstar, qui a été un des supports de motivation des plus importants durant toutes ces années de thèse. Il a accompagné mes déceptions et mes râles sans broncher, et sans le savoir a été l'objet d'une compétition acharnée dans le jeu du « c'est ki-ki-termine-le-plus-tôt » (que j'ai perdu haut la main avec deux belles années de retard).

À Vahé Tadévossian, l'Ange Gris. Sous son sourire angélique et charmeur se cache un ami têtu & obtus, une tête faite d'un bois presque aussi dur que la mienne, et bien sûr un ami sincère et généreux.

À Guillaume Jeulin, que je n'ai pas assez vu durant toutes ces années, mais avec qui je partage un goût prononcé pour la « qualité », notion fondamentale trop souvent oubliée de nos jours pour d'inutiles histoires de temps ou de rentabilité.

À ma famille, à mes parents et à ma sœur, qui m'ont toujours soutenu dans mes choix.

Enfin, à Tina Klein, à qui je réserve les pages commençant à la fin de ce manuscrit...

Table des matières

i Notations xiii
ii Terminologie xiv

1 Introduction **1**
1.1 Avant-propos 2
1.1.1 Pourquoi le traitement d'image ? 2
1.1.2 Traitement d'image et Morphologie Mathématique 3
1.1.3 Les enjeux 4
1.2 Cadre des projets 5
1.3 Plan du mémoire 7

2 Morphologie Mathématique et méta-programmation **9**
2.1 Introduction à la méta-programmation 9
2.1.1 Avant-propos sur la programmation objet 10
2.1.2 Méta-programmation 11
2.1.3 Mécanismes de méta-programmation 14
2.2 Méta-programmation des structures de données en Morphologie Mathématique 16
2.2.1 Les images 16
2.2.2 Les sous-ensembles 18
2.2.3 Les graphes 24
2.2.4 Les relations d'ordre et les treillis 25
2.2.5 Opérateurs 28
2.2.6 Les files d'attente hiérarchiques 33
2.2.7 Conclusion 38
2.3 Conception d'algorithmes 38
2.3.1 Approche mathématique de l'écriture algorithmique 38
2.3.2 Squelette d'algorithme 41
2.4 Conclusion 44

3 Distances **47**
3.1 Les transformations en distance 47
3.1.1 Définition 48
3.1.2 Transformée en distance 50
3.1.3 État de l'art et considérations algorithmiques 51
3.2 Distances exactes en n dimensions 57
3.2.1 Algorithme proposé 57
3.2.2 Résultats 62

3.2.3 Méta-programmation . . . 65
3.2.4 Conclusion . . . 65
3.3 Quasi-distances . . . 67
3.3.1 Introduction . . . 67
3.3.2 Algorithme pour le calcul des quasi-distances . . . 79
3.3.3 Extension des opérations résiduelles à la couleur . . . 91
3.4 Conclusion . . . 100

4 Vers le traitement multispectral **103**
4.1 Avant-propos : Approches connues du traitement multispectral . . . 104
4.2 Approches métriques et statistiques . . . 105
4.2.1 Espaces couleurs et métriques . . . 105
4.2.2 Approche statistique : utilisation sur des espaces cylindriques . . . 119
4.2.3 Récapitulatifs sur les gradients couleurs . . . 128
4.3 Approche algébrique : ordres lexicographiques . . . 130
4.3.1 Reconstruction lexicographiques . . . 130
4.3.2 Granulométries lexicographiques . . . 137
4.3.3 Conclusion sur la lexicographie . . . 139
4.4 Invariants chromatiques robustes de peau . . . 140
4.4.1 Traitement de la peau . . . 140
4.4.2 Influence de l'illuminant sur la représentation cylindrique . . . 149
4.4.3 Marquage des zones de peau . . . 157
4.5 Détection d'objets en mouvement . . . 164
4.5.1 Principales techniques de détection de mouvement . . . 164
4.5.2 Utilisation de la couleur . . . 166
4.6 Conclusion . . . 173

5 Inondation **177**
5.1 Ligne de partage des eaux . . . 178
5.1.1 Avant propos . . . 178
5.1.2 Cadre théorique . . . 179
5.2 Algorithme . . . 184
5.2.1 L'algorithme par files hiérarchiques d'attente . . . 185
5.2.2 Algorithme de ligne épaisse de partage des eaux isotrope, à files hiérarchiques d'attente . . . 190
5.2.3 Inondation en n dimensions . . . 198
5.2.4 Inondation dans $\mathbb{R}$ . . . 200
5.2.5 Inondation et treillis lexicographiques . . . 202
5.2.6 Conclusions sur la ligne de partage des eaux . . . 204
5.3 Injection d'information a priori . . . 206
5.3.1 Problème ouvert . . . 206
5.3.2 Nature de la connaissance a priori . . . 206
5.3.3 Méthode proposée . . . 207
5.3.4 Information de type « point » . . . 208
5.3.5 Information de type « région » . . . 209
5.3.6 Quelques détails d'implémentation . . . 215
5.3.7 Conclusion et perspectives . . . 216
5.4 Inondations localement contraintes . . . 218
5.4.1 Contrainte locale et viscosité . . . 218
5.4.2 Algorithme de propagation avec contrainte locale . . . 221
5.4.3 Expression par EDP et courbure locale . . . 235
5.4.4 Conclusion . . . 238
5.5 Conclusion sur les inondations . . . 240

6 Conclusion générale **241**
6.1 Rappel des objectifs . . . 241
6.2 Perspectives . . . 242

Annexes **247**

A La couleur **249**
A.1 Cadre physique . . . 249
A.2 Représentation numérique de la couleur . . . 251
A.3 Formules de transformation d'espace couleur . . . 255
A.3.1 Espaces informatiques et vidéos . . . 255
A.3.2 Espaces colorimétriques . . . 256
A.3.3 Espaces interface . . . 259

Bibliographie **263**

Table des figures

1.1 Les principes du traitement d'image . . . 3
1.2 Extraits de la base d'acquisition « Logitech » . . . 6
1.3 Images « CEA » et « CMM » . . . 7

2.1 Itérateur de la classe *Image* . . . 19
2.2 Trois exemple d'itérateur sur la classe image . . . 20
2.3 Application simple utilisant les itérateurs sur régions avec prédicat . . . 21
2.4 Quelques graphes de connexité très utilisés . . . 22
2.5 Diagramme de Hasse du treillis $\mathcal{M}_3$ . . . 27
2.6 Codage d'un contour à l'aide d'un lacet . . . 34
2.7 Adéquation entre le voisinage et le codage de la courbe en *fifo* . . . 34
2.8 Structure de *fah* générique. . . . 36
2.9 File d'attente à hiérarchie discrète . . . 37

3.1 Les boules unitées pour les distances d_p, $p \geq 1$ . . . 50
3.2 Illustration de la propagation pour le calcul de la fonction distance . . . 52
3.3 Déconnexion du diagramme de Voronoï en trame discrète . . . 53
3.4 Structure de donnée pour le stockage des points . . . 58
3.5 Distance euclidienne sur un ensemble de 30 points. Coupure à $d = 6$. . . . 63
3.6 Distance ℓ^5 sur un ensemble de 30 points. Coupure à $d = 6$. . . . 64
3.7 Distance non-isométrique . . . 64
3.8 Distance non-isométrique . . . 65
3.9 Application de la distance morphologique binaire pour l'exemple de séparation de grains 69
3.10 Non unicité de la valeur maximale du résidu . . . 70
3.11 Quasi-distance sur l'image « Salt » . . . 71
3.12 Violation de la contrainte lipchitzienne de la fonction distance par les érodés successifs dans le cas numérique. . . . 72
3.13 Utilisation de la quasi-distance sur les gradients de Malik . . . 74
3.14 Utilisation de la quasi-distance pour la vidéosurveillance . . . 74
3.15 Exemple de la transformation en Quasi-Distance . . . 75
3.16 Exemple de la transformation en Quasi-Distance . . . 76
3.17 Exemple de la transformation en Quasi-Distance . . . 77
3.18 Exemple de transformation en Quasi-distance . . . 77
3.19 Calcul de la famille d'érosions successives . . . 81
3.20 Régularisation de la fonction distance . . . 84

3.21 Tailles des files d'attente dans les deux algorithmes proposés pour l'image « Fleur Budapest » . . . 85
3.22 Taille des files d'attente dans les deux algorithmes proposés pour l'image « Fifer » . . . 85
3.23 Taille des files d'attente dans les deux algorithmes proposés pour l'image « Retina » . . . 86
3.24 Temps de calcul des érodés successifs . . . 87
3.25 Logarithme du rapport des temps de calcul des érodés successifs, par pixel . . . 87
3.26 Temps de calcul de la régularisation . . . 88
3.27 Logarithme du rapport des temps de calcul de la régularisation, par pixel . . . 88
3.28 Temps de calcul de l'approche hybride en fonction de k . . . 89
3.29 Temps de calcul de l'approche hybride en fonction de k - images inverses . . . 89
3.30 Gains maximaux de l'approche hybride . . . 90
3.31 Gains maximaux de l'approche hybride - images inverses . . . 90
3.32 Érosions lexicographiques dans le treillis L, S, H (ordres naturels) . . . 93
3.33 Érosions lexicographiques dans le treillis L, S, H (ordres inverses sur chaque canal) . . . 94
3.34 Érosions lexicographiques dans le treillis S, L, H (ordres naturel) . . . 94
3.35 Érosions lexicographiques dans le treillis S, L, H (ordres inverses) . . . 95
3.36 Résidus des érosions lexicographiques dans le treillis L, S, H par la métrique « HL - saturation pondéré » . . . 96
3.37 Résidus des érosions lexicographiques dans le treillis S, L, H (ordres inverses) par la métrique « HL - saturation pondéré » . . . 96
3.38 Quasi-distance sur le treillis L, S, H, et S, L, H . . . 97
3.39 Quasi-distance sur le treillis L, S inversé, H, et S inversé, L, H . . . 98

4.1 Géométrie de l'espace HLS . . . 107
4.2 Intérêt de la couleur dans le cadre de la détection d'objets en mouvement . . . 109
4.3 Gradient lexicographique sur l'image « Fleur Roumanie 1 » . . . 114
4.4 Gradient lexicographique sur l'image « Carmen » . . . 115
4.5 Gradient morphologique généralisé dans Lab, distance euclidienne - Image « Parrots » . . . 116
4.6 Gradient morphologique généralisé dans Lab, distance euclidienne - Image « CEA » . . . 116
4.7 Gradient morphologique généralisé dans Lab, distance en teinte - Image « Cavalier » . . . 117
4.8 Illustration du gradient morphologique généralisé sur l'image « Carmen » . . . 117
4.9 Comparaison de gradients et demi-gradient morphologiques pour l'image « Fleur Roumanie » . . . 118
4.10 Comparaison de gradients et demi-gradient morphologiques pour l'image « Fleur Budapest » . . . 118
4.11 Utilisation de la saturation comme facteur de pondération de la teinte . . . 122
4.12 Illustrations des gradients de dispersion - Fleur 1 . . . 124
4.13 Illustrations des gradients de dispersion - Fleur Budapest . . . 124
4.14 Illustrations des gradients de dispersion - Fleur 2 . . . 125
4.15 Illustrations des gradients de dispersion - Extrait base *Inria* . . . 125
4.16 Illustrations des gradients de dispersion - Extrait base *Logitech* . . . 126
4.17 Implémentation de la moyenne locale pour le gradient de dispersion . . . 126
4.18 Illustrations des gradients de dispersion - Parrots . . . 127
4.19 Illustration de la reconstruction . . . 131
4.20 Illustration de la reconstruction lexicographique sur « Ishihara 5 ». . . . 136
4.21 Extraits du calcul de la granulométrie. . . . 137
4.22 Fonction dérivée granulométrique de l'image 4.21 . . . 138
4.23 Extraits du calcul de la granulométrie avec des ouvertures par reconstruction. . . . 138
4.24 Fonction dérivée granulométrique de l'image 4.23 . . . 139
4.25 Exemple issu de la base Inria. . . . 142
4.26 Locus de peau dans xyY à partir de 3 primaires RGB différentes - *base T* . . . 144
4.27 Locus de peau dans Lab à partir de 3 primaires RGB différentes - *base T* . . . 144
4.28 Locus de peau dans xyY sur différents illuminants - *base T* . . . 145
4.29 Locus de peau dans Lab sur différents illuminants - *base T* . . . 145

4.30 Locus de peau dans Lab de l'ensemble des bases . . . 146
4.31 Locus de peau sur les espace HLSI1 et GHLS- base Hitachi. . . . 147
4.32 Locus de peau dans GHLS de l'ensemble des bases . . . 147
4.33 Rose des loci de peau dans GHLS sur l'ensemble des bases . . . 148
4.34 Exemple de changement d'illuminant - Base *Hitachi*. . . . 152
4.35 Exemple de changement d'illuminant - Base *Logitech*. . . . 152
4.36 Calcul local et global des paramètres $\bar{\mu_0}$ et $\bar{R}_0$ . . . 154
4.37 Influence sur la teinte du changement d'illuminant sur une image de la base *Hitachi* . . . 155
4.38 Influence sur la teinte du changement d'illuminant sur une image de la base *Inria* . . . 156
4.39 Teinte moyenne locale, intégrée sur la totalité du visage . . . 157
4.40 Comparaison de la dispersion moyenne avec et sans pondération par la saturation . . . 158
4.41 Dispersion moyenne par site et par base . . . 159
4.42 Distribution de $|\bar{R}_{0,l} - \bar{R}_{0,g}|$ . . . 160
4.43 Variance de $\bar{R}_{i,l}, i \in \{0,1,2,3\}$ - toutes bases confondues . . . 161
4.44 Résultats de l'utilisation de la dispersion et de la teinte. Séquence « Gabriel ». . . . 161
4.45 Résultats de l'utilisation de la dispersion et de la teinte. Séquence « Thomas ». . . . 162
4.46 Résultats de l'utilisation de la dispersion et de la teinte. Séquence « Romain ». . . . 162
4.47 Résultats de l'utilisation de la dispersion et de la teinte. Séquence « Beatriz ». . . . 163
4.48 Ouverture du gradient temporel . . . 164
4.49 Profil originaux en teinte, luminance et saturation sur un échantillon vidéo . . . 168
4.50 Teinte moyenne temporelle . . . 169
4.51 Moyenne spatiale en teinte - sans pondération - au cours du temps . . . 170
4.52 Intégration temporelle de la teinte prétraitée par moyenne locale . . . 171
4.53 Comparaison des approches en luminance et chrominance . . . 172

5.1 LPE - Étapes d'inondation d'un relief. . . . 178
5.2 Illustration de l'algorithme de *lpe* à base de *fah* . . . 186
5.3 Biais sur la détection locale des points de la ligne de partage des eaux . . . 188
5.4 Problème de parité de l'algorithme à base de *fah*, lié à l'ordre de traitement . . . 189
5.5 Illustration du traitement atomique des points de la *fah*. . . . 191
5.6 Illustration de l'algorithme de *lpe* pour la détection des lignes épaisses. . . . 194
5.7 Illustration de la collision de deux points contenus dans la *fah*. . . . 195
5.8 Illustration de l'algorithme de *lpe* pour la détection d'un point « normal ». . . . 197
5.9 Ligne de partage des eaux en $4D$ - Coupe 0 . . . 200
5.10 Ligne de partage des eaux en $4D$ - Coupe 6 . . . 201
5.11 Illustration du gradient vectoriel utilisé pour la *lpe* lexicographique. . . . 203
5.12 Dégradation du gradient. . . . 204
5.13 Application de la *lpe* lexicographique sur le gradient 5.12. . . . 205
5.14 Illustration de la contrainte globale sur l'image « Parrots » . . . 215
5.15 Illustration de la contrainte globale sur l'image « La Perte de Virginité » (Gauguin) . . . 216
5.16 Comparaison entre les algorithmes avec et sans mise à jour des points du front de propagation . . . 217
5.17 Défauts du gradient . . . 218
5.18 Comportements d'un fluide visqueux. . . . 220
5.19 Propagation et mise à jour locale de la priorité des points. . . . 226
5.20 Illustration de l'isotropie de l'inondation contrainte . . . 227
5.21 Illustration de la non-isotropie sur un schéma classique . . . 227
5.22 Illustration de l'inondation visqueuse sur un relief présentant des « brèches » . . . 228
5.23 Illustration de l'inondation visqueuse sur un relief présentant des « trous » lisses . . . 229
5.24 Illustration de l'inondation contrainte sur l'image « Nicolas » . . . 230
5.25 Illustration de l'inondation contrainte sur l'image « Raffi » . . . 231
5.26 Illustration de l'inondation visqueuse sur l'image « fleur roumanie » . . . 232
5.27 Illustration de l'inondation visqueuse sur l'image « fleur Roumanie 2 » . . . 233
5.28 Illustration du problème d'isotropie liée au centrage de la boule de contrainte . . . 234

5.29 Problème lié à la mesure sur un voisinage 235
5.30 Détails du calcul du rapport des surfaces 236
5.31 Surface en fonction du rayon de courbure 238

6.1 Une illustration de Oleg Delgadcshov 241

A.1 Fondamentales de projection CIE 252
A.2 Exemples de variation de température d'illuminant, issus de la base « The Amsterdam Library of Object Images » 253
A.3 Différence de teinte entre GHLS et HLS 1 255

Liste des tableaux

A.1 Coefficients $YC_{bleu}C_{rouge}$ 261
A.2 Coordonnées chromatiques des blancs de références dans l'espace xyY $(Y = 1)$ 261
A.3 Coordonnées de primaires rgb dans l'espace xyY 261

Liste des Algorithmes

2.1 Opérateur séquentiel de voisinage 32
2.2 Insertion d'un point (h, q) dans la *fah* 36
2.3 Squelette algorithmique de la labellisation 43
3.1 Distance euclidienne exacte nD - Initialisation 60
3.2 Distance euclidienne exacte nD - Construction de la nouvelle enveloppe 61
3.3 Calcul de l'indicatrice of q - Algorithme de S. Beucher [Beu03] 73
3.4 Regularisation de l'indicatrice q - Algorithme de S. Beucher [Beu03] 73
3.5 Calcul de la famille des érodés successifs 80
3.6 Régularisation des distances perchées 83
4.1 Opérateur de gradient morphologique généralisé 112
4.2 Reconstruction à base de files d'attente hiérarchiques 133
4.3 Squelette algorithmique pour la détection des extrema locaux 135
5.1 *lpe* à files d'attente hiérarchiques - Version originale par F. Meyer [Mey91] 187
5.2 *lpe* épaisse isotrope à file d'attente hiérarchique - Initialisation 192
5.3 *lpe* épaisse isotrope à file d'attente hiérarchique - Propagation 196
5.4 *lpe* épaisse isotrope à file d'attente hiérarchique - Propagation - suite 198
5.5 *lpe* épaisse isotrope à contrainte sur les points - Initialisation 208
5.6 *lpe* épaisse isotrope à contrainte sur les points - Propagation 208
5.7 *lpe* épaisse isotrope à contrainte sur les régions - Initialisation 211
5.8 *lpe* épaisse isotrope à contrainte sur les régions - Propagation 211
5.9 *lpe* épaisse isotrope à contrainte sur les régions - Mise à jour 213
5.10 Inondation par fluide à contrainte locale - Initialisation 222
5.11 Inondation par fluide à contrainte locale - Initialisation - Suite 223
5.12 Inondation par fluide à contrainte locale - Propagation 224
5.13 Inondation par fluide à contrainte locale - Propagation - Suite 225

i - Notations

Voici les notations que nous utiliserons tout au long du manuscrit:

$\mathbf{E}, \mathbf{M}, \mathbf{B}$	des ensembles
$\mathbf{E} = \mathbf{E}_1 \times \mathbf{E}_2$	le produit cartésien de E_1 et E_2
$\mathbf{E}^n$	$\mathbf{E} \times \mathbf{E} \ldots \times \mathbf{E}$ (n fois)
$\varnothing$	l'ensemble vide
$\mathcal{P}(\mathbf{E})$	l'ensemble des parties de $\mathbf{E}$
$\overline{\mathbf{E}}$	l'ensemble complémentaire de $\mathbf{E}$
$\mathbf{A} \setminus \mathbf{B}$	$\mathbf{A} \cap \overline{\mathbf{B}}$
$\mathcal{I}$	une image
$\mathbf{E}, \mathbf{F}$	resp. l'espace des coordonnées et l'espace image (pixels)
$\mathcal{N}, \mathcal{N}_p$	resp. voisinage et voisinage centré au point p
$\mathcal{N}^k$	voisinage à k connexités
$\triangleq$	une définition
$\vee, \wedge, \neg$	respectivement « ou », « et » et « négation » logiques
$\underline{ou}$	$a \ \underline{ou} \ b \Leftrightarrow (a \ \vee \ b) \ \wedge \neg (a \ \wedge \ b)$ (ou exclusif)
$\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}$	respectivement l'ensemble des entiers naturel, des entiers, des réels et des complexes.
$\mathbb{E}^*$	un ensemble $\mathbb{E}$ privé de élément neutre pour l'addition
$\mathbb{N}_k$	$\{0, 1, 2, \ldots, k\}$
$\mathbf{i}$	$\sqrt{-1}$
$\sharp$	cardinal d'ensemble dénombrable
$f : K \to K'$	une fonction f sur l'ensemble K à valeurs dans K'
$g \circ f$	composition des applications g et f : $g \circ f(x) = g(f(x))$
$f^{(n)}$	$f \circ f \ldots \circ f$ (n fois, $n \in \mathbb{N}$)
$\mathcal{A}(\mathbf{E}, \mathbf{F})$	espace des fonctions sur K à valeurs dans $\mathbf{F}$
$\mathbf{F}^{\mathbf{E}}$	espace des fonctions sur $\mathbf{E}$ à valeurs dans $\mathbf{F}$
$\mathbf{supp}(f)$	support de f
ℓ^k	norme de type $\ell^k(a) = \left(\sum_i \lvert a_i \rvert^k\right)^{\frac{1}{k}}$
d_k	fonction distance issue de la norme ℓ^k
$\mathcal{L}$	un treillis dont les propriétés seront énoncées
$\underline{\vee}, \overline{\wedge}$	respectivement les fonctions supremum et infimum sur un treillis
$\prec, <$	relation d'ordre stricte
$\preceq, \leq$	relation d'ordre large
$\cong_\prec$	équivalence selon $\prec$: $\neg(a \prec b) \wedge \neg(b \prec a) \Leftrightarrow a \cong_\prec b$
$a[c]$	a modulo c
$\mathcal{M}_{s_1 \to s_2}$	matrice de passage de l'espace s_1 vers l'espace s_2

$a * b$	produit de convolution de a et b
$\|a\angle b\|$	$\max(\|a-b\|[\pi], 2\pi - \|a-b\|[\pi])$: angle aigu non orienté entre a et b
$\oplus, \ominus$	resp. addition et soustraction de Minkowski
$\circ, \bullet$	resp. ouverture et fermeture à partir des opérateurs de Minkowski
ϵ, δ	resp. érosion et dilatation
γ, φ	resp. ouverture et fermeture
$\epsilon_{\mathbf{E}}, \delta_{\mathbf{E}}$	resp. érosion et dilatation géodésiques dans $\mathbf{E}$
$\gamma^{\infty}_{\mathbf{E}}, \varphi^{\infty}_{\mathbf{E}}$	resp. ouverture et fermeture par reconstruction dans $\mathbf{E}$
$\bar{\mu_0}$	moyenne circulaire
$\bar{R_0}$	dispersion circulaire
$\bar{\mu_p}$	moment circulaire d'ordre p
$\bar{R_p}$	dispersion circulaire centré d'ordre p
∇_g	gradient morphologique généralisé
$\nabla_g^{1/2}$	demi-gradient morphologique généralisé
∇_{ch}	gradient circulaire d'homogénéité
∇_{chp}	gradient circulaire d'homogénéité avec pondération par la saturation

ii - Terminologie

MM Morphologie Mathématique

fa *File d'attente*. Structure de données permettant une représentation pratique et efficace des contours et fronts de propagation.

fah file d'attente hiérarchique. File d'attente dotée d'une relation d'ordre. L'extraction des éléments de la file respectent cette relation d'ordre.

fifo file de type « *First In, First Out* »: premier arrivé, premier sorti

lifo file de type « *Last in, First Out* »: dernier arrivé, premier sorti

POO Programmation Orienté Objet

lpe Ligne de Partage des Eaux

JND Différence minimale perceptible (de l'anglais « *Just Noticeable Difference* »)

spd Distribution de puissance spectrale (de l'anglais « *Spectral Power Distribution* »)

SKIZ Squelette par zone d'influence (de l'anglais « *Skeleton by Influence Zone* »)

CHAPITRE 1

Introduction

> « Stratagème 33:
> *"C'est peut-être vrai en théorie, mais en pratique c'est faux"*. Grâce à ce sophisme, on admet les fondements tout en rejetant les conséquences; en contradiction avec la règle *a ratione ad rationatum valet consequentia* (la conséquence tirée de la raison première valide le raisonnement). Cette affirmation pose une impossibilité: ce qui est juste en théorie doit aussi l'être en pratique; si ce n'est pas le cas, c'est qu'il y a une erreur dans la théorie, qu'on a omis quelque chose, qu'on ne l'a pas fait rentrer en ligne de compte; par conséquent c'est également faux en théorie. »
> *L'art d'avoir toujours raison*, A. Shopenhauer.

Nous pouvons affirmer sans crainte que l'*informatique* est un liant de choix entre des développements théoriques d'une part, et les observations expérimentales d'autre part. En effet, de plus en plus souvent, la collecte des données expérimentales est effectuée sous forme numérique par des ordinateurs. Par cette position déterminante dans la démarche scientifique des *Mathématiques Appliquées*, l'informatique est une discipline qui ne peut être négligée. Le traitement d'images suit précisément cette logique des mathématiques appliquées: l'élaboration d'une théorie, puis son expérimentation et sa vérification à l'aide d'outils essentiellement informatiques.

Lorsque l'on se réfère aux ordinateurs, il est souvent mentionné l'*augmentation des capacités de calculs* et de *stockage*, en parfaite conformité avec la loi de *Moore*. Il est alors sous-entendu une augmentation de la *précision* et de la *qualité* des observations: l'augmentation de la capacité de stockage et de calcul conduit respectivement à une croissance de la précision de la collecte numérique des données, et à des traitements de ces mêmes données dans des temps équivalents.

Parallèlement à l'évolution des moyens matériels, l'évolution logicielle et plus particulièrement celle des langages de programmation, revêt une importance fondamentale dès lors que l'on s'intéresse à l'écriture d'algorithmes, dont la définition est la suivante:

Définition 1.1 (Algorithme (Larousse))
(De l'arabe al-kharezmi, surnom d'un mathématicien arabe)
Suite finie d'opérations élémentaires constituant un schéma de calcul ou de résolution d'un problème.

L'écriture algorithmique suit une organisation stratifiée du simple au complexe. Pour écrire les algorithmes les plus simples, il faut disposer d'entités élémentaires. Ces algorithmes deviennent ensuite et à leur tour des entités utilisables dans d'autres algorithmes plus complexes. Les entités élémentaires sont principalement les structures de données offrant des moyens de stockage et d'accès. Elles représentent la base de tous les développements et méritent une attention particulière: plus cette base sera riche et

bien conçue, plus les possibilités algorithmiques seront étendues.

Le travail présenté dans ce mémoire de thèse a pour origine l'utilisation de nouvelles techniques de programmation, rendues possibles par l'apparition de compilateurs puissants et leur standardisation. Ces techniques offrent un grand niveau d'abstraction, rapprochant en outre l'écriture algorithmique de la formulation mathématique du problème. Autour de cette base algorithmique sont présentés des algorithmes complexes, démontrant à la fois la faisabilité des traitements et la souplesse des moyens logiciels. Certains points clefs du traitement informatique des images sont abordés sous l'œil neuf des techniques introduites.

Ce mémoire s'adresse donc à un large public. Les personnes désireuses d'être initiées à quelques techniques nouvelles de programmation trouveront une description de ces techniques ainsi que des exemples d'utilisation - les algorithmes - montrant leur pertinence. Celles confrontées à l'élaboration d'une bibliothèque y verront un exemple d'agencement des notions sous-jacentes, qui leur permettra de bénéficier d'une base solide ainsi que d'une réduction des temps de développement. Les personnes plus proches de la théorie du traitement d'image ou de la Morphologie Mathématique (MM) se réjouiront du niveau d'abstraction et du formalisme adopté dans de nombreuses sections. Elles apprécieront également l'étendue des sujets traités autour de la MM. Les praticiens trouveront des méthodes et transformations nouvelles, ainsi que des exemples d'application concrètes issues du monde industriel. Enfin pour les personnes les plus courageuses, certains algorithmes présentés pourront être considérés comme des implémentations détaillées de référence.

1.1 Avant-propos

Selon un esprit d'ouverture scientifique, nous allons présenter succinctement le traitement d'image. Nous recentrerons ensuite le sujet sur la morphologie mathématique. Bien que le traitement d'image soit une discipline tendant à se démocratiser, et que le lecteur en connaisse très probablement les grandes lignes, nous souhaitons soulever quelques enjeux liés à l'élaboration des algorithmes ainsi qu'à leur traitement informatique.

1.1.1. Pourquoi le traitement d'image ?

« Traitement d'image » est un terme assez vague, et il semble opportun de commencer par définir les termes.

Définition 1.2 (Image (Larousse))
Ensemble de points ou d'éléments (pixels) représentatifs de l'apparence d'un objet, formés à partir du rayonnement émis, réfléchi, diffusé ou transmis par l'objet.

Insistons sur les mots « *apparence* » et « *rayonnement* ». Le rayonnement est une grandeur physique et peut être de nature aussi bien mécanique (ondes acoustiques d'un échographe), optique (lumière ambiante ou infrarouge), magnétique (résonance magnétique utilisée par certains microscopes) ou autre. Le mot *rayonnement* justifie à l'essentiel la discipline de traitement d'image: la création d'une représentation n'est pas *invasive* dans la mesure où elle ne modifie pas l'objet. Par ailleurs, il n'y a pas de contact direct entre l'objet et l'observateur. Enfin, dans de nombreux cas, elle est *passive*: l'environnement propre de l'objet étudié n'est pas modifié.

L'apparence ne signifie pas qu'il s'agit d'une réalité absolue de l'objet, mais d'une *représentation*. Elle est naturellement dépendante de nombreux paramètres extérieurs, comme le type de rayonnement, les positions des sources de rayonnement et d'observation, ... Le mot *apparence* contient donc une grande partie du travail induit par la représentation qui pourrait être résumé ainsi: déduire une réalité à partir de l'apparence.

Définition 1.3 (Traitement (Larousse))
Ensemble des opérations relatives à la collecte, à l'enregistrement, à l'élaboration, à la modification,

à l'édition, ... de données.

Mettons de cotés les termes *enregistrement* et *édition*. Le principe général du traitement d'image est donc à quelques détails près toujours le même (cf. figure 1.1): un système reçoit des images, y applique un traitement, et produit une information de nature liée à l'application visée.

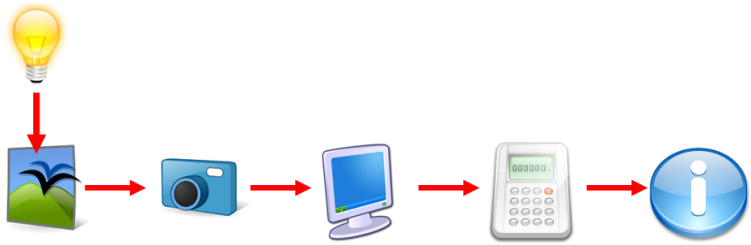


Fig. 1.1.: Les principes du traitement d'image: une source de rayonnement envoie des ondes sur un objet, qui sont ensuite réfléchies et collectées par un capteur. Le capteur transforme ces ondes en un ensemble de points. Ces points sont traités et une information est produite en sortie du système.

Ce que l'on appelle *traitement d'image* intervient donc après un processus complexe de *rayonnement* couplé avec celui de *collecte*. C'est pourquoi un traitement est lié à une représentation précise et ne peut en être dissocié.

Les opérations mentionnées dans la définition sont envisagées de manière informatique sous forme de programmes. Un programme est une substance *immatérielle* pouvant être remplacée sans nécessairement impliquer le changement du support matériel de fonctionnement. D'un point de vue industriel, cela signifierait en théorie qu'il est possible d'avoir plusieurs programmes avec une même machine. Voici quelques exemples: des voitures équipées d'un système de vision unique auraient un traitement de détection de posture et un autre de classification d'occupant. Pour de la vidéosurveillance embarquée à l'aide de caméras intelligentes - caméras avec processeur(s) de calcul - plusieurs modes selon les événements, l'heure de la journée, etc. pourraient exister. Enfin un traitement pourrait être déployé et amélioré par la suite. Ces remarques tendent à faire croire que la souplesse liée au support du traitement est un atout important.

Selon la terminologie « traitement d'image », il n'y a pas d'indication sur la méthode de résolution des problèmes. Le traitement d'image a ainsi suscité des théories très diverses de résolution de problèmes typiques de vision. Nous n'allons pas nous étendre sur l'ensemble de ces théories et allons recentrer la discussion autour de la Morphologie Mathématique.

1.1.2. Traitement d'image et Morphologie Mathématique

La MM se distingue des autres disciplines de traitement d'image par ses fondements théoriques. Des applications de *géostatistique* ont dans un premier temps conduit à la création d'une théorie particulière, et ont donné naissance à des opérateurs de traitement. Ces opérateurs ont d'abord été utilisés pour analyser des distributions en taille de grain (granulométrie), et pour analyser des textures. Le potentiel de ces opérateurs était tellement important que cela a incité leurs créateurs, Matheron & Serra, à y dédier une discipline: ce fût la naissance de la *Morphologie Mathématique*. Rapidement, la MM répondit avec succès à divers problèmes de géostatistique. L'article de Matheron & Serra [MS98] discute plus en détail des origines de la MM. Le champ d'application fût étendu par la suite à des applications de vision que l'on connaît aujourd'hui (multimédia, matériaux, médecine, biologie, etc.).

Par ses fondements, la MM repose sur sa théorie **ensembliste** et **topologique**. Les notions topologiques déployées par Matheron [Mat69] - intérieur, adhérence, compact, topologie myope, etc. - sont

cependant difficilement transposables en traitement informatique dans la mesure où l'espace de travail est discret.

Les deux notions essentielles, le **voisinage** et l'**adjonction**, sont toutefois suffisamment riches pour fournir un ensemble large et cohérent de traitements. Ces deux notions font par ailleurs la spécificité actuelle de la MM vis-à-vis d'autres disciplines de traitement d'image. Le *voisinage* est lié à la manière de construire des ensembles ou sous-ensembles. L'*adjonction* est plus complexe à décrire: elle est liée à la notion d'ordre d'un espace. Elle garantit en outre la stabilité de certains traitements (ouverture et fermeture), implique un traitement dual du *fond* et de la *forme*, etc. Il est intéressant d'insister sur le fait que ces deux notions sont indépendantes des espaces dans lesquels elles sont exprimées.

Enfin, la MM s'est avérée être une discipline particulièrement efficace dans la résolution de nombreux problèmes, dans de nombreux secteurs. Récemment, Walter [Wal03] a développé et utilisé des outils morphologiques pour la détection de lésions à l'intérieur de l'œil, pour la prévention de la rétinopathie diabétique. Fricout [Fri04] a utilisé des descripteurs morphologiques et des méthodes morphologiques de traitement (filtrages, segmentations, ...) pour la classification de tôles d'acier. Le domaine d'application de la MM est en constante expansion depuis presque une trentaine d'années.

1.1.3. Les enjeux

D'après la présentation du traitement d'image et de la MM, nous voyons clairement que les développements inhérents au traitement informatique sont intimement liés à la partie amont du système global de traitement. Cette partie concerne la représentation de l'objet d'étude. La dépendance se trouve principalement à deux niveaux:

1. **collecte de données** : la collecte de données pour la construction des images (ou acquisition) est un processus expérimental, et induit naturellement une incertitude sur les données. À l'incertitude classique des conditions d'acquisition maîtrisée, s'ajoute celle plus récente les conditions *réelles*, c'est à dire lorsqu'il est impossible d'émettre des hypothèses sur la partie en amont du capteur. L'utilisation de données comme la couleur, ou plus généralement des connaissances *a priori* sur les objets d'intérêt (géométriques, ...) peuvent rendre les traitements plus robustes.
2. **variété des données et indépendance algorithmique** : nous avons souligné le fait que les fondements de la MM permettent une indépendance vis à vis des espaces de représentation des images, que cela soit au niveau du système de coordonnées ou de la nature des points (pixels). Le fait de rendre les algorithmes indépendants des données est un atout intéressant, tant au niveau exploratoire (application de traitements connus sur de nouveaux espaces), qu'au niveau applicatif (base algorithmique similaire, quelles que soient les contraintes données par l'application).

Ces deux points convergent vers l'enrichissement de la partie algorithmique en MM, que nous nous proposons d'étudier dans cette thèse.

Les outils de programmation générique semblent apporter une solution intéressante au deuxième point ci-dessus. Comme nous allons le voir, certains mécanismes de méta-programmation permettent d'utiliser à bon escient le compilateur, et de rendre plus ou moins indépendant les algorithmes et les données que ces derniers traitent. Le processus de développement qui incombe au chercheur en est grandement amélioré. Dans des langages plus classique tel que le « C », il existe une dépendance très importante entre l'algorithme et les données qu'ils traitent. Cela a pour conséquence une redondance algorithmique très importante, obligeant le chercheur à développer par exemple une même fonction pour chaque format d'image. La *méta-programmation* répond à ce genre de problème.

Nous allons créer un système algorithmique à l'aide d'outils de développements génériques. Ces outils sont la *méta-programmation* et la *programmation orienté objet*. Ce système permettra effectivement d'arriver à l'indépendance des algorithmes vis-à-vis de la représentation des données. Nous illustrerons ce fait à travers l'expression et le développement d'algorithmes dans ce système d'une part, et leur application sur des espaces couleur ou nD d'autre part.

L'apport de nouvelles informations (dimension, taille des images, précision, couleur, nouveaux artefacts, etc.) est également un moteur de développement algorithmique. La couleur, ou plus généralement le traitement multi-spectral, est un exemple tendant à se démocratiser bien que son utilisation en MM soit encore très récente. L'introduction de connaissances supplémentaires lors de l'exécution d'un algorithme, de manière à mieux contrôler son comportement, est un autre exemple. Cela est une manière de répondre au premier point évoqué ci-dessus.

Comme nous allons le voir dans la partie suivante, les projets d'étude illustrent chacun de ces propos.

1.2 Cadre des projets

Nous explicitons ici de manière très succincte les différents projets qui ont motivé ou accompagné les travaux présentés dans ce manuscrit. Cette présentation a pour vocation de faire le lien entre des travaux algorithmiques, qui peuvent sembler parfois hermétiques, et leur réalité applicative.

Les développements présentés dans ce mémoire ont été motivés par la réalisation de deux projets, qui sont:

1. **COSMECA** - *COntinuité de Service de Mobilité Évaluée dans un Contexte Automobile*. Il s'agit d'un projet *RNRT* assez vaste à propos de l'amélioration des services fournis dans une voiture. Nous avons travaillé sur la *détection d'hypovigilance*: l'objectif est de surveiller les pertes de vigilance du conducteur pendant la conduite.
2. **CLOVIS** - *Composants LOgiciels pour la VIdéoSurveillance*. Ce projet *Eureka* a pour objectif le développement d'un cadre dédié à la vidéo-surveillance. Notre contribution a porté au développement algorithmique de détection de mouvement de personnes.

Outre ces applications industrielles, Romain Lerallut et moi-même avons développé une bibliothèque informatique de traitement d'image, et plus particulièrement de Morphologie Mathématique, portant le nom de « **Morphée** ». Cette bibliothèque tire profit des outils informatiques nouvellement disponibles - des compilateurs performants - et est particulièrement bien adaptée à la recherche algorithmique. Les développements issus des différents projets sont venus naturellement nourrir cette bibliothèque.

Voici les environnements spécifiques à ces applications.

Sécurité automobile - Projet COSMECA

Certaines études récentes [DG98, RPM01] ont mis en évidence une mesure physiologique permettant de déduire l'état de vigilance d'une personne. Cette mesure se nomme le *PERCLOS* et est défini de la manière suivante:

Définition 1.4 (Perclos - [DG98])
Le PERCLOS est le pourcentage du temps de recouvrement total de la pupille par la paupière. La mesure est représentative des lentes fermetures des yeux, plutôt que des clignements [(i)].

Le traitement d'image est bien adapté pour les problèmes de sécurité automobile, dans le sens où les capteurs sont sans contact et fonctionnent en lumière visible [(ii)].

La particularité principale de cette application est la grande variabilité des conditions d'illumination. Le système embarqué dans le véhicule reçoit de grandes fluctuations de lumière lors du déplacement du véhicule, comme le montre la figure 1.2.

La mesure même de PERCLOS conduit à un problème de détection des yeux. Ce problème peut être envisagé de plusieurs manières. Nous avons travaillé principalement sur le problème de détection du

(i) de l'anglais *« PERCLOS is the percentage of eyelid closure over the pupil over time and reflects slow eyelid closures (« droops ») rather than blinks »* - traduction libre

(ii) Nous n'avons pas considéré la lumière infrarouge, qui donnerait des résultats très intéressant, puisque les yeux réfléchissent le rouge. L'impact sur la fatigue des yeux, que provoquerait l'utilisation d'une source lumineuse infrarouge, doit préalablement être étudié avant toute utilisation; ce qui n'a pas été fait par les partenaires industriels.

visage dans ces conditions difficiles, permettant ensuite d'extraire la position des yeux. Les détections sont effectuées grâce à la couleur de la peau, et à l'outil de segmentation de ligne de partage des eaux. Nous ne proposerons pas une résolution complète de l'application de détection de PERCLOS. Au lieu de cela, nous nous focaliserons sur la détection robuste de visage dans des conditions d'illumination fortement variables.

Fig. 1.2.: Extraits de la base d'acquisition « Logitech ». Cette base est un extrait de nos propres acquisitions à partir d'un système simple: une caméra de type Webcam est fixée sous le rétroviseur et orientée vers le conducteur.

Vidéo-surveillance - Projet CLOVIS

> « La pire besogne a toujours été accomplie avec les meilleures intentions. »
> *Aphorismes*, O. Wilde.

L'objectif général est de développer des outils en relation avec des tâches courantes en vidéosurveillance, pour un réseau de caméras dites *intelligentes*. Dans ce cadre, chaque caméra est matériellement ou virtuellement associée à un processeur, et peut communiquer à l'aide d'un réseau informatique avec d'autres caméras ou des opérateurs.

À l'instar de l'application précédente, nous n'avons pas ici d'hypothèses forte ni par rapport aux conditions d'acquisition, ni par rapport aux objets devant être détectés. La qualité des acquisitions est également très variable: certaines séquences sont très peu bruitées et présentent des couleurs riches et fiables. D'autres sont plus mauvaises, et ont à la fois une luminance bruitée et des couleurs peu fiables (peu chromatiques). Quelques extraits - peu révélateurs des difficultés - sont données sur la figure 1.3.

Nous n'aborderons pas l'ensemble des développements réalisés dans le cadre de ce projet. Nous nous intéresserons particulièrement aux améliorations - par rapport aux approches classiques - de la détection des objets en mouvement grâce à l'utilisation de la couleur.

Morphée

Un ouvrage tel que celui de Soille [Soi99] - et son succès dans la communauté de MM - souligne l'importance d'avoir un recueil algorithmique de référence. *Morphée* est né d'un besoin totalement comparable: avoir à disposition un ensemble d'outils informatiques permettant d'une part l'implémentation efficace et correcte d'algorithmes existants, d'autre part la création « facile et rapide » d'algorithmes nouveaux. *Morphée* est devenu ainsi une librairie de traitement d'images complète.

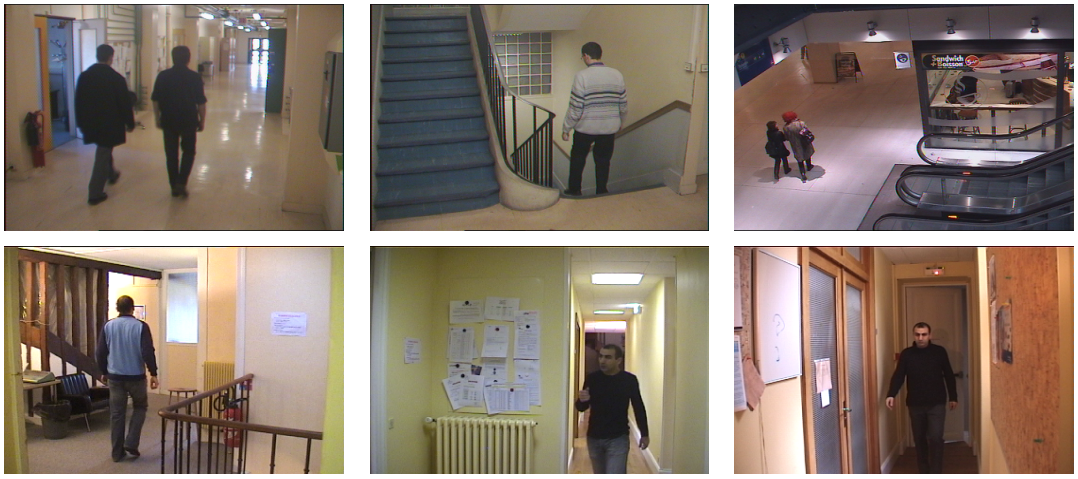


Fig. 1.3.: Images « CEA » (première ligne) et « CMM » (seconde ligne).

Les développements inhérents à Morphée possèdent un intérêt scientifique pertinent, et pour cela nous nous proposons d'en exposer les points clefs. L'intérêt se manifeste à plusieurs niveaux:

Programmation & pédagogie Conscients du fait que la programmation est parfois une discipline intimidant le chercheur (selon nos observations), nous pensons que la crainte principale concerne l'aspect technique du langage utilisé. Certains langages permettent d'écrire des équivalents informatiques aux notions théoriques manipulées par le chercheur. Le fait d'éclaircir ces équivalences transforme la programmation en un support pédagogique pertinent.

Connaissance des algorithmes Connaître l'existant est un moyen d'appréhender de nouveaux développements. L'étude des algorithmes est, à notre sens, un moyen simple de connaître l'existant. En effet, un algorithme est la résolution d'un problème complexe, mais selon une démarche éclatée en étapes élémentaires. Chacune de ces étapes est suffisamment simple pour leur compréhension immédiate. La compréhension va donc du complexe (algorithme, connaissance globale) vers le simple (étape). La démarche inverse est toutefois plus délicate car, précisément à cause de cette proximité avec le détail, la compréhension est confinée sur des points précis.

Extension de l'existant, création Par la maîtrise des détails algorithmiques, les variations potentielles apparaissent plus clairement. Cela donne naturellement naissance à de nouveaux algorithmes, de nouvelles méthodologies, etc.. Cela permet également et dans certains cas de formaliser plus précisément un type de traitement. Enfin, un algorithme est un *exemple* de résolution d'un problème complexe et, comme tout exemple, apporte ainsi des éléments de réponse à d'autres problèmes.

Réponse à des problématiques réelles L'utilisation de la bibliothèque *Morphée* n'est pas confinée dans l'activité de recherche. Cette bibliothèque est utilisée avec succès dans un nombre croissant d'applications industrielles (qui, pour des raisons de confidentialité, ne seront pas énumérées ici).

Tous ces points justifient l'axe de la présentation choisi dans ce mémoire.

1.3 Plan du mémoire

Le document s'articule autour des algorithmes en morphologie mathématique et en traitement d'image. Les projets industriels introduits précédemment serviront de support d'illustration aux développements algorithmiques. Les sujets suivants seront traités:

- Les méthodes de programmation offrant de nouvelles perspectives algorithmiques.
- Un exemple concret d'application de la **généricité algorithmique** pour le traitement en $\mathbf{n}$ **dimensions**, à travers l'étude d'une fonction de distance exacte.

- Des développements relatifs à la **couleur** et au traitement **multi-spectral** des données.
- Enfin la problématique de **segmentation**, envisagée de manière générique et contrôlée.

Le plan détaillé du manuscrit est exposé ci-après.

Le chapitre 2 reprend les principes de la programmation objet et de la programmation générique (ie. méta-programmation) pour les appliquer aux notions de MM. Ce chapitre est assez spécifique à la programmation. Cela conduit à l'élaboration de structures informatiques génériques, offrant un cadre nouveau et puissant. Parmi ces améliorations, la possibilité de traiter des images en dimension quelconque - en coordonnée ou en type - motive le développement d'algorithmes génériques. Enfin l'écriture algorithmique dans ce cadre générique est également modifiée; le chapitre souligne certaines méthodes, par rapport aux structures informatiques précédemment introduites, tirant profit de ce cadre.

L'indépendance vis à vis de la dimension des images nous conduit à proposer, dans le chapitre 3, un algorithme pour le calcul de distances exactes. Une étude théorique préalable nous permet d'émettre des conditions sur la métrique utilisée, et d'établir les étapes nécessaires pour obtenir l'exactitude de l'image des distances. De ceci, nous déduisons l'algorithme, indépendant de la dimension et sous condition de la distance et nous détaillons son comportement. Les distances morphologiques sont une autre approche dans la construction d'image de distance. Nous nous intéresserons particulièrement à une forme numérique de distance morphologique appelée « quasi-distance », récemment introduite par Beucher [Beu05]. Nous proposerons un algorithme améliorant les temps de calcul de l'algorithme existant. Enfin nous élargirons les possibilités de la quasi-distance aux images multi-valuées.

La couleur est un élément discriminant important. Son utilisation en Morphologie Mathématique n'est toutefois pas aisée. Nous envisageons plusieurs approches pouvant être ensuite facilement étendues à des espaces plus grands, ce que nous nommons images *multispectrales*. La première approche est liée à la notion de métrique pour la définition des opérateurs de gradients, permettant de passer de l'espace multi-spectral à une représentation scalaire. La seconde approche utilise des statistiques circulaires locales, dans l'espace couleur cylindrique HLS. Enfin, la troisième approche sera celle des treillis lexicographiques. Nous étudierons la manière d'adapter les algorithmes pour utiliser ce type d'ordre. Ces développements seront ensuite utilisés dans la résolution de deux problèmes : la détection de peau robuste aux changements d'illuminant, et la détection d'objets en mouvement grâce à la couleur.

La segmentation joue un rôle clef en Morphologie Mathématique. L'algorithme par excellence est la ligne de partage des eaux. Dans le chapitre 5, nous allons revoir en détail l'algorithme à base de files d'attente hiérarchiques, développé initialement par Meyer dans [Mey91]. Nous soulignerons les quelques biais de l'algorithme initial et en proposerons les corrections. Le cadre informatique développé au chapitre 2 montrera l'indépendance par rapport à la dimension de l'image et la nature du relief topographique. Nous verrons ensuite comment modifier l'algorithme original pour inclure, lors de la propagation, des contraintes autres que celles données par le relief topographique. Nous proposerons deux méthodes, l'une basée sur une mesure de la région inondée, l'autre sur une mesure géométrique locale du front de propagation.

Enfin, les conclusions de ces travaux seront énoncés dans le chapitre 6. Nous y développerons également les perspectives proches et lointaines des résultats présentés.

CHAPITRE 2

Morphologie Mathématique et méta-programmation

> « La programmation par objets est une idée exceptionnellement mauvaise qui ne pouvait naître qu'en Californie. »
> Edsger Wybe Dijkstra

Ce chapitre a pour objectif la présentation de développement d'opérateurs nouveaux de morphologie mathématique, expliquée également par des outils de programmation nouveaux.

Nous présenterons dans un premier temps ce que l'on appelle la « *méta-programmation* », et les mécanismes que cela offre dans l'écriture de structures informatiques de stockage ou de traitement. Nous envisagerons ensuite les outils informatiques utilisés en Morphologie Mathématique sous le point de vue de la méta-programmation. L'objectif est d'exprimer ces outils en s'inspirant de leur expression classique, mais en rendant leur champ d'utilisation extrêmement large. Ceci se fera grâce à un rapprochement vers les fonctionnalités premières des dits outils, et par l'utilisation de types abstraits. L'approche impacte l'ensemble des notions manipulées dans les traitements algorithmiques, que cela soit les unités de stockage, la construction d'ensemble, les ordres, etc.. Les avantages d'une telle approche seront soulignés dans ce chapitre, et seront démontrés et justifiés à travers l'élaboration de nouveaux algorithmes dans les chapitres suivants.

Nous sommes conscients que le terme « nouveau » restreint le champ d'application de cette approche dans le temps, et que dans quelques années ces développements seront certainement désuets. Néanmoins, l'approche choisie ici fait davantage référence à une organisation particulière des idées et des concepts fondamentaux de morphologie mathématique, ou plus globalement de traitement d'image, qu'à la maîtrise des outils permettant d'y parvenir, qui eux, sont susceptibles de changer rapidement.

2.1 Introduction à la méta-programmation

La définition de base de « méta-programmation » est l'écriture de programmes à l'aide de programmes. Plusieurs outils sont à disposition pour effectuer une telle tâche, et parmi ces outils l'utilisation de « patron »[(i)] s'impose par sa simplicité, sa puissance et ses performances (voir par exemple [Mey01] et [Str97]). Le second programme nous permettant de générer notre programme est donc dans ce cas un compilateur. L'utilisation de template est rendu possible par l'apparition de compilateurs puissants et de langages orientés *programmation objet* et *méta-programmation*. L'outil de référence aujourd'hui est le *C++*, mais l'organisation et la capitalisation des idées et des concepts ne sont pas liées à l'outil.

[(i)] de l'anglais « template », nous utiliserons indifféremment l'une ou l'autre des tournures

Si nous devions apporter une définition personnelle à la méta-programmation de template, nous dirions c'est une méthode de programmation informatique dont la force majeure est d'orienter l'écriture sous forme de concepts. C'est la raison pour laquelle nous nous attarderons plus à décrire la manière dont nous avons structuré notre librairie de traitement d'image et nous n'attacherons pas une importance capitale à l'écriture des algorithmes en code pur.

L'articulation de cette partie est la suivante. Dans un premier temps, nous allons reprendre quelques notions de Programmation Orienté Objet (*POO*) et introduire les notions informatiques essentielles pour la suite. Ensuite, nous allons reprendre les structures informatiques utilisées en traitement d'image, et particulièrement en Morphologie Mathématique, et allons les envisager de manière générique et conceptuelle. Nous décrirons d'abord les structures, et ensuite l'impact de ces approches sur l'algorithmique.

2.1.1. Avant-propos sur la programmation objet

Nous abordons dans cette partie une explication nécessaire de l'approche objet, telle qu'envisagée dans le reste du chapitre. Nous débutons notre discussion sur les objectifs de la programmation objet, puis nous présenterons quelques unes de ses entités informatiques.

Objectifs de la POO

L'approche objet en programmation se base sur la séparation des rôles et des fonctions des acteurs liés aux traitements informatiques. La séparation est d'ordre conceptuelle, structurale et fonctionnelle, guidée par les contraintes suivantes:

1. Pouvoir utiliser un objet sans le connaître en détail. Ce n'est pas seulement une contrainte de travail d'équipe, mais principalement une contrainte de modélisation. On ne s'intéresse alors qu'aux propriétés et fonctionnalités de l'objet, c'est à dire plus ou moins à ce qu'il est sensé faire, comment il réagit, etc. Il est donc possible d'utiliser un objet s'il répond à nos besoins et si nous arrivons à communiquer avec lui. Cela rejoint en quelque sorte les concepts de « *boîte noire* », ou encore de « *brique de base* ».
2. Les modifications ne doivent pas affecter toute la librairie mais seulement les objets concernés. Les objets possèdent des rôles cloisonnés. La programmation purement fonctionnelle - qui ne se définit que par ses actions - est particulièrement sensible au changement d'une action élémentaire puisqu'elle remet en question toute la chaîne de traitement connectée à cette action. Au contraire, les modifications sur le traitement des données, en *POO*, restent essentiellement à l'intérieur de l'unité de traitement. Les autres objets ne sont pas affectés pas ces modifications si les données en entrée et en sortie de l'objet modifié restent inchangées (en nature).

La Programmation Orienté Objet (*POO*) regroupe la représentation et le traitement des données, la principale contrainte est de faciliter les modifications - sur la représentation ou le traitement. En effet, d'une part les éléments de modification sont centralisés dans l'objet, d'autre part le comportement du reste du système face à l'objet est peu affecté. La portée de la *POO* est cependant beaucoup plus importante: l'usage de la *POO* nécessite de nouvelles notions que sont « modélisation », « classes », « encapsulation » et « interface ». Par ailleurs, le fait de cacher certaines parties des objets, de manière à ce qu'ils soient considérés comme des boîtes noires identifiées par leur comportement, modifie en profondeur l'approche de la programmation informatique. Nous allons décrire succinctement les points importants de la *POO*, puis nous décrirons les modifications spécifiques à la méta-programmation.

Éléments de POO

Classes Une *classe* est une entité possédant un *nom*, des *attributs* et des *méthodes*. Une instance est une réalisation particulière d'une classe: les méthodes dont bénéficie l'instance sont celles de la classe, mais l'état de l'instance, c'est à dire l'ensemble de ses attributs est une réalisation particulière de la classe. L'ensemble des attributs de l'objet définit son *état*, l'ensemble des méthodes son *comportement*. Il est possible de confondre les mots *objet* et *instance* dans certains cas.

Signature ou interface le fait de cloisonner, ou plutôt d'*encapsuler* les rôles des objets sépare l'implémentation et la vision du système sur ces objets. Un objet est défini alors par son rôle d'une part, mais également par les éléments qu'il « montre » au reste du système. Cette visibilité est nommée *signature* ou *interface* de l'objet. Indirectement, il s'agit d'un protocole de communication entre l'objet et le reste du système. Nous qualifierons cet échange de *communication structurelle*.

Les interfaces en *C++* présentent deux niveaux d'approche. Le premier niveau est extrêmement souple et concerne la « *méta*-signature » template, non typé. Ce premier niveau est utilisable par le compilateur, et sera le cœur du débat de l'écriture algorithmique de ce chapitre. On parlera alors plus volontier de *signature*. Le deuxième niveau est rigide et fixe car fortement typé. Il permet entre autre de proposer une librairie exécutable et directement utilisable par des tiers. C'est ce que nous appelons *interface* en *C++*, même si finalement il s'agit d'une signature à un niveau différent d'utilisation. Bien que nous ayons soigné ce type d'interface avec diligence et insatiable minutie, le discours qui y est lié concerne d'avantage l'architecture logicielle et est hors propos dans le contexte choisi; c'est pourquoi il sera volontairement omis.

Dans la mesure où l'interface d'une structure méta-programmée est non typée, l'utilisateur adopte un point de vue totalement différent sur le rôle et l'utilisation de cette structure. Le typage est un effet de bord de la programmation des langages typés, dans la mesure où il confine les comportements des structures sur les types connus. Ainsi, la phrase courante :

« *retourne un entier correspondant au pixel* (x_1, x_2) »

n'a aucun sens pour une image à précision réelle ou à trois dimensions. L'abstraction du type (méta-typage) permet plutôt de décrire la même fonction de la manière suivante:

« *retourne la valeur* v *du pixel* p »

Dans ce second cas de figure, les types de v et p interviennent peu dans la description fonctionnelle. Ils sont par contre dépendants du type de l'image. Heureusement, le *C++* permet de communiquer ces types aux structures qui en font la requête. C'est ce que nous avons nommé précédemment la *communication structurelle*.

Libéré de cette contrainte forte, la conception des structures informatiques (structures de donnée, algorithmes) se rapproche de l'aspect fonctionnel. Nous reviendrons régulièrement sur ce point au cours du chapitre.

Dépendance et modélisation Lorsqu'un objet O_1 a besoin d'un objet O_2 pour son fonctionnement propre, O_1 est dit *dépendant* de O_2. Nous dirons plus volontiers que O_1 est **client** de O_2. Cette relation de dépendance implique que si la signature de O_2 est modifiée, l'ensemble des clients de O_2 - dont O_1 - en seront affectés. Ces modifications sont bien sûr indésirables, d'autant plus que leur nuisance augmente avec la taille de la librairie. C'est la raison pour laquelle une étape antérieure de modélisation est souvent nécessaire à l'architecture de la librairie. La modélisation distribue les rôles entre les différentes entités informatiques. À partir de cela, il est possible, pour chacune des structures concernées, d'élaborer une signature *nécessaire et suffisante* au fonctionnement d'un objet et à l'exécution de son rôle vis à vis de son environnement. Le choix de la modélisation suit souvent une certaine réalité *physique*. Ce qu'il apparaît de ces remarques est qu'une définition claire du rôle d'une structure informatique est un pré-requis nécessaire. C'est pourquoi dans la mesure du possible nous n'omettrons pas d'expliciter les rôles des différents acteurs algorithmiques.

2.1.2. Méta-programmation

Structure de données génériques et librairies tierces: Parmi les outils de programmation, il existe des librairies gratuites et de licence non-restrictive, contenant un certain nombre de structures génériques. Dans le cas du *C++*, la « *Standard Template Library* » (STL) est une des librairies venant généralement avec le compilateur, et disponible sur un nombre important de plate-formes. Pour une programmation performante de la *STL*, nous renvoyons le lecteur aux ouvrages suivants: [Mey01] et

[Ale01]. Nous avons utilisé cette librairie dans nos implémentations car elle répond à nos contraintes de licence, d'efficacité et de généricité.

Les structures présentes dans la *STL* répondent à des cas d'utilisation récurrents. Prenons pour exemple le vecteur: ce type est dédié à l'accès indicé des éléments qu'il contient. Il s'apparente ainsi à son utilisation mathématique; il se distingue des autres types non par les types des éléments qu'il contient (puisque la structure est générique), mais par l'accès immédiat qu'il offre à ses éléments. Voici les structures d'intérêt de la *STL*:

Vecteur : Le *vecteur* permet un accès immédiat à ses éléments, mais ne permet pas une insertion rapide (en $O(1)$) d'un nouvel élément entre deux éléments existants. C'est donc une structure adaptée pour l'accès indicé ou séquentiel, mais non adaptée pour un changement de dimension.

Liste chaînée : La *liste chaînée* se distingue du vecteur par le fait qu'elle offre un comportement d'insertion et de suppression immédiat, mais un temps d'accès linéaire (par rapport au nombre d'élément que la structure contient). Elle sera choisie lorsque l'insertion d'éléments primera sur les accès. La prise en compte de ces comportements est assez important pour l'usage que nous voudrons en faire. Ils conditionnent en effet les performances de l'algorithme en terme de temps de calcul.

Dictionnaire : [(ii)] Les dictionnaires sont des structures de donnée acceptant deux types: un type de stockage « T » et un type de **clef** « C ». La clef est un type bénéficiant d'une relation d'ordre. La structure interne est souvent assez complexe, organisée sous forme d'arbre de recherche, de manière à garantir des temps optimaux de recherche (en complexité convergeant vers $O(\ln N)$, N étant le nombre de clefs présentes).
Selon la même philosophie que pour les structures génériques *vecteur* ou *liste chaînée*, les types de stockage « T » et de clef « C » sont génériques. Pour pouvoir fonctionner correctement, le type clef doit être muni d'une relation d'ordre, ou si la structure n'est pas munie directement de cette relation d'ordre, une fonction d'ordre doit informer le dictionnaire. Ainsi, si le type clef est un vecteur de dimension P et un opérateur d'ordre sur ces vecteurs est fourni, l'utilisation du dictionnaire est alors possible en utilisant comme clef les vecteurs de dimension P.
Puisque cette structure ordonne naturellement les clefs, nous l'utiliserons pour définir des files d'attente hiérarchique génériques en §2.2.6: la hiérarchie sera alors liée au type clef. La souplesse de la fonction d'ordre permettra en outre d'utiliser facilement des relations d'ordre complexes, tels que les ordres lexicographiques.
Enfin le type clef « C » doit être sur un espace totalement ordonné, mais la relation d'ordre est de type *stricte* et *faible*; cela signifie entre autre que pour générer un ordre total, la notion d'équivalence stricte n'intervient pas. En effet, dans le cas *faible*, si nous notons $<_{C,w}$ l'ordre stricte et $\equiv_C$ l'équivalence faible, nous avons [(iii)] :

$$\neg(a <_{C,w} b) \wedge \neg(b <_{C,w} a) \Leftrightarrow a \equiv_C b$$

Ce point est important dans le sens où il sera possible d'ordonner des espaces même si ceux-ci manquent de précision (égalité stricte inexistante). Les nombres en virgule flottante présentent ce défaut.

Conformément au principe d'encapsulation, les détails d'implémentation sont volontairement voilés. Nous reportons le lecteur aux ouvrages de références [Str97] et [Mey01]. Le but n'est pas de détailler la manière dont sont écrites ces structures, mais de préciser la distinction comportementale nous permettant ensuite de les utiliser dans des structures complexes adaptées à nos besoins. À noter également que ces structures répondent à un standard, et que ce dernier garantit les complexités (du moins asymptotiques).

(ii) de l'anglais « *map* »
(iii) cf. page xiii pour l'ensemble des notations

La librairie *Boost* [(iv)] propose également un certain nombre de projets très intéressant pour le développement de notre librairie. Elle propose en outre des mécanismes de méta-programmation extrêmement puissants, une structure de graphe très riche et une librairie permettant de faire le lien entre une librairie compilée et le langage de script *Python* [(v)] . Nous nous servirons beaucoup de cette librairie mais contrairement à la *STL*, son utilisation paraîtra beaucoup moins évidente. Elle interviendra peu pour la programmation directe des structures de donnée dans la librairie d'image, et servira essentiellement dans l'utilisation de mécanismes de méta-programmation (mécanismes que nous verrons un peu plus tard en §2.1.3).

Motifs de conception Les ouvrages tels que [Ale01, GHJV94] discutent des *motifs de conceptions* [(vi)]. Le but général des motifs de conception est de minimiser les interactions qui pourraient naître entre les différentes classes, et de proposer des solutions architecturales à des problèmes informatiques récurrents. Bien qu'ils adressent en réalité des problèmes de génie logiciel, les deux motifs de conception suivants seront activement utilisés lors de la conception de notre librairie.

Conteneurs & itérateurs: Un conteneur [(vii)] est une structure résultante de l'agrégation d'éléments *contenus*. Les conteneurs sont généralement des méta-structures, dont des exemples typiques sont les vecteurs ou les listes mentionnées précédemment (cf. §2.1.2).

Le concept d'itérateur naît de la fonctionnalité suivante: nous souhaitons parcourir tous les éléments contenus de manière séquentielle. L'avantage majeur des itérateurs est qu'ils cachent la structure interne du conteneur: ce dernier peut être simple ou complexe, la manière de le parcourir également, mais seules les fonctions permettant l'accès séquentiel sont visibles au système client. Les méthodes proposées par l'itérateur sont au nombre de trois:

1. accéder à l'élément courant
2. savoir si la totalité des éléments a été parcourue
3. passer à l'élément suivant

Ceci suppose que le conteneur correspondant fournisse une instance d'itérateur. En général, les développements d'un itérateur et de son conteneur sont simultanés. L'itérateur est une notion essentielle sur laquelle nous allons appuyer une grande partie des possibilités de notre librairie. Le point clef de l'itérateur est qu'il uniformise les accès à des structures de type conteneur.

Patron d'algorithme : [(viii)] Il arrive souvent qu'un algorithme soit en réalité une application particulière d'un algorithme initial, et dont les comportements diffèrent en un certain nombre de points facilement identifiables; les deux algorithmes définissent en fait une classe d'algorithmes possédant un tronc commun. Il est alors judicieux de séparer ce tronc de ses points de variation. Le *patron algorithmique* adresse ce type de développements: il contient les traitements et les comportements communs à une classe d'algorithme dans un *squelette algorithmique*. Ce patron algorithmique unique donne ensuite lieu à des algorithmes *concrets*, par association à des structures comportementales dédiées. Ces structures comportementales renseignent le fonctionnement du patron algorithmique en certaines étapes, précisément aux points où les variations comportementales peuvent exister. Cette approche possède de nombreux avantages, dont la réduction des duplications inutiles du patron, la capitalisation des modifications (une modification dans le patron affecte tous les algorithmes concrets déduits de ce patron), la réduction des développements des algorithmes concrets, etc.. Nous illustrerons cette démarche à travers l'algorithme de labellisation, au §2.3 page 38.

(iv) www.boost.org

(v) www.python.org : langage de script très riche et fonctionnel, dont la licence est également compatible avec le type d'exploitation que nous nous sommes fixé.

(vi) de l'anglais « *Design Patterns* »

(vii) de l'anglais « *container* »

(viii) de l'anglais « *template method* »

2.1.3. Mécanismes de méta-programmation

Nous abordons dans cette section certains mécanismes utilisés en méta-programmation. Ces mécanismes sont nombreux, nous orientons le discours selon deux points d'intérêt : d'une part la généralisation de l'écriture d'algorithmes par la création artificielle d'un moyen de communication entre les différentes structures informatiques, notamment par le renseignement des types; d'autre part la possibilité d'intégrer des cas particuliers en fonction de conditions flexibles.

Champs de type Les objets sont identifiés par leur fonctionnalités. Dans le cas de la méta-programmation, les fonctionnalités offertes par les structures ne concernent pas seulement leurs entrées/sorties, mais également des informations sur la structure informatique des instances de méta-structure. Supposons par exemple que nous ayons une méta-structure « s_meta_structure » fonction d'un méta-type « T », et proposant une méthode « call » prenant en entrée un argument de type « T ». Nous souhaitons déduire de « T » et de la structure « s_meta_structure » le type de retour de la méthode « call ».

Les champs de type interviennent sur ce point. On peut considérer cela comme un moyen offert aux méta-structures de communiquer des informations sur leur structure informatique lorsqu'elles sont instanciées.

Revenons sur notre exemple et supposons que notre méthode renvoie systématiquement un entier de type 8 bits UINT8:

```
1  template <class T> s_meta_structure {
2      typedef UINT8 result_type;
3      result_type  call (const T val){  /*  corps de la  méthode */ }
4  };
```

« result_type » ligne 2 est ce que l'on appelle un *champ de type*. Les clients de cette structure doivent stocker le résultat de l'opérateur « call » dans une valeur de type UINT8. Or le cas UINT8 n'est souvent qu'un cas particulier de l'ensemble des données que peut traiter le client, et le forcer à la valeur UINT8 réduit nécessairement ses fonctionnalités.

La solution réside dans les champs de type: les clients de « s_meta_structure » n'ont pas besoin de connaître exactement le type de retour, car ils peuvent le déduire lors de la compilation et si le champ de type « result_type » existe.

Prenons pour exemple une image $\mathcal{I}$, une fonction g dépendant d'une fonction f de la manière suivante:

$$g(\mathcal{I}) = \mathcal{I} - f(\mathcal{I})$$

appliquée à tous les éléments de $\mathcal{I}$. Nous souhaitons utiliser la même fonction g pour des fonctions f très différentes: f peut être le calcul de la moyenne de l'image, le minimum ou maximum, ou une fonction quelconque. Néanmoins, nous souhaitons écrire g de manière générique une seule fois. Pour arriver à ce but, il faut que l'image $\mathcal{I}$, f et g puissent communiquer *informatiquement* des informations sur leur structure, lors de la création d'une instance particulière de g en fonction de f et $\mathcal{I}$.

Prenons f comme étant la fonction « *calcul du minimum de l'image* ». Le minimum de l'image est du même type que les données de l'image. Si l'image intègre un champ « value_type », f peut être écrit de la manière suivante:

```
1  template <class Image> f {
2      typedef typename Image::value_type result_type;
3      result_type  call (const Image& im){ /* calcul  du minimum sur l'image im */ }
4  };
```

Le résultat de f est du même type que celui de l'image: les clients potentiels de f sont libérés de cette association de type - qui correspond à un cas d'utilisation particulier de l'ensemble des opérateurs f possibles - et n'utilisent que le champ de type « result_type » de f. Le mot clef « typename » est propre au *C++*, et permet de considérer un champ (ici « Image:: value_type ») comme étant un type.

g sera ensuite écrit de manière à utiliser à la fois le champ « value_type » de l'image, et le champ « result_type » de l'opérateur f. Puisque f est déjà fonction de l'image, nous pouvons supposer que les valeurs de g seront du même type de ceux de f (nous verrons lors des spécialisations que les valeurs de g peuvent en fait être fonction des deux types, image et f). g s'écrit alors:

```
1  template <class Image, class operateur_f> g {
2      typedef typename operateur_f::result_type result_type;
3
4      const result_type valeur_f;                    // déclaration de la constante de soustraction (valeur constante)
5      const Image &reference_im;                     // référence vers l'image $\mathcal{I}$
6
7      /* instanciation de g, avec initialisation de "valeur_f" sur "f(im)" */
8      g(const Image& im, const operateur_f &f) : valeur_f(f.call(im)), reference_im(im) {}
9
10     /* retour de g = im - f(im) en chaque point */
11     result_type call(const offset o) const {
12         return reference_im(o) - valeur_f;
13     }
14 };
```

Dans les quelques lignes ci-dessus, lors de l'instanciation de g (ligne 8), la valeur de $f(\mathcal{I})$ est stockée dans un attribut interne « valeur_f ». Cette valeur est globale et ne change pas lorsque l'on se déplace d'un point à un autre de $g(\mathcal{I})$, c'est pourquoi elle est calculée lors de la construction de g et assignée à une variable constante « valeur_f ». g stocke également une référence (pointeur) sur l'image $\mathcal{I}$. Ensuite, les clients de g sollicitent la valeur $g(\mathcal{I})$ en renseignant le point d'évaluation (11). Le point est donné par un type universel, que nous avons nommé « offset », et sur lequel nous reviendrons plus tard dans ce chapitre.

Spécialisations Nous avons montré jusqu'ici une manière très générale pour l'écriture d'algorithmes, de fonctions ou de structures. Cette généralité réduit parfois la flexibilité des développements. La situation classique est lorsqu'il existe une version plus efficace d'une structure ou d'un algorithme lorsque certaines conditions sur les types d'entrée sont réunies.

La *spécialisation* est un mécanisme de méta-programmation permettant d'intégrer certains cas particuliers dans la chaîne de compilation: elle permet de *préciser* une structure informatique lorsque certaines conditions sur les types d'entrée sont réunies. Plus précisément, supposons que nous ayons une fonction « s_alg » prenant pour entrée un type « T ». Supposons également que cette même fonction « s_alg » possède une version plus efficace que le cas générique lorsqu'utilisé sur les types *entiers* : le C++ permet d'écrire une spécialisation de l'algorithme « s_alg » pour intégrer ce cas d'utilisation.

Soit l'algorithme « s_alg » et sa définition:

```
1  template <class T> struct s_alg {
2          T call{/* ici le corps de l'algorithme */ }
3  };
```

Il est possible d'avoir une spécialisation de « s_alg » pour les types entiers 8 bits non signés (que l'on note UINT8), selon la manière suivante:

```
1  template <> struct s_alg<UINT8> {
2          T call{/* ici le corps de l'algorithme */ }
3  };
```

Dans ce cas précis, dès que la fonction « s_alg » est appelée [(ix)], le compilateur vérifie l'existence de spécialisations. Dans l'affirmative, il vérifie ensuite la concordance des paramètres de l'appel avec ceux de la spécialisation: la version spécialisée est alors utilisée lorsque cette concordance existe. Ce type de spécialisation est *explicite* puisque nous renseignons notre type UINT8, nous souhaitons à présent étendre cette spécialisation à l'ensemble des entiers.

Nous devons dans un premier temps distinguer les types entiers des autres types. Cela peut être également fait à l'aide d'une spécialisation explicite:

```
1  template <class T> struct t_isInteger {
2      enum {value = false};
3  };
4
```

[(ix)] Pour être plus précis, « s_alg » est nommé « *functor* ». Il s'agit en fait d'une structure ayant même objectif qu'une fonction, mais mieux adaptée à la méta-programmation. En effet, en C++ les functors offrent des champs de type, des spécialisations et des paramètres template par défaut, alors que le langage ne le permet pas pour les fonctions.

```
5  template <> struct t_isInteger<UINT8> { enum {value = true }; }
6  template <> struct t_isInteger<UINT16> { enum {value = true }; }
7  // spécialisation d'autres types entier
8  // ...
```

Par défaut la valeur associée à « t_isInteger <T>::is_integer » est *faux*, sauf si la structure est instanciée pour les types *entiers* pour lesquels nous l'avons spécialisé [(x)]. Cette structure annexe nous permettant de faire la distinction désirée, nous l'utilisons ensuite sur notre structure « s_alg » de la manière suivante, la version générique devient:

```
1  template <class T, bool b_is_integer = t_isInteger<T>::value >
2          struct s_alg {
3          T call { /* ici le corps de l'algorithme générique */ }
4  };
```

et la spécialisation pour les types entiers:

```
1  template <class T> struct s_alg<T, true> {
2          T call { /* ici le corps de l'algorithme dédié aux entiers */ }
3  };
```

La spécialisation se fait alors à travers une fonction « t_isInteger <T>::value »[(xi)] sur le type d'entrée « T »: cette structure tierce sert en quelque sorte de *fonction de type*, prenant un type en entrée et renvoyant un autre type ou une valeur. Ce mécanisme est extrêmement puissant et n'est limité que par l'imagination du programmeur et la qualité du compilateur. Nous utiliserons ce mécanisme dans notre librairie pour écrire des versions optimisées des files d'attente hiérarchiques, des spécialisations de quelques opérations morphologiques pour des types particuliers d'éléments structurants, des opérations sur pixels, etc..

2.2 Méta-programmation des structures de données en Morphologie Mathématique

Le traitement d'image est une discipline riche en outils de natures très diverses. Ces outils sont d'une part des outils très communs (stockage des images), mais répondent également aux besoins spécifiques d'une discipline. Par exemple, la base algébrique et topologique de la Morphologie Mathématique met en relief l'utilisation respectivement de relation d'ordre et de structure de voisinage. Enfin la perspective essentielle de l'informatique est le traitement des images à travers des fonctions, opérateurs ou algorithmes; dans ce sens il est naturel de présenter également les outils ayant une utilisation algorithmique directe (graphes, files d'attente, files d'attente hiérarchiques).

2.2.1. Les images

Ce paragraphe présente la structure d'image selon le point de vue de la méta-programmation. Nous rappelons la définition d'image et expliquons écriture par les méthodes de méta-programmation.

Les images représentent l'une des structures principales de stockage, l'autre moins utilisée aujourd'hui étant celle de *graphe*. Elle est la donnée informatique d'une fonction, $\mathcal{I}$, sur un ensemble d'étude $\mathbf{E}$, et à valeurs dans un ensemble $\mathbf{F}$. L'ensemble $\mathbf{E}$ représente des *positions* ou *coordonnées* et l'ensemble $\mathbf{F}$ les valeurs de $\mathcal{I}$ à chacune de ces positions:

$$\mathcal{I} : \mathbf{E} \rightarrow \mathbf{F}$$

$\mathbf{E}$ et $\mathbf{F}$ sont nécessairement des espaces séparables.

Cette définition, trop générale, n'est pas adaptée au traitement informatique. Aussi s'intéresse-t-on à la dimension de $\mathbf{E}$, que l'on assimile à la dimension de l'image. Une image dans le plan est ainsi une image bi-dimensionnelle ou $2D$, une image dans l'espace euclidien une image $3D$, dans l'espace

[(x)] Ce genre de mécanisme existe déjà dans la librairie *Boost* ce qui nous épargne de tels développements.

[(xi)] Il s'agit ici d'une fonction manipulée par le compilateur uniquement, dont l'unique dessein est de faciliter les spécialisations. C'est pourquoi nous l'appelons également « fonction ».

euclidien avec une information temporelle une image $4D$, etc.. $\mathbf{E}$ est donc assimilé à $\mathbb{R}^d$, d étant la dimension de l'espace. Malgré cette restriction, $\mathbf{E}$ reste généralement trop riche pour être traité facilement. C'est pourquoi on projette souvent $\mathbf{E}$ dans $\mathbb{Z}^d$: on discrétise l'espace de manière à pouvoir stocker l'image dans une mémoire informatique. On suppose souvent l'image comme ayant un support borné, il est donc possible d'y définir une origine et, par translation, de se restreindre aux plans positifs $\mathbb{N}^d$. Nous nous limitons à ce cadre dans toute la suite.

L'espace d'arrivée $\mathbf{F}$ a longtemps été uniquement binaire, c'est à dire à valeurs dans $\{0, 1\}$. Compte tenu de la structure des mémoires informatiques, chaque élément de $\mathbf{F}$ nécessite une certaine place de stockage en mémoire, mémoire limitée en nombre d'entités de stockage. Une valeur binaire nécessite une entité élémentaire de stockage: le bit. L'évolution de l'informatique, principalement concernant la quantité de mémoire disponible dans les ordinateurs, a permis une approximation de valeurs *scalaires*. Cette approximation a été faite d'abord par l'utilisation d'images à niveau de gris discret, sur une précision de 8 bits (soit $256 = 2^8$ valeurs) puis sur 16 bits, etc.. Aujourd'hui, il est courant d'avoir des images à précision beaucoup plus fine, à valeurs non plus entières mais « réelles ». Le mot réel est bien sûr une approximation, mais nous nous apercevons que plus l'espace de stockage des éléments de $\mathbf{F}$ est grand, plus il est possible d'approcher une précision demandée. L'arrivée d'images couleurs, ou plus généralement multispectrales [(xii)], augmente de nouveau les besoins: $\mathbf{F}$ n'est plus alors un espace scalaire mais un espace à p dimension, assimilé soit à $\mathbb{N}_{256}^p$, soit plus généralement à $\mathbb{R}^p$.

D'après ce qui précède, une image peut être considérée comme la valuation d'une fonction $\mathcal{I}$ à valeurs dans $\mathbf{F}$ et sur une grille régulière de dimension finie d; les ensembles de définition $\mathbf{E}$ - du moins sa dimension - et d'arrivée $\mathbf{F}$ peuvent alors être considérés comme des paramètres de cette structure. Nous emploierons souvent le terme de *générique* pour qualifier ce genre de structure: une grande partie des fonctionnalités peut être définie sans que l'on ait connaissance ni de $\mathbf{E}$, ni de $\mathbf{F}$. Ces fonctionnalités seront également souvent exprimées comme étant des variables de $\mathbf{E}$ et $\mathbf{F}$. $\mathbf{E}$ et/ou $\mathbf{F}$ varient selon des besoins externes (applications, données traitées, ...), mais puisque la structure reste identique, il est envisageable de coder cette structure indépendamment de $\mathbf{E}$ et $\mathbf{F}$. Nous verrons qu'il en est de même pour un grand nombre d'opérateurs et d'algorithmes. Ce qui est important ici est la séparation entre structure et instance: la structure d'image - ayant des propriétés intrinsèques - permet de stocker des instances d'image (par exemple une image à valeurs entière). Définir les opérations sur la structure permet de définir les opérations sur toutes les instances possibles, ce qui justifie l'écriture de la structure d'image par la méta-programmation.

Méta-programmation Le but d'une structure d'image est extrêmement simple, c'est pourquoi ses fonctionnalités seront assez restreintes.

L'image dépend conceptuellement de l'espace des coordonnées $\mathbf{E}$ et d'image $\mathbf{F}$. Cette dépendance se retrouve au niveau informatique: la structure d'image dépendra d'une classe de point (espace $\mathbf{E}$) et d'une classe de pixel (espace $\mathbf{F}$).

Puisque nous restreignons l'espace de coordonnées $\mathbf{E}$ à un grille régulière de type $\mathbb{N}^d$, la classe de point est entièrement déterminée par sa dimension d, la manière de coder chacune des coordonnées est par défaut *entière*. Une coordonnée de $\mathbf{E}$ est donc un vecteur d'entier. Nous avons une plus grande souplesse pour la classe de pixels (les valeurs de $\mathbf{F}$). Dans le cas des images multispectrales (images couleur, ou même des images à spectre plus important), la classe de pixels est déterminée à la fois par sa dimension et par les types sur chacune des dimensions; la représentation n'est pas unique et sera supposée libre. Il faut donc considérer le pixel comme étant un *méta-type*, paramètre de la structure d'image.

L'image est alors une structure générique *template* de deux paramètres : la dimension de son espace de coordonnées d et le type « T » de ses pixels. La plupart des méthodes de la classe image dépendent ensuite de ces deux paramètres, ce qui en fait une structure très souple de « conteneur » de pixels. Les

[(xii)] nous verrons par la suite comment est traitée la couleur au niveau informatique, et qu'elle est réduite à un espace à 3 dimensions.

images exposeront les propriétés liées à la géométrie du *support* de l'image, et des méthodes d'évaluation de l'image des points de ce support (c'est à dire des méthodes permettant d'évaluer $\mathcal{I}(x), x \in \mathbf{E}$.

En résumé, les méthodes et propriétés de la structure d'image concernent sa taille, l'allocation des ressources en mémoire, l'organisation des données, et l'accès à ces données. Concernant l'organisation de la mémoire image, nous avons simplement utilisé un tableau mono-dimensionnel dont la taille est le produit de la taille sur chacune des dimensions: cette représentation a pour avantage une grande souplesse pour n'importe quelle dimension. La méthode d'accès aux points est celle concernant l'application image elle-même, à savoir celle associant un point de $\mathbf{E}$ à son image par $\mathcal{I}$.

Puisque l'organisation de la mémoire image est mono-dimensionnelle, une coordonnée de $\mathbf{E}$ et son déplacement depuis l'origine de ce tableau sont des notions équivalentes. Nous appellerons ce déplacement l'*offset* du point, ce sera une manière compacte de coder une coordonnée. Par la suite, nous utiliserons l'une ou l'autre de ces représentations. Cela ne remet pas en cause la généricité de l'espace des coordonnées, à partir du moment où l'ensemble est homogène. Lorsque certains algorithmes devront manipuler des images de dimensions différentes, l'offset ne sera plus une représentation valable de $\mathbf{E}$ pour toutes les images manipulées.

Remarque Notons que les fonctionnalités exposées ici ne restreignent en rien les applications manipulant cette structure. La méthode d'accès à l'application image sera toujours une méthode unaire - ie. à un seul paramètre - du type:

```
1  template <class T, int d = 3>
2          class Image:
3  {
4          /*  ...  */
5
6          //  accès par une représentation "coordonnée"
7          T& pixel(const coord<d> &p);
8          //  accès par une représentation "offset"
9          T& pixel(const offset p);
10
11 };
```

où « coord<d> » est un point de $\mathbb{N}^d$.

La structure d'image seule ne présente pas de grand intérêt. C'est en effet une structure souple de « conteneur » de pixels, mais elle ne propose pas de méthode d'accès pratique en tant que telle. C'est ce que nous allons voir à présent.

2.2.2. Les sous-ensembles

On s'intéresse souvent non à $\mathbf{E}$ dans sa totalité, mais à un sous-ensemble $\mathbf{E}' \subset \mathbf{E}$, lorsqu'il s'agit par exemple d'effectuer une opération de voisinage, ou sur un domaine restreint de $\mathbf{E}$. Il y a donc deux concepts différents: la construction de $\mathbf{E}'$ à partir d'une fonction sur $\mathbf{E}$ et éventuellement d'autres ensembles, et la collecte effective de $\mathcal{I}(\mathbf{E}')$, image de $\mathbf{E}'$ par $\mathcal{I}$.

Cette définition générale de création de sous-ensemble ne permet pas une exploitation informatique pratique. La programmation ces dernières années s'est particulièrement orientée vers une méthode de création, ou collecte, d'ensemble et de sous-ensembles, appelée *itérateurs*. Les itérateurs peuvent être vus comme des fonctions sur des ensembles vers une suite de parties de cet ensemble, telle que la donnée de la suite permet de définir entièrement ou en partie l'ensemble. Notons It l'application *itérateur*:

$$It(\mathbf{E}) : \begin{cases} \mathcal{P}(\mathbf{E}) & \rightarrow & \mathcal{P}(\mathbf{E})^{\mathbb{N}} \\ \mathbf{X} & \mapsto & (u_n^{\mathbf{X}})_n / \bigcup_{n \in \mathbb{N}^*} u_n^{\mathbf{X}} \subseteq \mathbf{X} \end{cases}$$

En général, on ne s'intéresse qu'aux points de $\mathbf{X}$ et non à des sous-ensembles, It produit une suite de points décrivant tout ou partie de $\mathbf{X}$ ($It(\mathbf{E})$ à valeur dans $\mathbf{E}^{\mathbb{N}}$.) Dans le cas particulier des images, les informations d'intérêt sont donc les pixels. Mais plus précisément, l'intérêt est porté à la fois sur les coordonnées des points parcourus, et sur l'image de ces points par $\mathcal{I}$. Un itérateur sur des structures d'image proposera donc ces deux informations. Cependant, si nous avons la coordonnée,

il est toujours possible d'évaluer l'image de cette coordonnée à travers l'application image; pour cela nous ne considérerons que la génération de coordonnées. Les ensembles traités étant dénombrables et finis, donc de cardinalité entière, l'itération sur la suite $(u_n^{\mathbf{X}})_n$ pour un $\mathbf{X}$ fixé a donc un *début* et une *fin*.

L'attrait particulier des itérateurs est que leur action, ou indirectement la description de l'ensemble $\mathbf{X}$ sous-jacent, ne nécessite la définition que de quatre fonctions uniques, et ce quel que soit l'ensemble $\mathbf{X}$. Ces quatre fonctions sont:

1. une fonction de début, fournissant le premier élément de la suite: $u_1^{\mathbf{X}}$. Nous ferons référence à cette fonction par « begin ».
2. une fonction de fin, fournissant le dernier élément de la suite: $u_N^{\mathbf{X}}$ (fonction « end »).
3. une fonction d'*itération*, effectuant le passage de u_n^X à $u_{n+1}^{\mathbf{X}}$. Nous référencerons cette fonction par « $++$ », qui fera avancer l'itérateur sur l'occurrence suivante.
4. une fonction de comparaison, permettant de savoir si on a atteint la fin de la récurrence. La fin est donc connue par comparaison entre l'élément courant u_n^X et le dernier élément $u_N^{\mathbf{X}}$.

La comparaison n'est bien sûr par liée à la valeur $u_n^{\mathbf{X}}$ mais à l'indice n, puisqu'il est tout à fait envisageable de parcourir plusieurs fois le même point, la suite $(u_n^{\mathbf{X}})$ n'étant pas une injection sur $\mathbf{X}$.

Remarque On pourrait s'interroger sur le fait que fournir N, c'est à dire la cardinalité de $\mathbf{X}$, permettrait de s'astreindre de la comparaison qui semble complexe. Or, la détermination de la cardinalité imposerait - dans la plupart des structures rencontrées - un comptage des éléments de $\mathbf{X}$, c'est à dire un parcours total de $\mathbf{X}$. Cette action, bien que simple, est lourde et redondante par rapport à l'usage que l'on souhaite précisément de l'itérateur (le parcours de X pour effectuer un traitement). C'est pourquoi on demande aux itérateurs de ne connaître que leur position de début et de fin, la fonction de récurrence étant liée à l'organisation des données.

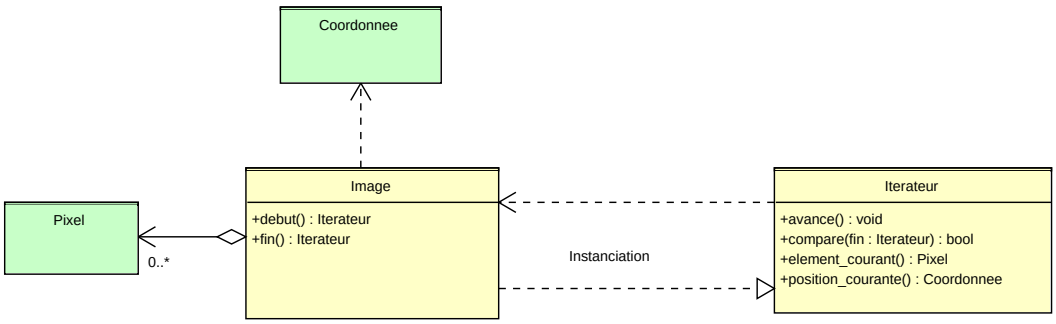


Fig. 2.1.: Itérateur de la classe *Image*. L'image est considérée comme étant un agrégat de pixels (ligne présentant une flèche et un diamant). Elle est également dépendante de la structure de coordonnée (ligne fléchée pointillée). Elle peut créer un itérateur d'image (ligne avec une flèche pleine), implémentant les fonctions de comparaison et d'itération, ainsi que des fonctions d'accès aux coordonnées ou au pixel pointé. L'itérateur est dépendant de l'image, et cache cette dernière à ses clients.

2.2.2.1. Sous-ensemble d'image

Les itérateurs appliqués directement aux images permettent déjà la construction d'applications complexes. Nous avons divisé les itérateurs d'image en trois classes principales:

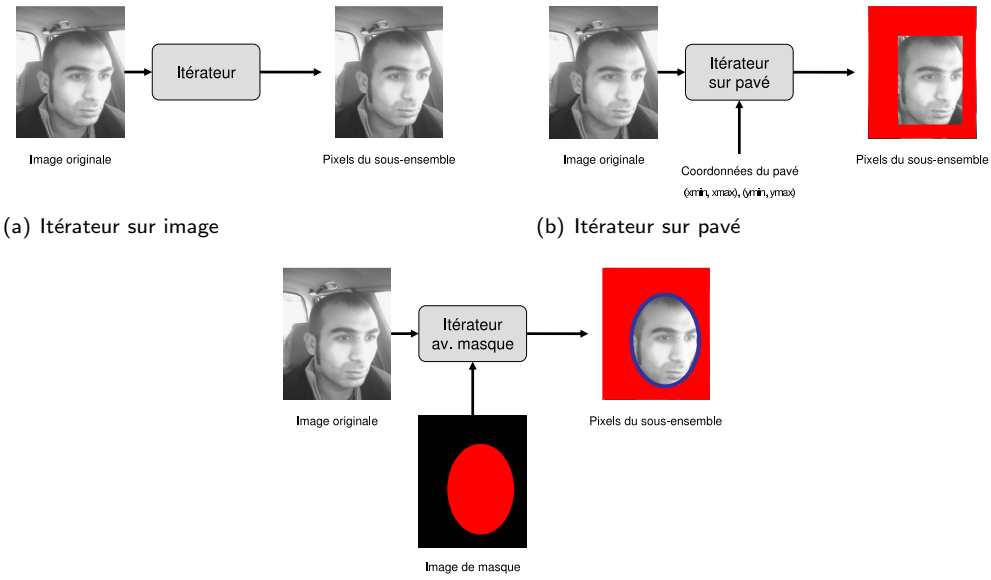
1. Image : Le cas immédiat est lorsque l'ensemble d'intérêt est l'image $\mathcal{I}$ dans sa totalité. La suite créée par l'itérateur est alors le support $\mathbf{E}$ de l'image.

2. Pavés : Un cas un peu plus spécifique est lorsque l'on souhaite parcourir un pavé à l'intérieur de l'image. Les pavés sont des rectangles pour les images en deux dimensions, des parallélépipèdes rectangles pour celle en trois dimensions... L'intérêt des pavés provient de la simplicité de leur représentation : seules les coordonnées de début et de fin selon chaque dimension sont nécessaires. Un itérateur peut se servir de cette information pour ne parcourir que les points à l'intérieur du pavé d'intérêt.

3. Région d'intérêt : bien sûr il est possible de s'intéresser à des régions d'intérêt autre que celles ressemblant à des pavés. Supposons une deuxième image $\mathcal{I}'$ de même dimension que $\mathcal{I}$. Il est possible d'appliquer un itérateur sur $\mathcal{I}$ mais en se restreignant aux points de $\mathcal{I}'$ vérifiant un prédicat. La fonction de récurrence de ce type d'itérateur passe alors les points ne vérifiant pas ce prédicat, et cela de manière totalement transparente pour la fonction qui utilise l'itérateur.

Le principal intérêt de cette approche vient de l'absence de dépendance informatique entre la notion d'itérateur et la classe contenant les points (la classe conteneur). La dépendance est bien entendu réelle, mais les opérateurs utilisant les itérateurs n'ont pas besoin de connaître le conteneur, ni même la nature de l'itérateur (sur l'image, sur un pavé ou sur une région d'intérêt). Le concept d'itérateur donne un moyen *universel* pour créer des ensembles de points dans l'image. La liberté d'utilisation est très grande, et il est tout à fait envisageable d'utiliser dans un opérateur unique des itérateurs parcourant les supports des images de différentes manières. Hormis quelques opérateurs spécifiques (une transposition selon un axe, une décimation, etc.) cette fonctionnalité présente un intérêt limité. Nous supposerons donc que dans la majeure partie des cas, les itérateurs issus d'une même structure (une image de dimension précise) « itèrent » de la même manière.

Quelques illustrations sont données sur les figures 2.2 et 2.3.



(a) Itérateur sur image

(b) Itérateur sur pavé

(c) Itérateur sur région

Fig. 2.2.: Trois exemple d'itérateur sur la classe image. (a): l'itérateur sur image ne se sert que de la structure de l'image. (b): les coordonnées du pavé interviennent pour restreindre le support de l'itérateur. (c): une image $\mathcal{I}'$ sert de fonction indicatrice pour déterminer l'ensemble des points d'intérêt lors de l'itération.

Remarque On est souvent intéressé par un ordre de parcours particulier sur $\mathbf{E}$, par exemple le « parcours vidéo » pixels ordonnées de la même manière que le parcours du faisceau d'un téléviseur - bien

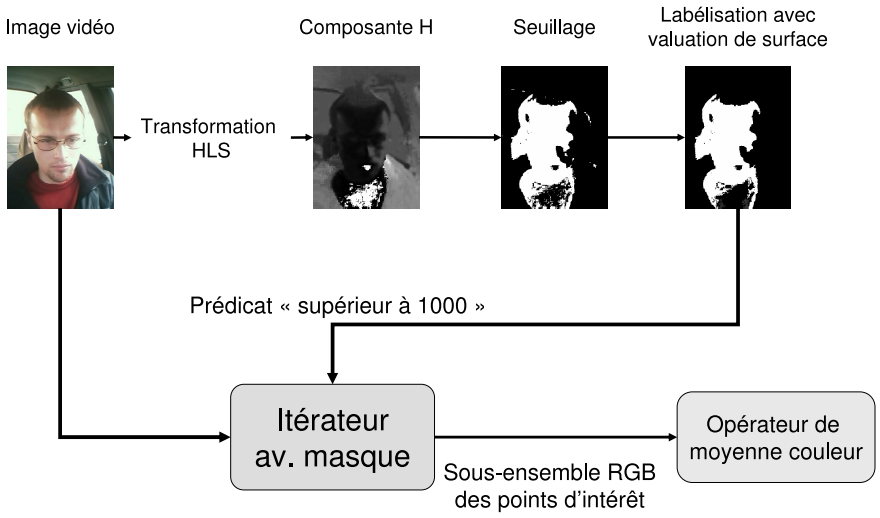


Fig. 2.3.: Application simple utilisant les itérateurs sur régions avec prédicat: nous souhaitons effectuer une moyenne sur le visage, dans un contexte défini. Un seuillage de la teinte H est effectué, ensuite une valuation des composantes connexes par leur surface (nombre de pixels) permet ensuite à l'itérateur, muni d'un prédicat binaire, de ne parcourir que les points d'intérêt. L'interface de l'itérateur étant universelle, l'opérateur de moyenne vectorielle ne se charge pas de la construction de l'ensemble et n'effectue ainsi que l'opération pour laquelle il a été affecté.

connu en traitement d'image. Cependant nous considérerons que dès lors que ce genre de contrainte apparaît, il est nécessaire de s'interroger sur la structure interne de stockage de l'image $\mathcal{I}$. Or ce genre de préoccupation est à l'encontre de la généricité que nous nous sommes imposés. De plus nous souhaiterions qu'en première approximation, les algorithmes ne soient pas dépendant de cet ordre de parcours. Bien sûr cela freine les performances en terme de temps de calcul, car des hypothèses *a priori* fortes sur l'organisation de la structure d'image permettraient une écriture adaptée des algorithmes. Cela fait partie intégrante de l'**optimisation**, qui est une autre discipline.

2.2.2.2. Voisinages

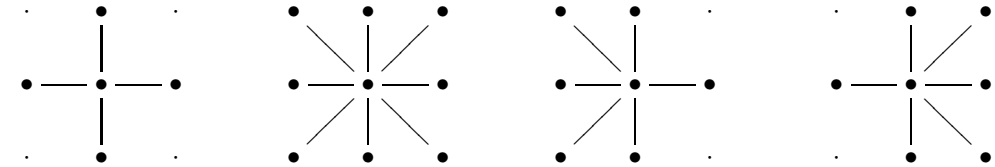
Nous avons vu quelques types d'itérateur particuliers appliqués directement aux images. Les quelques structures utilisées en Morphologie Mathématique que nous allons voir à présent tirent profit du cadre des itérateurs de manière analogue. Un itérateur n'étant rien d'autre qu'un moyen informatique permettant de décrire un sous-ensemble, nous allons construire ces sous-ensembles de manière à induire sur l'espace des points $\mathbf{E}$ une *topologie de voisinage*.

Un voisinage $\mathcal{N}$ est une application de $\mathbf{E}$ vers l'ensemble des parties de $\mathbf{E}$:

$$\mathcal{N} : \left\{ \begin{array}{ccc} \mathbf{E} & \rightarrow & \mathcal{P}(\mathbf{E}) \\ p & \mapsto & \{v, v \in \mathbf{E}\} \end{array} \right.$$

Un voisinage est simplement une application construisant un sous-ensemble de $\mathbf{E}$ et fonction d'un point $p \in \mathbf{E}$. Nous appellerons le point p le *centre* du voisinage $\mathcal{N}$ et noterons $\mathcal{N}_p = \mathcal{N}(p)$ ce voisinage.

À l'instar des remarques précédentes, l'ordre des points du sous-ensemble construit n'est pas une propriété sur laquelle nous porterons notre attention. La construction du voisinage peut être de complexité variable, et dépend de la nature du voisinage. Voyons à présent les types de voisinage qu'il est possible de rencontrer.



(a) 4-connexe (b) 8-connexe (c) 6-connexe (1) (d) 6-connexe (2)

Fig. 2.4.: Quelques graphes de connexité très utilisés. • signifie la présence d'un point. (c) et (d) sont utilisés pour simuler la trame hexagonale: l'un est appliqué sur les lignes paires, l'autre sur les lignes impaires. Par ailleurs, l'un est le transposé de l'autre.

Graphe de voisinage Un *graphe de voisinage* $\mathcal{N}^g$ est la manière la plus simple de construire les voisinages. On utilise souvent le degré de connexité k pour décrire le graphe: k est une borne supérieure de la cardinalité du voisinage. On parle alors de k-connexité (4-connexité, 6-connexité, etc.). Le nombre de voisins est en général constant, mais peut être réduit si la géodésie de l'image sur laquelle est appliqué le voisinage l'impose.

La simplicité du graphe de voisinage provient du fait que les voisins sont déductibles par une fonction *simple* de la position p de son centre. Passons à la représentation suivante: soit p un point et $\mathcal{N}_p^g$ ses voisins; nous notons $\Delta v(p) = \left\{p - v, v \in \mathcal{N}_p^g\right\}$ les déplacements des voisins par rapport au centre pour le point p. Deux cas se présentent:

- La simplicité maximale est obtenue lorsque le voisinage est *invariant par translation*: les déplacements Δv sont alors constants pour tout $p \in \mathbf{E}$. Dans ce cas précis, chaque voisinage se déduit donc par translation par rapport à son centre. Lorsque l'image et l'ensemble des coordonnées relatives [(xiii)] des voisins sont disponibles, il est même possible de calculer les différences en terme d'offset, et la translation revient à effectuer k sommes sur l'espace mono-dimensionnel des offsets.

- La propriété d'invariance par translation est parfois trop forte. Il est également possible d'avoir i configurations. L'indépendance entre la position du centre et les déplacements n'est plus totale: une fonction g du centre à valeurs dans $\mathbb{N}_i^*$ les relie. De la même manière que précédemment, une fois l'image et les configurations de voisinages connues, il est possible de pré-calculer les déplacements Δv sous forme de différence d'offset par rapport au centre, mais cette fois pour toutes les configurations. Une fois le voisinage centré, un appel à la fonction g permet de choisir la configuration qui est ensuite translatée de la même manière que pour le cas précédent.

Le premier cas mentionné ci-dessus est celui utilisé par exemple par les graphes de connexités 4 et 8 (cf. figure 2.4). Le deuxième cas est plus pragmatique et sert par exemple à simuler une trame hexagonale sur une image. En effet, la trame hexagonale n'est pas possible directement sur une image dont la grille est supposée régulière et « carrée ». La trame hexagonale est approchée par une alternance de deux configurations (cf. figures 2.4(c) et 2.4(d)).

Ces deux cas sont bien-entendu équivalents en pratique; l'intérêt du graphe de voisinage réside dans la manière de déduire les points du voisinage à partir d'un point *centre* et d'un vecteur de déplacements. Que ce vecteur soit constant ou non, la construction des points du voisinage est la même - translation et somme des déplacements - à un appel de fonction près.

Généralisation du concept de voisinage L'exemple du graphe de voisinage introduit la généralisation de cette notion. Nous avons vu que le graphe de voisinage se composait d'un certain nombre de vecteur de déplacement. Ces vecteurs de déplacement indiquent la manière de créer le voisinage en chaque point et sont une représentation universelle du type de voisinage « graphe de connexité ». Nous avons également vu qu'une fois l'image et les vecteurs de déplacement connus, certains pré-calculs (les

[(xiii)] par rapport au centre

vecteurs d'offsets) étaient possibles, ainsi qu'une représentation suffisante du voisinage.

Ce qu'il est important de remarquer ici est que la manière de construire le voisinage appartient à l'application $\mathcal{N}$, et non aux algorithmes comme cela peut se faire couramment. De ce fait et dans l'optique de la *POO*, le voisinage devient un objet proposant les moyens suivants:

1. *initialisation* : simple ou complexe, elle permet d'effectuer certains pré-calculs liés à la géométrie de l'image
2. *centrage* : il s'agit de l'étape pendant laquelle la valuation de $\mathcal{N}_p$ est effectuée. La classe de voisinage construit de manière interne le sous-ensemble des points voisins de p
3. *parcours* : puisque l'image du centre p par le voisinage $\mathcal{N}$ est un sous-ensemble, la méthode d'accès par les itérateurs est totalement adaptée. Ces derniers parcourent le sous-ensemble de points voisins créés pendant l'étape de centrage

L'itérateur du voisinage se comporte de la même manière que celui issu des images, et propose la même interface à ses clients. Il informe à la fois sur les coordonnées des voisins et leur valuation par l'application image. Nous reviendrons sur ce point extrêmement important lorsque nous discuterons des opérateurs. Il est difficile de montrer l'étendue des possibilités offertes par cette approche générale sans parler des opérateurs. Il est toutefois possible d'exposer quelques propriétés inhérentes à l'approche des voisinages, ainsi que des exemples montrant la généricité de l'approche.

Voisinage et bords puisque la construction des points voisins est cachée dans le centrage du voisinage, le voisinage gère de manière également cachée les effets de bords. Cette tâche n'incombe plus à l'algorithme. Plusieurs solutions sont envisagées:

1. *géodésie totale* : chaque centrage du voisinage effectue l'intersection du voisinage avec le support de l'image. La liste des points voisins construite est celle contenue à l'intérieur de ce support
2. *constante de bord :* certaines approches ajoutaient de manière artificielle un bord d'image, ce qui permettait de s'astreindre de la gestion des bords dans l'algorithme. Il est possible de simuler cette approche lors de la construction des points voisins, par l'utilisation d'une structure légèrement plus complexe que le vecteur de points. La liste des voisins comporte alors à la fois la coordonnée du voisin, mais également un indicateur pour savoir si le point est dans l'image ou non. Cet indicateur est mis à jour dans le centrage du voisinage, et utilisé lors de l'accès à l'image du point par le biais de l'itérateur. Toutefois nous utilisons très peu cette méthode, car si la valuation de l'image en dehors de son support est possible, il est difficile de générer une coordonnée valide en dehors du support de l'image.
3. *fonction géométrique complexe :* il est enfin possible d'inclure dans le processus de centrage certaines transformations géométriques de la structure initiale en forme de pavé de l'image. Cela peut être par exemple une symétrie axiale du point sortant par rapport au bord, un bouclage sur le bord opposé, etc.. Cette approche permet entre autre de générer un espace plus important que l'espace initial, par exemple pour simuler une périodisation de l'image (transformation de Fourier).

Fonctions structurantes Les fonctions structurantes trouvent plusieurs définitions. Il s'agit soit d'un élément structurant non plan, à valeurs dans $\mathbf{F}$, et qui peut être confondu avec une image, soit d'une fonction créant un voisinage dont le contenu (ie. les points du voisinage) est une fonction complexe, de l'image ou d'une image tierce. Voici deux exemples récents de fonction structurante.

Laveau [LB05, Lav05] s'est servi de cette approche pour créer des voisinages suivants le mouvement sur des séquences vidéos. Une image spatio-temporelle de champ de mouvement, initialement calculée à partir de la séquence, donne des vecteurs de déplacement en chaque point de l'espace spatio-temporel. Ce champ de mouvement est utilisé de manière transparente dans l'objet de voisinage. Un graphe de voisinage fixe donne un premier choix des voisins similairement aux graphes de voisinage « classiques ». Le processus de centrage déforme ce graphe de voisinage de manière à ce que celui-ci épouse le champ

de mouvement. Cela permet entre autres d'effectuer des filtrages morphologiques et des segmentations cohérents par rapport au mouvement de la séquence vidéo analysée. Ce que voient ensuite les opérateurs est le même cycle « centrage/itération sur les voisins » que pour le cas standard. Les opérateurs de voisinage compatibles sont ensuite immédiatement disponibles sur cette classe de fonction structurante.

Lerallut [LDM05, Ler06] utilise de manière analogue cette approche pour les *amibes morphologiques*. Les amibes sont des fonctions structurantes dépendantes d'un temps géodésique par rapport au centre, et sur une image de contrôle autre que l'image sur laquelle elles sont appliquées. Elles se déforment de manière à épouser certaines caractéristiques de l'image de contrôle, comme par exemple des lignes de gradients forts. Encore une fois, l'avantage de l'approche réside dans le fait qu'une fois l'amibe implémentée, aucune modification des opérateurs de voisinage n'est nécessaire, puisque les méthodes d'accès au voisinage sont les mêmes. Toute les fonctionnalités de l'amibe sont cachées dans sa méthode de centrage.

Remarque conclusive Nous avons volontairement mis les notions de voisinage dans la section plus générale de sous-ensembles. Cette démarche vient directement de la définition de voisinage comme fonctions génératrices de sous-ensembles, définition sur laquelle le cadre des itérateurs s'étend naturellement.

Ce qui nous intéresse ici est de montrer que la manière de parcourir les voisinages peut être perçue de manière similaire par les clients des itérateurs, quel que soit l'élément structurant choisi. Ceci permettra ensuite d'utiliser les mêmes opérateurs sur les itérateurs créés sur des sous-ensembles, à partir d'une base d'élément structurant, d'une base géométrique, ou encore à partir de fonctions tierces.

2.2.2.3. Récapitulatif

Dans cette partie nous avons vu ce qu'était un *itérateur* et dans quelle mesure il s'agissait d'une notion parfaitement adaptée à la construction de sous-ensembles, dans le cadre du traitement d'image et de la Morphologie Mathématique. Nous avons également vu que l'idée d'itération se déclinait sous un certain nombre de formes, et ce sans modifier la structure informatique des clients. Certains itérateurs sont triviaux, comme ceux décrivant toute l'image: ils seront essentiellement utilisés pour pouvoir extraire des informations de position, par exemple pour le parcours complet d'une image dans un algorithme, ou pour centrer un voisinage. Les itérateurs restreints à un support géométrique, support décrit par exemple par un rectangle ou par une fonction binaire tierce, représentent un moyen certes simple mais puissant pour effectuer des opérations, par exemple de mesure. Le même concept peut être utilisé pour une implémentation générique de la structure de voisinage utilisée en MM, pour un coût légèrement supplémentaire. Il faut en effet pouvoir informer le centre du voisinage; une fois ce centre informé, le voisinage - terme pris au sens large - peut être construit de manière fixe ou variable, selon un graphe de connexité constant, ou changeant sa structure selon la position géométrique, selon une fonction tierce, etc.. .

2.2.3. Les graphes

La structure de graphe est une structure mathématique très générale et très utilisée dans beaucoup de domaines, en traitement d'image, en réseau de télécommunication, etc.. Son intérêt majeur, tout du moins son intérêt informatique, réside dans le fait qu'elle permet une représentation très compacte, en terme de nombre d'éléments et d'occupation mémoire, des données d'études.

Bien que nous ne présenterons pas par la suite de développements liés aux graphes, cette structure a suscité un intérêt particulier dans notre librairie. En effet, les graphes permettent de changer de représentation et de manipuler des éléments sémantiquement plus riches que les points d'une image. Ainsi il n'est pas rare qu'une segmentation par ligne de partage des eaux soit suivie d'une construction de graphe d'adjacence entre les régions. L'espace de travail est alors modifié et les entités ne sont plus les mêmes. L'objectif est de montrer qu'à partir de la définition formelle de graphe, il est tout à fait envisageable d'appliquer les mêmes principes de méta-programmation exposés précédemment pour en faire une structure informatique très souple.

Définition 2.1 (Paire)
Soit S un ensemble, on note $S^{(2)}$ l'ensemble des paires d'éléments de S: $S^{(2)} = \{(p,q), p \in S, q \in S\}$.

$S^{(2)}$ est l'ensemble des couples d'éléments de S mais dans lequel on identifie chaque couple (s_1, s_2) avec son « opposé » (s_2, s_1).

Définition 2.2 (Graphe)
*Un **graphe** G est la donnée de deux ensembles S_G et A_G et d'une fonction*

$$\delta_G : A_G \mapsto S_G^2$$

*Les éléments de S_G sont appelés les **sommets** du graphe G, les éléments de A_G sont les **arêtes** (ou flèches) du graphe.*

Définition 2.3 (Graphe non orienté)
Un graphe non-orienté est la donnée de deux ensembles, S_G et A_G et d'une fonction

$$\delta_G : A_G \mapsto S_G^{(2)}$$

La terminologie de S_G et A_G est la même que pour le cas orienté.

Méta-programmation Les ensembles sommets et arêtes S_G et A_G sont, selon la définition, quelconques. La représentation de ces ensembles peut être aisément méta-programmée. L'application δ est par contre spécifique à la structure de graphe. Il existe plusieurs représentations très différentes, et le choix est très souvent guidé par des considérations algorithmiques et l'orientation ou non des arêtes. Parmi ces représentations, nous pouvons nommer la matrice d'adjacence qui offre une structure très simple et rapide d'accès, mais peu compacte en mémoire (la matrice code toutes les combinaisons possibles de nœuds).

De la même manière que pour les images, des itérateurs permettent ensuite de parcourir l'ensemble des nœuds ou des arêtes. Ensuite, à partir de l'une ou l'autre de ces entités nœud ou arête, il est possible de remonter simplement par l'application « graphe » aux entités adjacentes (respectivement arêtes et nœuds).

2.2.4. Les relations d'ordre et les treillis

La morphologie mathématique dispose d'un cadre algébrique établi sur la structure de treillis. Cette structure algébrique fait naturellement intervenir la notion d'ordre. Rappelons dans un premier temps les définitions concernant les ordres et les équivalences. Cela nous permettra par la suite de définir un certain nombre d'opérateurs par ces outils. Nous rappellerons ensuite la notion de treillis et les liens existant avec les relations d'ordre.

Définition 2.4 (Relation binaire)
Soit $\mathbf{E}_1$ et $\mathbf{E}_2$ deux ensembles. Le sous-ensemble $\mathcal{R} \subset \mathbf{E}_1 \times \mathbf{E}_2$ est une relation binaire sur $\mathbf{E}_1 \times \mathbf{E}_2$.

Une relation binaire peut être interprétée comme une correspondance entre les éléments de $\mathbf{E}_1$ et $\mathbf{E}_2$. Dans la description d'une relation, il est surtout important de savoir si une paire fait partie de la relation, les éléments de la paire sont alors reliés par la relation (nommée parfois « application flèche » car reliant x_1 à x_2). La notation $(x_1, x_2) \in \mathcal{R}$ fait alors place à $x_1 \mathcal{R} x_2$.

Définition 2.5 (Relation d'ordre partiel)
Une relation binaire $\preceq_{\mathcal{R}} \subset \mathbf{F} \times \mathbf{F}$ est une relation d'ordre partiel si et seulement si elle est réflexive, anti-symétrique et transitive. $\forall (x,y,z) \in \mathbf{F}^3$:

$$\begin{array}{lcll} x \preceq_{\mathcal{R}} x & & & \textit{réflexivité} \\ (x \preceq_{\mathcal{R}} y) \wedge (y \preceq_{\mathcal{R}} x) & \Rightarrow & x = y & \textit{anti-symétrie} \\ (x \preceq_{\mathcal{R}} y) \wedge (y \preceq_{\mathcal{R}} z) & \Rightarrow & x \preceq_{\mathcal{R}} z & \textit{transitivité} \end{array}$$

L'ensemble $\mathbf{F}$ muni d'un tel ordre est dit « partiellement ordonné », car un couple d'éléments dans $\mathbf{F}^2$ n'appartient pas nécessairement à $\preceq_{\mathcal{R}}$, ou en d'autre termes, n'est pas nécessairement comparable par la relation $\preceq_{\mathcal{R}}$.

Définition 2.6 (Relation d'ordre partiel stricte)
Une relation binaire $\prec_{\mathcal{R}} \subset \mathbf{F} \times \mathbf{F}$ est une relation d'ordre partiel stricte s'il existe une relation d'ordre partiel $\preceq_{\mathcal{R}} \subset \mathbf{F} \times \mathbf{F}$ tel que:

$$\begin{cases} \prec_{\mathcal{R}} \subset \preceq_{\mathcal{R}} \\ \forall (x,y) \in \preceq_{\mathcal{R}},\ (x \prec_{\mathcal{R}} y) \Leftrightarrow (x \preceq_{\mathcal{R}} y)\ \wedge\ \neg (x = y) \end{cases}$$

Définition 2.7 (Relation d'ordre total)
Un ordre partiel $\preceq_{\mathcal{R}}$ sur un ensemble $\mathbf{F}$ est dit total si l'ordre peut s'appliquer à tout couple d'éléments de $\mathbf{F}$, dans un sens ou dans l'autre:

$$\forall (x,y) \in \mathbf{F}^2,\ (x \preceq_{\mathcal{R}} y)\ \vee\ (y \preceq_{\mathcal{R}} x)$$

Un ensemble totalement ordonné est alors appelé une « chaîne ».

Définition 2.8 (Relation d'équivalence, Classe d'équivalence)
Une relation binaire $\cong_{\mathcal{R}} \subset \mathbf{F} \times \mathbf{F}$ est une relation d'équivalence si et seulement si elle est réflexive, symétrique et transitive.
Soit $x \in \mathbf{F}$, l'ensemble $\mathcal{C}_x = \{y \in \mathbf{F} / y \cong_{\mathcal{R}} x\}$ est appelé classe d'équivalence de x.

Le réflexivité de $\cong_{\mathcal{R}}$ entraîne $\mathcal{C}_x \neq \varnothing$, et la transitivité fait que chaque élément de $\mathbf{F}$ appartient à une et une seule classe d'équivalence. Les classes d'équivalence sont donc disjointes deux à deux et forment une partition de $\mathbf{F}$.

Définition 2.9 (Ensemble quotient)
L'ensemble $\mathbf{F}_{\setminus \cong_{\mathcal{R}}} = \{\mathcal{C}_x, x \in \mathbf{F}\}$ est appelé ensemble quotient de $\mathbf{F}$ par rapport à la relation $\cong_{\mathcal{R}}$.

Nous nous servirons de la notion de d'ensemble quotient lorsqu'il s'agira de définir plus précisément la « labellisation ».

Définition 2.10 (Relation d'ordre lexicographique)
Soit $\mathbf{F}$ un ensemble séparable tel que $\mathbf{F} = \mathbf{F}_1 \times \mathbf{F}_2 \ldots \times \mathbf{F}_p$, chaque espace muni des ordres totaux $\prec_{\mathbf{F}_1}$, $\prec_{\mathbf{F}_2}$, ... $\prec_{\mathbf{F}_p}$, et $\sigma : \mathbb{N}_p^* \rightarrow \mathbb{N}_p^*$ une permutation.

Un ordre lexicographique $\prec_{\mathbf{F},\sigma}$ sur $\mathbf{F}$ est un ordre défini de la manière récurrente suivante:
$\forall (x,y) \in \mathbf{F}^2, x = (x_1, x_2, \ldots, x_n),\ y = (y_1, y_2, \ldots, y_n)$*:*

$$(x \prec_{\mathbf{F},\sigma} y) \Leftrightarrow \exists k \in \mathbb{N}_p^* \,/ \begin{cases} \neg \left(x_{\sigma(l)} \prec_{\mathbf{F}_{\sigma(l)}} y_{\sigma(l)}\right) \wedge \neg \left(y_{\sigma(l)} \prec_{\mathbf{F}_{\sigma(l)}} x_{\sigma(l)}\right) & ,\forall l \in \mathbb{N}_{k-1}^* \\ x_{\sigma(k)} \prec_{\mathbf{F}_{\sigma(k)}} y_{\sigma(k)} \end{cases}$$

L'ordre lexicographique est un outil très simple permettant d'étendre et d'appliquer, sur des espaces à plusieurs dimensions, les outils de comparaison courants. L'ordre lexicographique reste bien évidement une relation binaire.

La permutation σ permet de créer implicitement un ordre de pertinence concernant les comparaisons sur les espaces F_k. Ainsi, $F_{\sigma(1)}$ est l'espace sur lequel l'ordre sera testé en premier, $F_{\sigma(2)}$ le second espace, etc.. $F_{\sigma(1)}$ apparaît donc plus important que $F_{\sigma(2)}$. S'il y a égalité sur l'espace $F_{\sigma(k)}, k \in \mathbb{N}_{p-1}^*$, c'est à dire si la comparaison n'apporte aucune information, alors elle se poursuit sur l'espace suivant $F_{\sigma(k+1)}$. Ainsi, $F_{\sigma(k+1)}$ apparaît comme un espace qui en quelque sorte, améliore ou enrichit la précision de la comparaison effectuée sur $F_{\sigma(k)}$ et par récurrence sur les $F_{\sigma(i)}, i \in \mathbb{N}_k^*$. Nous nous servirons beaucoup de cette explication pour introduire des nouveaux algorithmes de morphologie mathématique

basés sur des ordres lexicographiques. Retenons particulièrement l'ordre d'importance implicite des espaces selon la permutation σ.

Un ensemble muni d'une relation d'ordre est une structure algébrique particulière présentant des propriétés riches. Après les définitions des relations d'ordre, il semble naturel de définir les treillis, mais la définition simple ne permettra pas une exploitation informatique particulière. Par contre, étant donné la structure des opérations de relation d'ordre et d'équivalence, une exploitation informatique sera possible sous certaines conditions.

Définition 2.11 (Borne supérieure, borne inférieure)
Soit X un ensemble ordonné et $x \in X$. On dit que $m \in X$ est la borne supérieure de X si:
- *m majore x*
  $(m \geq x)$
- *m est le plus petit majorant de X:*
  $\forall u \in X, (u \geq x) \Rightarrow u \geq m$

La même définition s'applique pour la borne inférieure en remplaçant l'ordre $\geq$ par $\leq$.

Définition 2.12 (Treillis - définition algébrique)
Un treillis $\mathcal{L}$ est un ensemble dans lequel toute paire d'éléments admet une borne inférieure et une borne supérieure dans $\mathcal{L}$. Plus précisément, un treillis $\mathcal{L}$ est un ensemble non vide muni de deux lois internes, notées $\veebar$ et $\barwedge$, et vérifiant les trois propriétés suivantes, $\forall (a,b) \in \mathcal{L}^2$:
- *Commutativité et associativité*
- *Idempotence :*
  $$\begin{cases} a \veebar a = a \\ a \barwedge a = a \end{cases}$$
- *Absorption :*
  $$\begin{cases} a \barwedge (a \veebar b) = a \\ a \veebar (a \barwedge b) = a \end{cases}$$

Le treillis $\mathcal{L}$ est souvent noté $\langle \mathcal{L}, \veebar, \barwedge \rangle$ en notation algébrique.

Proposition 2.1
Soit $\langle \mathcal{L}, \veebar, \barwedge \rangle$ un treillis. Les lois internes $\veebar$ et $\barwedge$ induisent une relation d'ordre $\preceq$ sur $\mathcal{L}$, ce entraînant l'équivalence des deux représentations : $\langle \mathcal{L}, \veebar, \barwedge \rangle = \langle \mathcal{L}, \preceq \rangle$

L'équivalence des deux représentations, ainsi que la transitivité de la relation d'ordre permet de définir les treillis sous forme de diagramme de Hasse. Un exemple célèbre de treillis est donné en figure 2.5 sous cette forme.

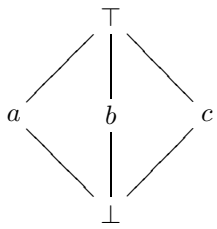


Fig. 2.5.: Le treillis $\mathcal{M}_3$ représenté sous forme de diagramme de Hasse. $\top$ et $\bot$ sont respectivement le supremum et l'infimum du treillis.

Définition 2.13 (Treillis complet)
Un treillis dont tous les sous-ensembles non vides admettent un majorant est dit ***complet****.*

La complétion d'un treillis permet d'écrire les opérateurs $\overline{\wedge}$ et $\overline{\vee}$ sur des sous-ensembles du treillis : cette propriété assure la stabilité de la structure algébrique, alors qu'un treillis sans hypothèse de complétion n'assure la stabilité de $\overline{\wedge}$ et $\overline{\vee}$ que pour des paires d'éléments. La complétion est une notion souvent sous-entendue, elle est pourtant **capitale** dans la mesure où elle permet d'écrire des opérateurs sur des sous-ensembles quelconques. Les voisinages sont un exemple de sous-ensemble. Une dilatation sur des voisinages, définie comme le supremum des ensembles du voisinage, se base sur la complétion du treillis.

Proposition 2.2 ([DP02, page 46])
Tout treillis fini est complet.

Cette proposition définit le cadre dans lequel nous nous placerons: nous supposerons nos treillis toujours complets. Soit l'ensemble de travail est fini, et dans ce cas la proposition 2.2 l'assure la completion, soit nous travaillerons sur des chaînes fermées et bornées (ou des espaces produits de chaîne fermées bornées). Les structures de chaînes naturelles sont par exemple l'ensemble $\mathbb{R}$, non dénombrable: un intervalle borné et fermé possède par contre une structure de treillis complet, une union dénombrable de tels intervalles également. Les espaces produits de chaînes fermées bornées héritent naturellement de la structure de treillis.

Programmation d'un ordre lexicographique Nous avons utilisé dans le paragraphe décrivant la structure d'image (§2.2.1) le terme *pixel*. La structure de pixel ne bénéficie pas directement d'une relation d'ordre, car les possibilités sont beaucoup trop importantes pour pouvoir les fixer définitivement. Ce qu'il est par contre possible de faire est d'*augmenter* la structure de pixel de telle manière à ce que cette nouvelle structure propose une relation d'ordre. Ce que l'on suppose par contre acquis est que chaque canal du pixel bénéficie indépendamment d'une structure d'ordre. Nous avons vu qu'une relation d'ordre lexicographique est la composition des trois entités suivantes:

1. le pixel: les opérations élémentaires vont être héritées de la structure de pixel définie
2. une permutation: la permutation code informatiquement et de manière irréductible l'ordre sur les différents canaux de la structure de pixel.
3. un vecteur d'ordre: chaque élément du vecteur code la relation d'ordre utilisé pour chaque canal indépendamment. Si l'on imagine par exemple un canal circulaire et deux canaux scalaires, la relation d'ordre ne peut pas être la même pour tous les canaux.

Remarque Il y a essentiellement deux manières d'utiliser la notion d'ordre sur l'espace image $\mathbf{F}$: soit on enrichit la structure de pixel - *non ordonnée* - avec une relation d'ordre [(xiv)], soit on modifie légèrement les algorithmes de manière à ce qu'ils utilisent des fonctions d'ordre lorsqu'ils doivent comparer deux pixels. La première méthode a pour avantage qu'une fois l'ordre établit sur le pixel, cet ordre se propage dans l'ensemble de la librairie. L'inconvénient majeur de cette approche est qu'elle rend difficile l'utilisation de plusieurs ordres au sein d'une même librairie. Pour cette raison, nous préférerons dans la mesure du possible la deuxième méthode: l'ordre devient alors un paramètre de l'algorithme. Outre l'utilisation concurrente de plusieurs ordres, cette approche réduit également les erreurs inhérentes à la manipulation d'ordre implicites.

2.2.5. Opérateurs

Le terme « opérateur » peut être compris comme *fonction*, avec une connotation mathématique. On parle ainsi d'opérateur arithmétique, algébrique, linéaire, de voisinage (topologie) etc.. D'un point de vue informatique, les opérateurs présentent un certain nombre d'entrées, qu'ils transforment en un certain nombre de sorties. On définit par *arité* le nombre d'entrées de l'opérateur, mais ce terme n'informe pas sur la nature des opérations effectuées. De ce fait, en Morphologie Mathématique on classe les opérateurs en deux catégories fondamentales:

[(xiv)] par exemple par une classe dérivée du pixel, et implémentant les opérateurs d'ordre naturels

1. les opérateurs de points
2. les opérateurs de voisinage

Les opérateurs de point n'ont pas de notion de voisinage, ils opèrent sur des points pris indépendamment de leur contexte. Les opérateurs de voisinage font intervenir une notion de voisinage. Ce voisinage peut être de différente nature. Nous avons vu que les sous-ensembles de voisinage faisaient intervenir une fonction de centrage de voisinage. Ainsi un opérateur de changement d'espace couleur opère sur chaque éléments de $\mathbf{E}$, depuis un espace F_1 vers un espace F_2. Un opérateur de voisinage, tel qu'une érosion par un élément structurant plan, devra extraire un sous-ensemble $\mathbf{E}'$ de $\mathbf{E}$ en chaque point de $\mathbf{E}$, puis effectuer l'infimum sur l'image de $\mathbf{E}'$ et enfin restituer le résultat en son centre.

2.2.5.1. Opérateur de point

Les opérateurs de point peuvent s'exprimer sous la forme suivante:

$$\forall p \in \mathbf{E} : \mathcal{I}_o(p) = f\left(\mathcal{I}_{i_1}(p), \mathcal{I}_{i_2}(p), \ldots, \mathcal{I}_{i_n}(p)\right)$$

ou même plus simplement:

$$\mathcal{I}_o = f\left(\mathcal{I}_{i_1}, \mathcal{I}_{i_2}, \ldots, \mathcal{I}_{i_n}\right)$$

$\mathcal{I}_o$ est l'image de résultat, $(\mathcal{I}_{i_k})_{k\in\mathbb{N}_n^*}$ les images d'entrée, et f l'opérateur de point. $\mathbf{E}$ est l'espace des points commun à toutes les images, entrées ou sortie. Dans ce cas, f est bien entendu un opérateur d'arité n. Puisque nous avons une méthode universelle d'accès aux points, nous nous servons de celle-ci pour écrire un patron de fonction:

```
1  template <class t_im_i1, class t_im_o, class f_operator>
2          struct s_unary_operator {
3
4          void operator()(const t_im_i1& im_in, f_operator f,  t_im_o& im_out) const
5          {
6                  // récupération des itérateurs  respectivement au début et à la  fin  de "im_in"
7                  typename t_im_i1::const_iterator  it_i  = im_in.begin(),  it_i_end  = im_in.end();
8
9                  // récupération des itérateurs  de "im_out"
10                 typename t_im_o::iterator it_o  = im_out.begin(),  it_o_end = im_out.end();
11
12                 // boucle sur les  sous-ensembles (itération)
13                 for (;
14                      ( it_i  !=  it_i_end)  && (it_o  !=  it_o_end);  // épuisement d'un des sous-ensembles
15                      ++it_i ,  ++it_o)  // avancement des itérateurs
16                 {
17                     // appel de la  fonction  f  sur  l'image du point en entrée
18                     //  résultat  dans un point de l'image de sortie
19                     *it_o  = f(* it_i );
20                 }
21                 return;
22         }
23 };
```

Le patron de fonction donne un squelette d'algorithme pour une arité définie, unaire dans l'exemple (ligne 19). Il est utilisable de manière analogue pour toutes les fonctions d'arité 1. Les exemples d'utilisation d'un patron de fonction unaire sont très nombreux: les fonctions de transformation d'espace couleur, ou *arithmétique et logique* unaire (inversion logique, négation, valeur absolue, etc.). Il est également possible d'utiliser la fonction « f » du patron précédent comme *objet*. « f » contient alors des informations supplémentaires, comme par exemple une valeur constante. Le patron algorithmique fonctionne de la même manière, mais la fonction appelant ce patron a une connaissance de cette constante. D'autres fonctions légèrement plus complexe sont alors possibles, comme l'addition d'une constante, la complémentation par rapport à cette constante etc.. Il est en fait possible de définir de nombreuses fonctions à partir de ce simple patron, à partir du moment où seule une seule image est en entrée.

Spécialisation La librairie est organisée de manière à ce que tous les opérateurs de point utilisent les patrons d'algorithme décrits dans le paragraphe précédent. La centralisation de ces patrons permet deux choses très importantes:

- lorsque les conditions le permettent, notamment en terme de géométrie, il est possible de spécialiser des fonctions optimisées dans le patron d'algorithme. Ces spécialisations ont une connaissance fine de la structure d'image et dépendent directement de l'organisation mémoire des images.
- seules les spécialisations du patron principal sont informées de l'organisation en mémoire de l'image, ce qui n'est pas le cas pour les fonctions de point: ceci préserve les mécanismes d'abstraction que nous nous sommes fixés dans le contexte de la *POO*.

L'optimisation est justifiée par le fait que l'utilisation d'itérateurs entraîne un ralentissement global de l'accès aux images. Ce ralentissement peut être assez important et provient de la gestion interne de la géométrie de l'image par l'itérateur. Dans ce cas, le traitement est trop complexe pour permettre au compilateur d'effectuer des optimisations de code.

Exemple de spécialisation sur l'adéquation de géométrie: Reprenons l'exemple de l'opérateur unaire. Lorsque les deux images possèdent une géométrie similaire (par exemple des rectangles de taille égale en 2D), les coordonnées des deux images génèrent les mêmes offsets. Il n'est plus besoin d'utiliser deux itérateurs, mais un seul. Cet itérateurs génère les offsets du traitement, qui sont utilisés ensuite pour un accès rapide aux points pour les deux images. On économise ainsi le coût de gestion d'un itérateur. Le test d'adéquation des offsets se fait lors de l'exécution, et peut être vu comme une branche particulière du cas générique.

Lorsque l'image d'entrée et de sortie sont de même dimension (toutes les deux 3D par exemple), l'organisation des données en mémoire est supposée connue. La génération des offsets - ie. l'accès aux points - est simple et peut être effectuée directement par l'opérateur unaire. Pour les images 3D, la génération des offsets se fait selon une itération imbriquée sur les trois axes de l'image (triple boucle « for »). La fonction f est ensuite appliquée sur chaque point. La génération de l'offset de cette manière est moins lourde en calcul et offre une visibilité plus grande au compilateur. Les nouveaux compilateurs reconnaissent parfois ce type d'imbrication de boucle, et utilisent un jeu d'instruction étendu pour l'application de la fonction unaire f sur des paquets de points, capitalisant de cette manière l'accès en mémoire.

Exemple de spécialisation sur l'adéquation de type : La recopie d'image: lorsque les images sont de type différent, T_{i_1} et T_o (par exemple respectivement $\mathbb{N}_{256}$ et à virgule flottante), la recopie d'image de chaque pixels de la première image vers la seconde doit passer par un changement de type de T_{i_1} vers T_o, ce qui est assez coûteux. L'algorithme généraliste lit chaque pixel de $\mathcal{I}_1$, transforme le type, puis l'affecte à $\mathcal{I}_o$, en se basant sur les itérateurs. Donc chaque pixel coûte une lecture, une écriture, une opération de comparaison et enfin une opération d'incrémentation (itérateur). Lorsque la condition $T_{i_1} = T_o$ est vérifiée, la conversion de type n'est plus nécessaire. L'organisation des images étant la même, si les images ont une géométrie similaire (ie. les pavés sont égaux), il est possible d'utiliser des méthodes de recopie de zone mémoire au lieu d'utiliser des méthodes de copie par point. Les méthodes de copie mémoire utilisent en générale des instructions spécifiques à l'architecture, tirent profit de jeux d'instruction étendu et gèrent de manière fine l'occupation du bus mémoire. Pour brancher automatiquement sur ce cas, il suffit d'écrire une spécialisation de la fonction de copie d'image lorsque les types T_{i_1} et T_o sont égaux. La spécialisation teste alors la condition sur la géométrie, et utilise ou non les fonctions de transfert rapide en mémoire. Les autres fonctions utilisant la fonction de recopie restent inchangées, et profitent naturellement de cette spécialisation.

Nous nous rendons compte encore une fois de la puissance de ce mécanisme de spécialisation, d'autant plus que son développement n'est pas nécessaire puisque le patron générique suffit au fonctionnement de tous les opérateurs de point. Les optimisations sont toutefois un atout majeur pour une librairie de traitement d'image. Nous reportons le lecteur aux travaux de Brambor [Bra06] pour des optimisations spécifiques à des architectures matérielles orientées flux.

2.2.5.2. Opérateur de voisinage

Nous avons précédemment décrit les voisinages et leur représentation informatique universelle. Passons sur l'initialisation du voisinage: la génération du voisinage est effectuée par centrage de l'application de voisinage, c'est à dire par l'appel d'une fonction permettant de centrer ce voisinage. Le voisinage génère de manière interne les voisins, et offre des itérateurs pour les parcourir.

Selon cette architecture, un algorithme $\mathcal{A}$ de type *collecte sur voisinage*, transformant une image $\mathcal{I}_{i_1} : \mathbf{E} \rightarrow F_{i_1}$ en une image $\mathcal{I}_o : \mathbf{E} \rightarrow F_o$, selon une fonction f appliquée sur les voisinages $\mathcal{N}$ de chaque site de $\mathcal{I}_{i_1}$, et s'écrivant de la manière suivante:

$$\mathcal{A} : \left\{ \begin{array}{lcl} F_{i_1}^{\mathbf{E}} & \rightarrow & F_o^{\mathbf{E}} \\ \mathcal{I} & \mapsto & \mathcal{AI} \ / \ \forall p \in \mathbf{E}, (\mathcal{AI})(p) = f(\mathcal{I}(\mathcal{N}_p)) \end{array} \right.$$

s'écrit informatiquement de la manière suivante:

```
1  template <class t_im_i1, class SE, , class f_operator, class t_im_o>
2          struct s_neighborhood_unary_operator {
3
4          void operator()(const t_im_i1& im_in, const SE& se, f_operator f, t_im_o& im_out) const
5          {
6                  // récupération des itérateurs sur les images
7                  typename t_im_i1::const_iterator it_i = im_in.begin(), it_i_end = im_in.end();
8                  typename t_im_o::iterator it_o = im_out.begin(), it_o_end = im_out.end();
9
10                 // création du voisinage, selon une structure "se", et sur l'image "im_in"
11                 Neighborhood<SE, t_im_i1> nh(im_in, se);
12
13                 // boucle sur les sous-ensembles (itération)
14                 for(;
15                     ( it_i != it_i_end) && (it_o != it_o_end); // épuisement d'un des sous-ensembles
16                     ++it_i , ++it_o) // avancement des itérateurs
17                 {
18                     // centrage du voisinage
19                     nh.center( it_i );
20
21                     // appel de la fonction f sur le voisinage (nh.begin(), nh.end()) de l'image d'entrée
22                     // résultat dans un point de l'image de sortie
23                     *it_o = f(nh.begin(), nh.end());
24                 }
25                 return;
26         }
27 };
```

Le code ci-dessus résume l'essentiel de l'approche. Le voisinage doit être créé en fonction de l'image et de l'élément structurant, ce qui est fait ligne 11. Par rapport aux opérateurs de points, un niveau d'indirection supplémentaire pour le centrage du voisinage - sur chaque point de l'image - est nécessaire (ligne 19). Ensuite, l'appel d'un opérateur sur un sous-ensemble permet de traiter le voisinage (ligne 23). L'opérateur « f_operator » est dans ce cas un opérateur agissant sur un sous-ensemble de taille quelconque, et non d'arité fixe comme c'était le cas pour les opérateurs de point. Pour ne pas avoir à typer l'itérateur en entrée de f, il suffit de l'écrire sous la forme de template d'itérateur.

```
1  template <class It_>
2          struct s_fonction
3  {
4          typedef typename it::value_type result_type;
5          result_type operator()(It_ it , It_ end){
6                  /* seules les méthodes communes à tous les itérateurs sont
7                   utilisées dans le corps de la fonction */
8          }
9  };
```

Puisque les itérateurs implémentent tous les mêmes méthodes, le compilateur saura ensuite compiler le code de manière adéquate en fonction de la nature de l'itérateur, et indirectement en fonction du type de voisinage. La fonction « f » doit néanmoins satisfaire une condition importante. En effet, l'ordre des voisins n'est pas une propriété des itérateurs. L'opérateur « f » doit donc être indépendant de l'ordre de

construction du voisinage. Si l'opérateur est composé d'un processus séquentiel et peut s'écrire sous la forme donnée en algorithme 2.1, alors la commutativité et l'associativité de l'application binaire « $\star$ » sont des propriétés *suffisantes*. Ces propriétés sont très fortes, mais assez courante en MM. Il ne sera par contre pas possible de définir un gradient directionnel avec un tel opérateur.

Algorithme 2.1 : Opérateur séquentiel de voisinage

Data : $V = \{v_1, \ldots, v_n\}$: ensemble de voisins, h fonction de terminaison
1 $v \leftarrow v_1$
2 **forall** $i \in \mathbb{N}_n \setminus \{0, 1\}$ **do** $v \leftarrow v_i \star v$
3 **return** $h(v)$

Exemple Soit l'opérateur d'infimum

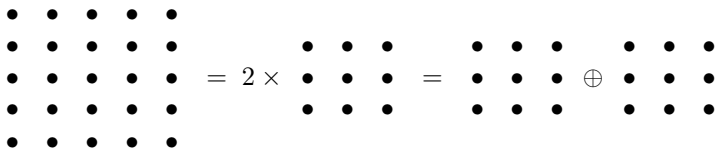
$$\bigwedge : \begin{cases} \mathcal{P}(\mathbf{F}) & \rightarrow & \mathbf{F} \\ v = \{v_i, i \in \mathbb{N}_n^*\} & \mapsto & \bigwedge_{i \in \mathbb{N}_n^*} v_i \end{cases}$$

Il s'écrit très simplement de manière séquentielle selon l'algorithme 2.1 avec $\star \equiv \wedge$. Appliqué à un voisinage centré sur chacun des points, nous obtenons l'érosion par un élément structurant plan, définie par l'objet « ne ». De manière analogue, l'application de $\bigvee$ sur le voisinage donne la dilatation par le transposé de « ne ». Les autres applications immédiates sont les filtres de rang - dont le filtre médian -, les demi-gradients supérieur et inférieur et le gradient morphologique [(xv)], des statistiques locales (moyenne, variance ou moment d'ordre quelconque), etc.. Pour l'ensemble de ces fonctions locales ou de voisinage, le squelette principal a été écrit pour l'ensemble des voisinages connus de la librairie. Il suffit au chercheur d'implémenter correctement la fonction à appliquer sur chaque site (point de l'image induisant ensuite un voisinage).

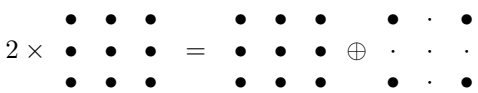
Spécialisation sur les types de voisinage L'exemple précédent présente l'application très proche du pixel. Il est également possible d'effectuer des spécialisations pour des traitements de plus haut niveau. Les éléments structurants dits homothétiques tirent avantage de ce cadre. Un élément structurant S_h est homothétique s'il peut être déduit d'un élément structurant plus petit S_u par homothétie (appelons ℓ le rapport). D'un point de vue ensembliste cela suppose que S_h (et donc S_u) est compact et convexe. Par associativité de l'addition de Minkowski et pour $\ell \in \mathbb{N}^*$:

$$S_h = \ell \times S_u = \oplus_{i \in \mathbb{N}_\ell^*} S_u$$

L'avantage de cette représentation demeure dans la réduction des calculs. Ainsi, l'éléments homothétique 5×5 se décompose de la manière suivante:



ce qui conduit à $2 \times 9 = 18$ opérations par site au lieu $5 \times 5 = 25$. Les décompositions peuvent être plus astucieuses:



(2.1)

Soit $9 + 4 = 13$ opérations élémentaires pour obtenir le même résultat, puisque l'addition de Minkowski est commutative. Par ailleurs, puisque

$$X \ominus (S_1 \oplus S_2) = (X \ominus S_1) \ominus S_2$$

[(xv)] la définition exacte du gradient morphologique n'est cependant pas respectée lors d'une implémentation de la différence entre $\bigvee$ et $\bigwedge$ directement dans l'opérateur de voisinage, à cause de la transposition du graphe de connexité

les décompositions sont utilisables de manière analogue pour les soustractions de Minkowski.

Les algorithmes de décomposition ne sont pas triviaux (voir par exemple [FHB03]), mais certaines trames (8 ou 4 connexités) sont récurrentes. Il est possible de tirer profit de ces simplifications de manière transparente à l'aide des spécialisations. La spécialisation de l'opérateur d'érosion pour l'élément structurant de type homothétique appelle classiquement ℓ fois l'érosion avec l'élément structurant unitaire S_u (même chose pour la dilatation). Il est alors inutile de spécialiser les opérateurs d'ouverture et de fermeture puisqu'ils utiliseront naturellement l'érosion ou la dilatation spécialisées si l'élément structurant l'impose. La logique de la boucle d'homothétie peut être faite ensuite de manière plus astucieuse: selon les remarques sur la décomposition, il est possible d'ajouter des tests sur l'élément structurant unitaire S_u utilisé. S'il possède une composition connue, comme sur l'exemple de l'équation 2.1, la logique d'homothétie utilisera cette composition de manière transparente. Ces simplifications de calcul se propagent alors de manière naturelle à toutes les opérations dépendantes des éléments structurants homothétique: érosions-dilatations, ouverture-fermetures, gradients morphologiques épais, alternés séquentiels, etc.. Bien que ces compositions soient des cas particuliers en nombre réduit, l'utilisation pratique est grandement améliorée. Enfin, notons que ce type de développement impacte uniquement la logique liée à l'homothétie et non le reste du système.

2.2.5.3. Autre type d'opérateur

Les autres types d'opérateur sont bien sûr très nombreux, mais il s'agit de discuter ici des opérateurs ne fonctionnant pas sur le pixel, et ne pouvant tirer profit de voisinages comme il a été décrit précédemment. Nous pouvons placer dans cette catégorie tous les opérateurs nécessitant une génération de voisinage selon un ordre bien précis. Les opérateurs utilisant par exemple un gradient (classique) selon un des axes entrent dans cette catégorie: le gradient ne peut être calculé simplement à partir d'une extraction du voisinage, une connaissance sur l'ordre des points est nécessaire.

Pour assurer le fonctionnement de ces opérateurs, la gestion des coordonnées doit être faite plus précisément. Puisque cette gestion dépend fortement de l'opérateur concerné, la solution la plus adéquate est d'inclure dans l'opérateur la génération des coordonnées d'intérêt. Ainsi, le gradient générera les coordonnées dont il a besoin pour son calcul, et ne fera pas appel à des itérateurs de voisinage. Les filtres diffusifs entrent également dans cette catégorie.

Nous ne détaillerons pas davantage ces opérateurs.

2.2.6. Les files d'attente hiérarchiques

Une file d'attente hiérarchique, ou *fah* , est une structure informatique utilisée dans certains algorithmes. Elle est souvent la clef de voûte pour ce qui est des performances de temps de calcul. Elle permet généralement de ne s'intéresser qu'à un sous-ensemble du support $\mathbf{E}$ de l'image. Elle offre cependant la possibilité de créer un ordre dans ce sous-ensemble et se distingue ainsi d'une liste simple. L'ordre induit est précisément assimilé à la hiérarchie.

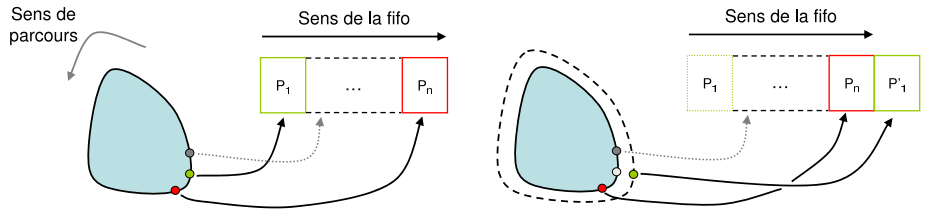
Files d'attente simples Classiquement, les files d'attente « simples » se définissent par un ordre sur l'arrivée temporelle/séquentielle des points dans la file d'attente.On distingue alors les types de files suivants:

- *fifo*- « first in, first out ». Le premier élément sorti est le premier arrivé, l'ordre est conservé.
- *lifo*- « last in, first out ». Le premier élément sorti est le dernier arrivé, l'ordre est totalement inversé.

Il existe des structures *génériques* simples permettant d'utiliser ces deux comportements. La généricité se manifeste sur le type stocké, mais le comportement reste explicite. Les vecteurs, les listes chaînées, les « files »... sont des exemples de structure *générique* pouvant être utilisées en tant que file d'attente simple (cf. §2.1.2).

Remarque: Idéalement, les algorithmes en MM ne doivent pas être sensibles à l'ordre d'insertion des points. La raison est la suivante: les files d'attente simples ont été utilisées pour définir des lacets (cf. par exemple les travaux de L. Vincent [Vin90]). Les lacets peuvent être considérés comme étant des contours d'ensembles fermés. Chaque élément de la file d'attente représente alors un point du lacet (figure 2.6(a)). L'avantage du codage des ensembles sous forme de lacet réside dans le fait que ce codage est minimal: les points du lacet permettent de déduire la totalité de l'ensemble et sont les points pertinents lorsque le traitement agit sur le contour de l'ensemble.

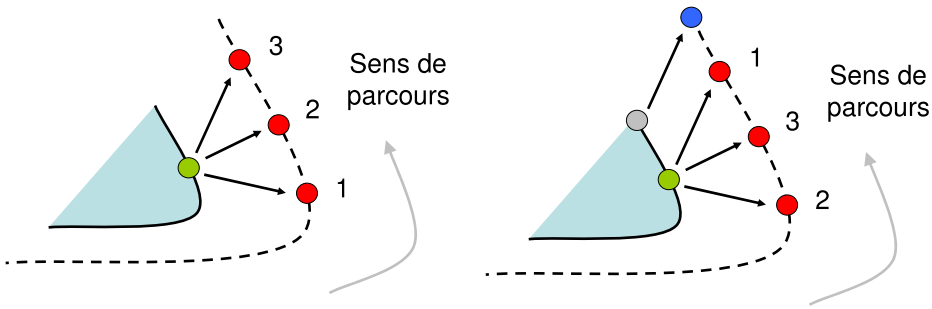
L'agencement des données sous forme de lacet est souvent le préalable à une propagation. La propagation est simplement un élargissement du contour sur les points voisins et vers l'extérieur de l'ensemble considéré: c'est en quelque sorte une manière astucieuse d'effectuer une *addition de Minkowski*.



(a) Configuration initiale sous forme de lacet (b) Étape élémentaire de propagation

Fig. 2.6.: Codage d'un contour à l'aide d'un lacet. (a): le contour initial et la *fifo* utilisée pour la coder. Un sens de parcours arbitraire est supposé; le point vert est le premier point, le point rouge le dernier. (b): début de la propagation de la courbe de départ: P_1 est retiré de la *fifo*, sa propagation donne lieu à un nouveau point P_1' en fin de file.

Une seule file est généralement utilisée pour la propagation: le premier point est supprimé de la file et ses voisins sont ajoutés en fin de file (figure 2.6(b)). Seulement, la recherche des voisins est effectuée selon un graphe de connexité, et la découverte des voisins est dépendante de l'implémentation de ce graphe. La figure 2.7 montre deux graphes de voisinage qui doivent effectuer la même propagation. Si la file d'attente utilisée était du type *lifo*, le point traité à une étape $n+1$ serait le premier voisin du premier point de l'étape n et la propagation ne se ferait que dans une direction.



(a) Recherche de nouveaux voisins adaptée au lacet (b) Recherche de nouveaux voisins non adaptée au lacet

Fig. 2.7.: Adéquation entre le voisinage et le codage de la courbe en *fifo*. (a): la recherche de nouveaux voisins préserve l'ordre des points pour un codage *fifo*. (b): le point bleu voisin de gris n'est pas connexe au dernier point voisin de vert: l'ordre est détruit.

Deux moyens existent pour rendre la propagation isotrope: utiliser une *fifo*, ou ne pas faire d'hypothèse sur l'ordre au niveau algorithmique. La première solution permet d'obtenir l'isotropie dans certains cas simples, mais cette propriété est mise en échec dans des cas plus complexes. L'exemple de la figure 2.7(b) en est un, dont la résolution est encore plus difficile en dimension supérieure; en dimension 3, les lacets deviennent des enveloppes, et plusieurs ordres de parcours sont possibles: l'isotropie devient

alors assez hasardeuse.

La deuxième solution est bien sûr idéale, mais plus difficile à obtenir en des termes algorithmiques. Elle présente également le défaut de forcer des contrôles supplémentaires, ce qui a généralement pour conséquence un ralentissement de l'algorithme. Toutefois, elle présente la particularité très intéressante d'être extensible à n dimensions de manière complètement naturelle. Nous qualifierons ce genre de propagation d'**atomique**, dans le sens où le traitement des points de l'étape $n+1$ ne se fait qu'après considération de la totalité des voisins de l'étape n.

Nous verrons dans le chapitre 5 un cas concret et subtil de défaut d'isotropie pour l'algorithme classique de ligne de partage des eaux. Pour l'heure, considérons simplement et définitivement une file d'attente simple comme une structure représentant un ensemble non-ordonné de points, et permettant des opérations d'ajout et de retrait de points de manière extrêmement simple, sans aucune propriété supplémentaire.

Files d'attente hiérarchiques La hiérarchie, c'est à dire la structure d'ordre, n'est pas directement donnée par les files d'attente simple. Elle est apportée par une structure d'ordre supplémentant un ensemble de files d'attente simples. Deux espaces sont nécessaires à la définition totale de *fah*:

1. Δ : l'espace des données.
2. Ω : l'espace des priorités.

L'*espace des données* contient les éléments qui seront stockés dans les différentes files d'attente simples, et dont les propriétés sont communes à tous les niveaux de priorités. Il s'agira du type utilisé pour le patron de file d'attente simple. Cela peut être par exemple, une coordonnée dans l'espace $\mathbf{E}$ de dimension n, une valeur, un décalage par rapport au début de l'image, une structure composée, etc.. Plus généralement, ce type représentera les données sur lesquelles travaille un algorithme.

L'*espace des priorités* est l'espace sur lequel va s'exprimer la hiérarchie. Il s'agit par exemple, pour les algorithmes de type *lpe* ou reconstruction numérique, de la valeur de gris du point inséré dans la file. Chaque élément inséré dans la file d'attente hiérarchique est un élément de $\Omega \times \Delta$. Lorsqu'un nouvel élément est inséré dans la *fah*, la valeur de l'élément résout la file d'attente simple dans laquelle insérer la donnée de l'élément. Une file d'attente hiérarchique est donc un ensemble variable de paires, formées d'une priorité et d'une file d'attente simple :

$$hq = \{(h_i, q_i)\}_{i \in \mathbb{N}_N}$$

Ω est muni d'une relation d'ordre $\preceq_\Omega$, et nous avons

$$\forall i \in \mathbb{N}_N, \begin{cases} h_i \in \Omega \\ \forall u \in q_i, u \in \Delta \end{cases}$$

Détails d'implémentation et performances Rappelons que nous souhaitons ordonner un ensemble de file d'attente simple. L'implémentation d'une telle structure nécessite une structure générique gérant une relation d'ordre, ce qui est précisément un objet « dictionnaire » dont nous avons détaillé le comportement dans le paragraphe §2.1.2. La structure de *fah* est une combinaison des structures de dictionnaire et file d'attente simple, selon le schéma de la figure 2.8.

La manipulation du dictionnaire implique un surcoût du nombre d'opération effectuées par éléments. Ce surcoût intervient lorsque l'on doit accéder à l'espace des priorités. Les opérations suivantes sont concernées:

1. insertion d'un élément (paire $\{(h, q)\} \in \Omega \times \Delta$)
2. recherche d'un élément
3. recherche d'un plateau (ie. ensemble des éléments de même priorité)

Lorsqu'un algorithme fait la requête d'un point unique à partir de sa priorité, le renseignement de la priorité implique plus d'information transférée et une recherche dans le dictionnaire. Pour diminuer ce surcoût, une méthode serait d'extraire en un unique accès tous les éléments d'une même priorité (itération à priorité constante); cette méthode serait utilisable par exemple pour l'algorithme de ligne

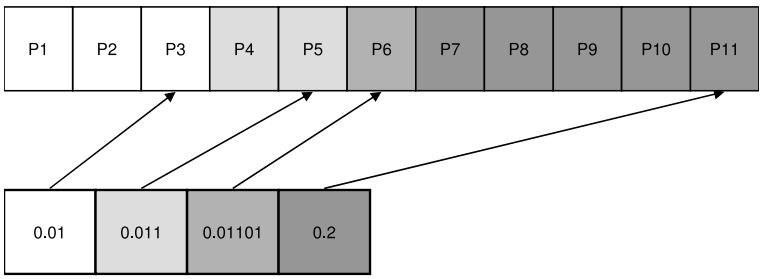


Fig. 2.8.: Structure de *fah* générique. Les points $P1$ à $P11$ représentent les données de l'espace Δ. La structure associée est une liste de la *STL*, permettant d'insérer à des coûts quasi-nuls des nouveaux points, que cela soit à une priorité existante ou non. Les valeurs 0.01, 0.011, 0.01101 et 0.2 représentent les priorités dans l'espace Ω des différents plateaux. Ces priorités sont associées aux points $P1$ à $P11$. La structure utilisée pour la représentation de ces valeurs est le dictionnaire de la *STL*. Les valeurs ici sont choisies sur ce schéma sont volontairement des nombres réels, mais il est tout à fait possible d'avoir des valeurs vectorielles à partir du moment où le dictionnaire est informé d'une relation d'ordre sur Ω.

de partage des eaux. Le surcoût est alors réduit puisque le coût de recherche de la priorité est dilué par le nombre d'éléments du plateau. Le surcoût critique concerne l'insertion d'un élément.

Pour chaque insertion, il est nécessaire d'effectuer les opérations décrites dans l'algorithme 2.2. L'insertion d'un nouvel élément (h, q) s'accompagne donc de la recherche parmi les priorités de la file f de la priorité h (en $O(\ln N)$) et des coûts de gestion des files simples. Ces derniers sont par contre limités, puisque l'insertion pour les listes ou vecteurs - en début ou fin de liste - est en $O(1)$.

Algorithme 2.2 : Insertion d'un point (h, q) dans la *fah*

```
1 if h ∈ ensemble des priorités de f then
2 |   fs ← file simple de f à la priorité h;
3 |   fs ← ajout de p;
4 else
5 |   fs ← création d'une nouvelle file d'attente;
6 |   f ← insertion de fs à la priorité h;
7 |   fs ← ajout de p;
8
```

Bien entendu, ce coût de recherche a un impact direct sur les performances des algorithmes utilisant une implémentation à base de *fah*. L'idée émanant de ces remarques est de s'astreindre du coût de recherche dans l'espace de priorités de la *fah*. Cela serait *a priori* possible pour Ω dénombrable, ou même mieux, fini.

File d'attente hiérarchique et spécialisation Nous avons déjà mentionné les spécialisations, et il est possible d'utiliser ce mécanisme également ici. Pour des combinaisons simple de priorité et de relation d'ordre, comme les entiers non signé 8 bits UINT8 [(xvi)] muni de la relation d'ordre naturelle, une implémentation différente de la hiérarchie semble plus efficace. Précisément pour les UINT8, il est possible de s'astreindre de l'utilisation d'un dictionnaire en faveur de celui d'un tableau. Cette implémentation est celle classiquement utilisée (voir figure 2.9).

Il est alors possible de spécialiser l'utilisation des *fah* lorsque Ω est du type UINT8. Cela peut se faire simplement à l'aide d'une classe intermédiaire, précisant le type de *fah* à utiliser.

```
1 // comportement par défaut sur la structure générique à base de dictionnaire
```

(xvi) type encore très répandu pour coder les images à niveau de gris

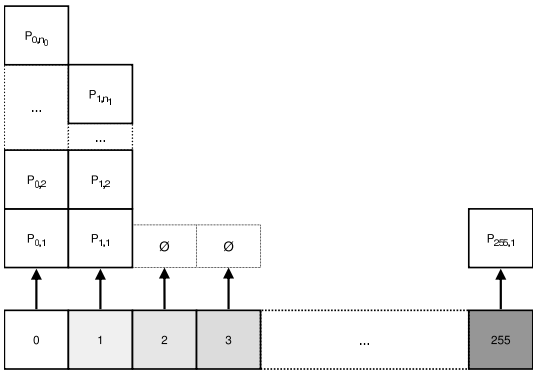


Fig. 2.9.: File d'attente à hiérarchie discrète

```
template <class C, class T, class order_operator = std::less<_Key> >
  struct pq_strategy {
      typedef common_priority_queue<C, T, order_operator>      PQ;
};

// Spécialisation pour le cas entier 8 bit et relation d'ordre "strictement inférieur à"
template <class T>
  struct pq_strategy<UINT8, T, std::less<UINT8> > {
      typedef discrete_priority_queue<UINT8, T, std::less<UINT8> >       PQ;
};
```

Un algorithme client accède ensuite de manière *universelle* aux *fah* de la manière suivante:

```
typedef typename pq_strategy<C, T, order_operator>::PQ priority_queue;
```

« C » code la valeur de priorité (espace Ω): pour un algorithme de ligne de partage des eaux, il s'agit du type de l'image de relief. « T » est le type stocké dans chacune des files d'attente simple (espace Δ). Il s'agit souvent de la position d'un point du front d'onde.

Nous n'avons pas comparé de manière quantitative les performances des deux implémentations de *fah*. Dans le cas 8 bits, la file utilisant un tableau de correspondance s'avère plus rapide que la *fah* générique. Mais les différences ne sont pas grandes. La raison de cela est que souvent, seul un nombre réduit de points est présent dans la file (front d'onde). Parmi ce nombre réduit, il y a un nombre encore plus réduit de priorités différentes, et les coûts de recherche (en $O(\ln N)$ avec N le nombre de priorités) restent limités. L'efficacité supérieure du tableau de correspondance n'est plus vrai dans le cas 16 bits, et cela pour plusieurs raisons. Premièrement, bien que l'accès à une priorité dans le cas d'un tableau de correspondance soit immédiat, il faut stocker la totalité du tableau (soit 2^{16} éléments) alors que seul un nombre limité de priorités est présent; d'où des utilisations de ressources mémoires inutiles. Deuxièmement, lorsque l'on passe d'un plateau au suivant, il faut connaître la priorité suivante pour pouvoir continuer l'algorithme. Cette recherche est extrêmement coûteuse car il faut tester un grand nombre de priorités avant de détecter une priorité non-vide. Dans le cas d'une image à deux valeurs, mais codée sur 16 bits, le comportement de la *fah* générique est celui attendu - à savoir rapide - puisque les coûts de recherche sont quasi-négligeable et les coût d'insertion constants; alors que la file basée sur le tableau de correspondance est extrêmement lente.

Puisque les images sont à support fini, le nombre de priorités N est borné par le nombre de points dans l'image. Une idée intéressante serait dans un premier temps de déterminer N puis d'établir un tableau de correspondance à deux indirections; par exemple un tableau de 2^{16} éléments pointant dans un tableau de N éléments. Cependant les performances de la *fah* nous ont parut suffisamment proches de celles de la *fah* spécialisée, et nous n'avons pas effectué de tels développements. Nous laissons donc ces travaux en perspective.

Remarques Nous anticipons légèrement en ajoutant deux fonctionnalités des *fah* vis à vis de certains algorithmes de MM. La première fonctionnalité est l'accès à la totalité d'une priorité (priorité

considérée comme une hauteur du relief dans l'algorithme de ligne de partage des eaux). La deuxième est l'utilisation d'un jeton bloquant l'insertion de nouveaux points dans la file d'attente. La raison de ce jeton sera détaillée dans l'algorithme de *lpe* (cf. chapitre 5 page 177) et servira de point dur pour améliorer l'isotropie des algorithmes.

Enfin, nous avons précisé le fait que l'ordre sur les dictionnaires doit être **strict** et **faible**. C'est le cas pour les types à virgule flottante gérés par les compilateurs. Il sera donc possible d'utiliser cette même implémentation de file d'attente hiérarchique pour l'utilisation d'algorithmes sur des images à valuation flottante.

2.2.7. Conclusion

Dans cette section, nous avons passé en revue quelques mécanismes offert par un outil nouveau: la méta-programmation. Nous avons exprimé quelques structures très utilisées en Morphologie Mathématique au moyen de ce cadre. Ces structures seules n'interagissent pas, ou très peu, entre elles. Cet isolement est totalement intentionnel: les structures constituent les briques de base à toute conception algorithmique qui, par opposition, sont dépendantes d'un nombre important de structures.

2.3 Conception d'algorithmes

La partie précédente était consacrée à la description de structures utilisées dans le traitement informatique des images. Les algorithmes sont des structures informatiques de traitement. Elles sont clientes des structures de données, et naturellement celles décrites dans la section précédente sont celles qui interviendront dans les algorithmes de Morphologie Mathématique.

Du fait de l'intervention de nombreuses structures au sein d'un même algorithme, les schémas de dépendance informatique deviennent plus complexes. Les mécanismes de méta-programmation permettent de lier les dépendances au niveau « structure informatique ». C'est sur ce point qu'interviennent par exemple les champs de type précédemment décrits.

Trois points nous intéressent particulièrement:

1. l'approche des patrons d'algorithme au niveau structurel, et les méthodes pour abstraire les structures de données traitées
2. la séparation du squelette d'un algorithme avec des structures fonctionnelles tierces modifiant le comportement
3. la mise en relief entre certaines notions mathématiques et l'écriture algorithmique correspondante

La section s'articule de la manière suivante: dans un premier temps, nous allons utiliser les développements concernant les méta-structures informatiques et les mécanisme de méta-programmation. L'objectif est de montrer comment la méta-programmation répond de manière efficace aux besoins existants, et capitalise les possibilités applicatives sur des développements réduits. Nous verrons ensuite de quelle manière il est possible d'appliquer la méta-programmation sur les structures algorithmiques. Nous clarifierons notre approche par un exemple concret de *labellisation* de composantes connexes.

2.3.1. Approche mathématique de l'écriture algorithmique

Dans la mesure où un algorithme est client de nombreuses autres structures, cette partie aura pour rôle essentiel la synthèse des éléments qui ont été développés dans le paragraphe §2.2.

2.3.1.1. Données traitées

La méta-programmation fait naturellement abstraction des types traités. Cependant, il est parfois nécessaire d'avoir un certain contrôle sur ces types, sans toutefois se contraindre au typage explicite.

Types Nous avons fourni des exemples sur la manipulation de types de données, sans faire intervenir le typage explicite. La manipulation de type se fait essentiellement par les champs de type des structures de données que l'algorithme manipule. Par exemple:

```
1  typename image1::value_type v1 = im1.pixel(offset);
2  typename image2::value_type v2 = im2.pixel(offset);
3  im3.pixel(offset) = v1 + v2;
```

Cette portion de code ne connaît pas explicitement les types des images. Les types des variables sont directement déduits des images, ils sont *fonctions* (ou dépendants) des images. Bien que valide et orienté méta-programmation, ce genre d'écriture entraînera souvent des avertissements de la part du compilateur. Ces problèmes sont liés à la nature de l'addition effectuée, qui nécessite souvent un type de stockage plus grand, *en bits*, que les termes de l'addition. Si l'image d'arrivée, « im3 », est du même type que l'une ou l'autre des images d'entrée, une partie des données risquent d'être perdue; même si le compilateur effectue l'addition de manière à éviter cette perte, le résultat ne peut être stocké en totalité dans l'image. Ceci souligne le fait que la manipulation des types de donnée est certes abstraite, mais délicate.

Si le développeur sait plus ou moins ce qu'il fait, il peut forcer le sous-typage explicitement: certaines données sont alors perdues en toute connaissance de cause de l'utilisateur. Dans le cas général ce n'est pas le cas.

Pour une addition, le contrôle et les divers comportements sont assez faciles à manier. Ce n'est plus le cas pour d'autres type d'opérateur: une labellisation valuant chaque composante connexe par la valeur de sa surface, une fonction intégrale locale, ou une fonction de statistique, peuvent se heurter au problème de choix de type. Ce problème est récurrent, et trouve une solution à l'aide des spécialisations.

Pour chaque type de donnée, il est possible de déléguer le choix de type pour le calcul à une classe externe. Nous avons déjà mentionné une structure informant sur la qualité entière d'un type (§2.1.3 page 14) par une méthode de *spécialisation* et de *champ de type*. Il s'agit ici du même mécanisme. Lorsqu'un algorithme doit *accumuler* sur un type, typiquement pour compter sur ce type, l'algorithme se sert de cette structure pour déduire le type servant à l'accumulation. Ce type sera différent pour les entiers 8 bits, 16 bits, « flottant », vectoriel 16 bits, etc.. L'important ici est qu'il n'incombe pas à l'algorithme de choisir arbitrairement un type, choix qui limiterait son fonctionnement à un nombre fini de cas (et impliquerait son dysfonctionnement dans les autres cas).

Si on appelle cette structure « regle_calcul », un algorithme d'intégration sur un type « T » s'écrira:

```
1  typename regle_calcul<T>::accumulation_pour_somme accu(0);
2  for(; it != it_end; ++it){
3          accu += *it;
4  }
```

L'algorithme ne sachant pas par avance la taille des données sur lesquelles l'intégration devra se faire, le choix ne lui appartient pas. Si « T » est du type « 3 canaux 8 bits non signé », l'accumulation sera en « 3 canaux 32 bits non signé ».

Coordonnées nous séparons volontairement les coordonnées des types de données manipulés. Nous avons vu que la structure d'image permettait d'avoir une représentation spéciale d'une coordonnée, selon un type universel qu'est « l'offset ». La plupart du temps, les algorithmes ne manipuleront que les « offsets »; écrire en effet :

$$\forall p \in \mathbf{E}, \ldots$$

sans interroger un point p sur sa valeur selon une dimension, rend la représentation vectorielle inutile. Pour ce genre de situation, l'algorithme peut être considéré comme un **substrat** sur lequel *transitent* des coordonnées d'une structure vers une autre. L'algorithme ne génère pas de coordonnées et se contente de les passer d'une structure à une autre, par exemple d'un itérateur d'image à un voisinage. Cette analogie peut être appliquée également aux types de données.

Certains algorithmes doivent toutefois manipuler et générer des coordonnées, par exemple pour le calcul d'une dérivée selon un axe, etc.. Pour les structures autres que celles manipulant l'espace de coordonnées $\mathbf{E}$ (images, voisinages sur image), les coordonnées n'ont pas de sens.

2.3.1.2. Parcours de contenu

Lorsqu'un algorithme a besoin de parcourir les éléments d'un conteneur, image, graphe ou autre, le mécanisme des itérateurs apporte une solution générique.

Lorsqu'un algorithme a besoin de parcourir tous les éléments d'une image, il doit s'informer sur son support $\mathbf{E}$ et connaître un moyen de parcourir ce support. Si cela est simple pour une image puisque le domaine est de type *pavé*, ce n'est pas le cas pour un graphe. Pourtant l'action est identique.

Les itérateurs répondent à ce besoin: le mécanisme de découverte de domaine n'appartient plus à l'algorithme. Ainsi, écrire:

$$\forall p \in \mathbf{supp}(\mathcal{I}), \dots$$

revient informatiquement à écrire, à l'aide des itérateurs:

```
1 typename type_entree::iterator it = instance_entree.begin(), it_end = instance_entree.end();
2 for(; it != it_end; ++it){
3         /* traitement */
4 }
```

où « type_entree » est le type de la structure devant être parcourue, et « instance_entree » est une instance de ce type. Insistons bien sur le fait que cela s'applique à toutes les structures de conteneur: image, graphe, voisinage, file simple, file hiérarchique, etc.. Sans être péjoratif ni réducteur, de nombreux algorithmes se résument à des parcours imbriqués de points. Il ne faut pas négliger cet aspect dans la conception algorithmique.

Voisinages Les voisinages sont fréquemment rencontrés dans nos algorithmes. Ils ont principalement deux usages: ils servent soit à définir une famille de points sur laquelle une fonction est calculée, soit à créer un nouvel ensemble de points selon certaines conditions. Dans la première catégorie se placent les algorithmes classiques de voisinages, érosion/dilatation etc., alors que la deuxième catégorie contient tous les algorithmes dits à *propagation*. Nous verrons un tel exemple d'algorithme grâce à la fonction distance, au chapitre 3.

La différence remarquable entre les deux catégories se trouve dans la structure de voisinage utilisée: il n'y a aucune restriction lorsqu'il s'agit de créer une famille de points pour la valuation d'une fonction (élément structurant de grande taille, fonction structurante, etc..), alors que la propagation n'a de sens, *selon notre avis*, qu'avec des graphes de connexité de taille proche de l'unité.

Bien que les structures définissant le voisinage soient différentes, la formulation algorithmique des parcours d'ensemble et sous-ensembles, c'est à dire images et voisinages est totalement similaire (cf. §2.2.2). L'approche de *conteneur* dont bénéficie la structure de voisinage uniformise les moyens informatiques de son parcours. La différence avec l'itérateur d'image se trouve uniquement dans le centrage du voisinage préalablement à son parcours. La même approche n'est par ailleurs pas restreinte aux structures d'image, et est utilisée pour parcourir par exemple des graphes.

2.3.1.3. Ordre

Certains algorithmes font intervenir une relation d'ordre. Nous avons vu que les relations d'ordre ne sont ni plus ni moins que des fonctions booléennes d'arité 2. Deux notions sont importantes dans la manipulation des ordres par les algorithmes: la dualité et l'équivalence. Il est important ici de préciser comment il est possible pour un algorithme de déduire ces deux notions uniquement à partir de la donnée d'un ordre.

Dualité La notion de dualité est essentielle dans la plupart des algorithmes de Morphologie Mathématique. Elle est très simple à utiliser puisque l'ordre dual $\succ_{\mathcal{R}}$ d'une relation $\prec_{\mathcal{R}}$ est obtenu simplement par inversion des éléments en entrée de $\prec_{\mathcal{R}}$. Dans l'optique de la méta-programmation, il est intéressant de donner la possibilité à une relation $\prec_{\mathcal{R}}$ d'informer sur sa relation duale. Cela se fait simplement grâce aux champs de type: une relation d'ordre contient le champ « dual » informant la structure de relation d'ordre duale.

```
1  template <class T, bool is_dual = false>
2          struct ordering {
3  public:
4          typedef ordering<T, true> dual;
5          bool operator()(const T& l, const T &r) const { /* implémentation de l'ordre*/ }
6  };
```

L'opérateur dual ne possède pas d'implémentation et se contente uniquement d'inverser l'ordre des éléments à comparer.

```
1  template <class T>
2          struct ordering<T, true> :  public ordering<T, false> {
3  public:
4          typedef ordering<T, false> dual;
5          bool operator()(const T& l, const T &r) const { return dual::operator()(r, l );  }
6  };
```

Seule l'implémentation de l'ordre $\prec_{\mathcal{R}}$ existe, puisque l'ordre dual est directement déduit de ce dernier. Bien entendu plusieurs possibilités existent, nous avons trouvé la déduction de dualité proposée assez facile à manier. Lorsqu'une instance « order » d'un type d'ordre « Order » est fournie à un algorithme, ce dernier peut construire si besoin une instance « dual_order » d'ordre dual simplement de la manière suivante:

```
1  typename Order::dual dual_order(order);
```

Nous avons besoin de manipuler des instances d'ordre. Si nous prenons en effet le cas des ordres lexicographiques, ils contiennent deux vecteurs de même taille: la permutation et l'ordre sur chaque espace. L'ordre dual doit préserver ces deux données.

Ordre strict et équivalence Dans certain cas, nous disposons uniquement d'un ordre strict en entrée d'un algorithme, et nous devons en déduire une équivalence entre deux éléments. Si l'ordre $\prec_{\mathcal{R}}$ est effectivement strict, l'équivalence que nous qualifions de faible [(xvii)] est obtenue selon :

$$\neg(a \prec_{\mathcal{R}} b) \wedge \neg(b \prec_{\mathcal{R}} a) \Leftrightarrow a \equiv_{\mathcal{R}} b$$

Ce type d'équivalence implique souvent un coût supplémentaire car il faut faire deux opérations au lieu d'une seule. Encore une fois, nous contournons ce problème en enrichissant légèrement l'ordre strict par un champ de type pointant sur la structure utilisée pour l'équivalence. Par défaut, l'équivalence est déduite de manière faible selon la relation ci-dessus. Dans certain cas et pour les types connus, la structure d'ordre informe sur la relation d'équivalence qu'il est préférable d'utiliser.

2.3.2. Squelette d'algorithme

Les algorithmes complexes sont souvent décomposés en plusieurs étapes: par exemple une initialisation créant des sous-ensembles, suivie d'une étape de recherche de nouveaux points (propagation) etc.. De nombreux algorithmes proposés dans la littérature peuvent toutefois être considérés comme des *variantes* d'une algorithme original, la variation étant par exemple dans la manière de rechercher de nouveaux points, basée sur une métrique différente.

2.3.2.1. Exemple d'introduction

Afin de clarifier le contenu, prenons un exemple simple, la *labellisation*. Le but est d'identifier chaque composante connexe d'une image $\mathcal{I}$. Le procédé d'identification est généralement une nouvelle image $\mathcal{O}$ dans laquelle chaque composante connexe possède une valeur unique, généralement un nombre entier, permettant ainsi d'identifier chaque point de $\mathcal{I}$ à sa composante connexe. Un algorithme à base de lacets, dont nous nous servirons, a été proposé dans [Sch89].

Commençons par la labellisation pour les images binaires: il y a ici deux *classes* de point: le fond et la forme. Deux points de $\mathcal{I}$ font partie de la même composante connexe si les deux conditions suivantes sont réunies:

[(xvii)] déduite de deux relations « non-vraies »

1. les deux points sont voisins
2. les deux points font partie de la même classe, fond ou forme

Continuons notre exemple sur des images à teinte de gris (par exemple dans $\mathbb{N}_{256}$) avec la labellisation dite en « zones plates » [CSS+97, SS95]; la condition caractéristique est la suivante: deux points *voisins* appartiennent à la même composante connexe (ici, zone plate) s'ils ont même teinte de gris. De manière analogue, la labellisation en zones λ-plates définit la composante connexe par un assouplissement de l'égalité stricte du cadre de la zone plate: deux points *voisins* appartiennent à la même composante connexe si leur teinte de gris ne diffère par de plus de λ, λ étant un paramètre de la labellisation.

Plus généralement, nous observons que la labellisation est une transformation d'une image $\mathcal{I}$ en classes d'équivalence. Chaque classe d'équivalence définit une composante connexe. Ce qui est caractéristique à toute labellisation est la connexité (de voisinage) de deux points qui, soumise à une équivalence supplémentaire, donne lieu à la relation d'équivalence caractéristique suivante:

$$\cong_{\mathcal{N},\mathcal{R}}: \left\{ \begin{array}{lcl} \mathbf{E}^2 & \rightarrow & \{0,1\} \\ (x,y) & \mapsto & (x \cong_{\mathcal{N}} y) \wedge (\mathcal{I}(x) \cong_{\mathcal{R}} \mathcal{I}(y)) \end{array} \right.$$

avec

$$\cong_{\mathcal{N}}: \left\{ \begin{array}{lcl} \mathbf{E}^2 & \rightarrow & \{0,1\} \\ (x,y) & \mapsto & (x \in \mathcal{N}_y) \wedge (y \in \mathcal{N}_x) \end{array} \right.$$

Une image labélisée en composantes connexes est donc l'ensemble quotient [(xviii)] de l'image de départ selon la relation d'équivalence $\cong_{\mathcal{N},\mathcal{R}}$. Comme nous l'avons exprimé, la relation d'équivalence $\cong_{\mathcal{N}}$ - connexité des éléments de la composante connexe - est caractéristique du procédé de labellisation, et donc aucune labellisation ne peut s'y soustraire. Par contre, la relation d'équivalence $\cong_{\mathcal{R}}$ est dépendante de l'application visée. Ainsi, la labellisation en zones plates est liée à la relation:

$$\cong_{flatzones}: \left\{ \begin{array}{lcl} \mathbf{E}^2 & \rightarrow & \{0,1\} \\ (x,y) & \mapsto & (\mathcal{I}(x) = \mathcal{I}(y)) \end{array} \right.$$

au sens de l'égalité stricte. La labellisation en λ-zone plate est liée à la relation:

$$\cong_{\lambda-flatzones}: \left\{ \begin{array}{lcl} \mathbf{E}^2 & \rightarrow & \{0,1\} \\ (x,y) & \mapsto & (|\mathcal{I}(x) - \mathcal{I}(y)| \leq \lambda) \end{array} \right.$$

ou plus généralement sur l'espace $\mathbf{F}$ normé par $\cdot \mapsto \|\cdot\|_{\mathbf{F}}$:

$$\cong_{\lambda-flatzones,\|\cdot\|_{\mathbf{F}}}: \left\{ \begin{array}{lcl} \mathbf{E}^2 & \rightarrow & \{0,1\} \\ (x,y) & \mapsto & (\|\mathcal{I}(x) - \mathcal{I}(y)\|_{\mathbf{F}} \leq \lambda) \end{array} \right.$$

Étant donné le fait que la labellisation est l'ensemble quotient d'une relation d'équivalence, et par transitivité même de cette relation, le processus n'est pas dépendant de l'ordre de découverte des points et le résultat est unique. Par ailleurs, nous remarquons que cette définition n'est pas restreinte aux seules images binaires ou scalaires; en effet, la notion de norme ou de distance existe, par exemple, dans de nombreux espaces couleurs [(xix)].

Remarque nous ne discutons ici que de la manière de construire les composantes connexes du processus de labellisation. Il est bien sûr possible de créer également des variations de l'algorithme sur la fonction utilisée pour la valuation des composantes connexes découvertes: par exemple la valuation par la couleur moyenne d'une image de référence sur l'ensemble des points de la composante connexe. Par ailleurs la donnée sortie par l'algorithme n'est pas nécessairement une image: elle peut être une information quelconque sur les composantes connexes (liste de points, liste de boites englobantes, graphe de connexité, etc.).

[(xviii)] cf. définition 2.9 page 26

[(xix)] nous discuterons des distances couleurs dans le chapitre 4

2.3.2.2. Squelettes d'algorithme

L'exemple des labellisations est assez éloquent. Le nombre de combinaisons possibles résultant de l'algorithme original augmente exponentiellement par rapport aux diverses variations possibles. Malgré cela, nous voyons que la labellisation est constituée d'un squelette algorithmique fixe, sur lequel se greffent des structures informatiques de variation. L'algorithme squelette est un patron algorithmique, non utilisable en tant que tel. L'association avec ces structures de variation en font ensuite un algorithme « *concret* ».

Une labellisation générique L'algorithme 2.3 reprend l'algorithme de labellisation. Dans ce squelette, L est l'opérateur implémentant certaines étapes du processus de labellisation. Cet opérateur est appelé à certaines étapes clefs de la labellisation:

- lors de la découverte d'une nouvelle composante connexe, ligne 5: L peut préparer en interne des informations concernant la valuation de la composante connexe. Exemple: dans le cas d'une valuation par la valeur moyenne sur une image tierce, la somme est mise à zéro.
- lors de la découverte d'un nouveau point, ligne 13: L met à jour l'information concernant la composante connexe courante avec ce nouveau point.
- le test effectif de l'équivalence de deux points en terme de connexité, ligne 15: L teste la relation d'équivalence pour un voisin v du point p courant de la composante connexe. Il est également possible d'ajouter d'autres traitements: par exemple, si l'équivalence n'est pas vérifiée, un graphe d'adjacence des composantes connexes peut être mis à jour ici par la création d'une arête entre les deux composantes d'une image tierce [(xx)].
- la valuation de l'ensemble des points de la composante connexe, ligne 18: la donnée de sortie $\mathcal{D}$ est mise à jour avec les informations préalablement calculées. Pour une image de sortie, il s'agit d'une donnée calculée sur les points de la composante (identification de la composante par une valeur unique, moyenne d'une image tierce, etc.). La donnée $\mathcal{D}$ peut être autre chose qu'une image: une liste des boîtes englobantes des composantes connexes, un graphe de connexité, etc..

Algorithme 2.3 : Squelette algorithmique de la labellisation

Data : image d'entrée $\mathcal{I}_\mathbf{E}$, opérateur de labellisation L
Result : donnée de sortie $\mathcal{D}$

```
1  I_W ← non-traité
2  L ← initialisation des données de sortie
3  forall p ∈ I_E do
4      if I_W(p) = traité then continuer
5      L ← nouvelle composante connexe
6      q1 ← {p}                                              // fifo simple
7      q2 ← ∅
8      while q1 ≠ ∅ do
9          p = premier élément de q1
10         q1 ← supprimer le premier élément
11         q2 ← ajouter {p}
12         I_W(p) = traité                                   // chaque point n'appartient qu'à une seule composante
13         L ← nouveau point (p)
14         forall v ∈ {N_p \ p} do
15             if L_≅R (I_E(p), I_E(v)) ∧ I_W(p) ≠traité then
16                 q1 ← ajouter v
17
18     L ← valuation sur (q2)
19     L ← composante suivante
```

[(xx)] Dans ce cas, l'image tierce en question est par exemple une image préalablement labélisée

Nous le voyons sur cet exemple, l'éventail des possibilités est beaucoup plus important par la démarche de patron algorithmique. Les développements ne concernent que l'opérateur L sur lequel sont délégués des traitements spécifiques. Le squelette est toujours le même et les variantes sont implémentées dans L.

Le squelette de cet algorithme est une structure informatique cliente de l'opérateur de délégation L et, de ce point de vue, ne peut fonctionner sans cet opérateur. L informe en effet la totalité des étapes clefs de la labellisation. Dans la mesure où L doit implémenter toutes les étapes clefs, la démarche est contraignante; mais malgré cela les développements sont minimaux dans la pratique, et la redondance algorithmique est réduite au maximum. De plus une modification dans le squelette impacte directement toutes les implémentations réelles de l'algorithme. Enfin, dans le cadre d'une librairie, la capitalisation des traitements algorithmiques selon cette méthode réduit considérablement le risque d'erreurs d'implémentation: les différents développeurs n'ont pas besoin de connaître exactement l'algorithme en question, le nombre de recopies est réduit, etc.

Remarque La pratique de squelette d'algorithme est essentiellement guidée par l'expérience et l'utilisation. Il est certes nécessaire d'avoir une bonne connaissance de l'algorithme initial pour la mettre en pratique, mais également de comprendre ce qui est irréductible. Nous verrons dans le chapitre 4 comment appliquer ce principe à l'algorithme de reconstructions morphologiques.

2.4 Conclusion

Ce chapitre avait pour but d'introduire l'approche générale utilisée pour la création d'une librairie *générique* de traitement d'image dédiée à la Morphologie Mathématique. Quelques mécanismes de méta-programmation ont été explicités. Nous avons ensuite présenté les différentes notions et outils manipulés lors de l'écriture algorithmique, et nous avons détaillé leurs équivalents informatiques. La notion d'itérateur nous a par ailleurs fourni un moyen *universel* d'accès aux points des images (ou à d'autres structures, comme les graphes) et de ses sous-ensembles. Nous avons utilisé la méta-programmation pour la définition de ces outils, ce qui nous a permis d'obtenir une base logicielle extrêmement flexible. Nous avons également montré la puissance des méthodes de *spécialisation*, permettant de créer des cas de traitements spécifiques, et mariant judicieusement les termes *généralisation* et *optimisation* au sein d'une même librairie généraliste. Enfin, nous avons montré l'impact positif de ces outils sur l'écriture algorithme.

Le champ d'application de la Morphologie Mathématique est très large. Parmi l'ensemble des possibilités offertes par notre librairie, deux aspects retiennent particulièrement notre attention: la possibilité de manipuler des images de dimension quelconque, d'une part dans l'espace des coordonnées et d'autre part l'espace image. Ces deux aspects sont par nature différents et conduisent également à des traitements et approches différents.

Le premier point suit l'évolution de techniques particulières d'acquisition, typiquement les images médicales tri-dimensionnelles avec une dimension temporelle, mais ouvre également la Morphologie Mathématique sur des domaines d'applications très peu étudiés jusqu'à maintenant. Parmi ces applications nous avons l'apprentissage et la classification.

Le deuxième point suit l'évolution des capteurs en terme de mixité et de richesse d'information. La couleur et sa démocratisation (multimédia, vidéo-surveillance, photographie numérique, etc.) est un exemple immédiat. L'utilisation de microscopes dont les capteurs réagissent à des notions physiques mixtes, et produisant des images multispectrales de grande taille est un autre exemple.

La suite de ce manuscrit s'inscrit dans la continuité des éléments développés dans ce chapitre. Nous allons revoir un certain nombre d'algorithmes classiques du point de vue *générique*. Nous disposons en effet des outils informatiques de base pour exprimer les algorithmes existants de Morphologie Mathématique dans ce contexte. Dans un soucis d'exactitude, nous reverrons un certain nombre d'entre-eux de manière à réduire leurs biais et à généraliser leur expression pour l'utilisation en dimension quelconque.

Ainsi nous étudierons au début du chapitre 3 un algorithme de distance exacte en n dimensions, en faisant abstraction de la fonction distance utilisée. Nous développerons également des améliorations d'algorithmes de distance morphologique nouvelle que l'on nomme « quasi-distance ». Cette distance profitera en outre du cadre de notre librairie pour sa généralisation à la couleur.

Nous verrons au chapitre 4 certains traitements concernant des images multi-valuées, où plus globalement « multispectrales ». Dans cette étude, certains outils tels que les gradients morphologiques seront définis dans ce cadre à l'aide de fonction distance. Nous verrons également certaines possibilités liées à l'utilisation d'ordres lexicographiques.

Nous verrons ensuite au chapitre 5 l'algorithme de ligne de partage des eaux, algorithme de segmentation par excellence de la Morphologie Mathématique. Nous mettrons en évidence les biais de l'algorithme original de Meyer [Mey95, Mey91] et proposerons un algorithme non biaisé. L'algorithme profitera également du cadre que nous venons du développer: il fonctionne en dimension quelconque et il est même possible d'utiliser un gradient multi-valué pour rendre l'ordre plus fin. Nous utiliserons également les voisinages pour contraindre localement le processus de propagation, ainsi que les méthodes de *POO* pour développer des algorithmes génériques utilisant des contraintes de propagation.

CHAPITRE 3

Distances

« Une année-lumière est une distance considérable, particulièrement quand elle est bissextile. »
Mots et grumots, Marc Escayrol

« La distance rend toute chose infiniment plus précieuse. »
2001 - l'odyssey de l'espace - Arthur Charles Clarke

Ce chapitre se décompose en deux parties distinctes, structurées autour des fonctions de distance sur des images. La première partie, aux sections §3.1 et §3.2, discutera du calcul exact de fonction de distance sur des grilles régulière. La deuxième partie discutera du calcul de distances morphologiques particulières dites « *quasi-distances* ».

Nous avons bien souligné dans le chapitre 2 précédent la possibilité de travailler sur des images en n *dimension*. Nous avons également insisté sur la *généricité* de la programmation, laissant un champ beaucoup plus vaste à la manipulation des algorithmes.

Nous allons illustrer ces propos par l'étude d'un algorithme de fonction de distance *générique*. La généricité de l'algorithme se manifestera à deux niveaux, et l'algorithme sera indépendant de:

- la fonction de distance utilisée pour le calcul sur la grille.
- la dimension des images, supposée ici *quelconque*.

L'objectif ne sera pas de proposer un algorithme plus performant que d'autres connus dans la littérature, mais de proposer une transformée **exacte**, $\mathbf{nD}$ et laissant beaucoup de liberté à l'utilisateur sur le choix de la fonction distance.

Nous continuerons ensuite sur les distances selon un tout autre point de vue. Nous verrons en partie 3.3 les quasi-distances qui sont des fonctions distances *morphologiques* nouvellement introduites par Beucher [Beu03, Beu05], et qui fonctionnent sur des images à niveau de gris. Après une description de la fonction distance, nous proposerons un algorithme de transformation rapide des quasi-distances. Nous quantifierons l'amélioration qu'apporte l'algorithme proposé, et élargirons son champs d'utilisation aux images couleur grâce notamment au cadre décrit au chapitre 2 .

3.1 Les transformations en distance

Pour cela, nous allons dans un premier temps introduire les notions liées aux fonctions distance et qui nous serviront dans la suite de l'étude. Nous verrons ensuite comment tirer profit des nouvelles

structures de données introduites en partie 2 pour mettre au point un algorithme de distance euclidienne exacte capable de fonctionner sur des images de dimension quelconque (section 3.2).

3.1.1. Définition

Définition 3.1 (Distance)
Soit $\mathbf{E}$ *un espace. Une fonction* $d : \mathbf{E}^2 \mapsto \mathbb{R}_+$ *est une* ***distance*** *si pour tout triplet* $(x, y, z) \in \mathbf{E}^3$ *les conditions suivantes sont satisfaites :*

$$\begin{array}{ll} d(x,y) = d(y,x) & \textit{symétrie} \\ d(x,y) = 0 \Leftrightarrow x = y & \textit{séparation} \\ d(x,z) \leq d(x,y) + d(y,z) & \textit{inégalité triangulaire} \end{array} \tag{3.1}$$

Soit $\mathbf{X} \subset \mathbf{E}$ *un sous-ensemble de* $\mathbf{E}$ *et* $x \in \mathbf{E}$ *un point. La distance entre* x *et* $\mathbf{X}$ *selon* d *est définie par:*

$$d(x, \mathbf{X}) = \inf \{d(x, b), b \in \mathbf{X}\} \tag{3.2}$$

Soit Ξ *l'ensemble des translations sur* $\mathbf{E}$*. La distance* d *est dite* ***invariante par translation*** *si:*

$$\forall \tau \in \Xi, \forall (x, y) \in \mathbf{E}^2, d(\tau(x), \tau(y)) = d(x, y)$$

Si $\mathbf{E}$ est un espace vectoriel muni d'une norme $\cdot \mapsto \|\cdot\|$, la distance d telle que $d : (x, y) \mapsto \|x - y\|$ est dite distance **induite** par la norme, et nécessairement invariante par translation. Nous nous restreindrons à ce cadre dans la suite, les termes distance et norme deviennent alors équivalent.

Proposition 3.1
Soit $\mathbf{E}$ *un espace vectoriel de dimension finie. Nous notons* B_d *la boule unité pour la distance* d*, ou simplement :*

$$B_d = \{p \in \mathbf{E}, d(p, 0) \leq 1\}$$

Soit $x \in \mathbf{E}$*, nous notons* $B_d(x)$ *la même boule unité centrée en* x*:* $B_d(x) = \{p \in \mathbf{E}, d(p, x) \leq 1\}$*. La relation suivante est vérifiée:*

$$d_i \leq d_j \Rightarrow \forall x \in \mathbf{E}, B_{d_i}(x) \supset B_{d_j}(x)$$

Soit $x \in \mathbf{E}, p \in B_{d_j}(x)$. Nous avons $d_j(p, x) \leq 1$. Par hypothèse $d_i(p, x) \leq d_j(p, x)$. D'où $d_i(p, x) \leq 1$ et donc $p \in B_{d_i}(x)$. ■

Proposition 3.2
Soit $\mathbf{E}$ *un espace vectoriel de dimension finie,* $\mathbf{X} \subset \mathbf{E}$ *un sous-ensemble de* $\mathbf{E}$*. Soit* $\iota \in \mathbb{R}_+$*, nous notons* $B_d(\mathbf{X}, \iota)$ *la boule de taille* ι *autour de* $\mathbf{X}$ *pour la distance* d*:*

$$B_d(\mathbf{X}, \iota) = \{p \in \mathbf{E}, d(p, \mathbf{X}) \leq \iota\}$$

La relation suivante est vérifiée, pour $\iota \in \mathbb{R}$*:*

$$d_i \leq d_j \Rightarrow \forall \iota \in \mathbb{R}_+, \mathbf{X} \subset \mathbf{E}, B_{d_i}(\mathbf{X}, \iota) \supset B_{d_j}(\mathbf{X}, \iota)$$

Démonstration analogue à la précédente. ■

Définition 3.2 (Norme ℓ^p)
Soit $\mathbf{E}$ *un espace vectoriel de dimension finie* n, $x = (x_1, x_2 \ldots, x_n)$ *un vecteur de* $\mathbf{E}$, *et* $p \in \mathbb{N}^*$. *La norme* ℓ^p, *que l'on notera* $\cdot \mapsto \|\cdot\|_p$, *est définie de la manière suivante:*

$$\|\cdot\|_p : \begin{cases} \mathbf{E}^2 & \to & \mathbb{R}_+ \\ x & \mapsto & \left(\sum_{i \in \mathbb{N}_n^*} |x_i|^p\right)^{1/p} \end{cases}$$

Ainsi, ℓ^2 est la norme euclidienne, ℓ^1 la norme de *Manhattan*, et la norme infinie ℓ^∞ est définie par:

$$\|\cdot\|_\infty : x \mapsto \bigvee_{i \in \mathbb{N}_n^*} |x_i|$$

De manière similaire, nous noterons d_p la distance induite par la norme ℓ^p. La définition de distance n'est bien sûr par restreinte aux définitions des ℓ^p. Enfin nous noterons simplement B_k pour la boule unité donnée par la norme ℓ^k.

Lemme 3.1 (Encadrement de d_p)
Soit $p \geq 1$ *un entier,* $\mathbf{E}$ *un espace vectoriel de dimension* n. *La relation suivante est vérifiée sur* $\mathbf{E}$*:*

$$d_\infty \leq d_p \leq d_1 \leq n^{\frac{1}{p}} \cdot d_\infty$$

En effet, à p fixé, $\forall (x, y) \in \mathbf{E}^2$, en notant $x = (x_1, x_2, \ldots, x_n)$ et $y = (y_1, y_2, \ldots, y_n)$,

$$\bigvee_i |x_i - y_i| = \left[\left(\bigvee_i |x_i - y_i|\right)^p\right]^{1/p}$$

or $\cdot \mapsto |\cdot|^p$ et $\cdot \mapsto |\cdot|^{1/p}$ sont des applications croissantes, d'où:

$$\left(\bigvee_i |x_i - y_i|\right)^p = \bigvee_i |x_i - y_i|^p$$

et

$$\begin{array}{ll} \bigvee_i |x_i - y_i|^p & \leq \sum_i |x_i - y_i|^p \\ & \leq n * \bigvee_i |x_i - y_i|^p \\ & \leq n * \left(\bigvee_i |x_i - y_i|\right)^p \end{array}$$

Par application de $\cdot \mapsto |\cdot|^{1/p}$:

$$\bigvee_{i \in \mathbb{N}_n^*} |x_i - y_i| \leq \left(\sum_{i \in \mathbb{N}_n^*} |x_i - y_i|^p\right)^{1/p} \leq n^{\frac{1}{p}} * \left(\bigvee_{i \in \mathbb{N}_n^*} |x_i - y_i|\right)$$

Soit $f_p : x \mapsto \frac{(1+x)^p}{1+x^p}$ sur $\mathbb{R}_+$. Nous avons $f \geq 1$ pour $p \geq 1$, et $f \leq 1$ pour $p < 1$. Or $\forall (a, b) \in \mathbb{R}_+^* \times \mathbb{R}_+^*$, $\frac{(a+b)^p}{a^p + b^p} = f(\frac{a}{b}) = f(\frac{b}{a})$. Par application de $\cdot \mapsto |\cdot|^{1/p}$,

$$\forall (a, b) \in \mathbb{R}_+^* \times \mathbb{R}_+^*, \begin{cases} a + b \geq (a^p + b^p)^{\frac{1}{p}} & p \geq 1 \\ a + b \leq (a^p + b^p)^{\frac{1}{p}} & p \leq 1 \end{cases}$$

Le résultat final se déduit ensuite par croissance de $\cdot \mapsto |\cdot|^{1/p}$ et récurrence sur n.
■

La figure 3.1 donne un aperçu de quelques boules unitées pour des valeurs de p supérieures ou égales à 1 (toutes les boules sont convexes).

Ceci signifie que pour une taille ι donnée, et selon l'encadrement des fonctions distance, la boule $B_{d_\infty}(\mathbf{X}, \iota)$ contiendra toutes les autres boules $B_{d_p}(\mathbf{X}, \iota)$, $p > 1$. Nous allons nous servir de cette propriété dans l'élaboration d'un algorithme pour la *transformée en distance*, décrite ci-après.

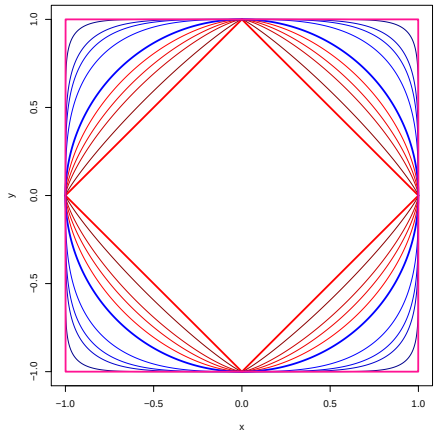


Fig. 3.1.: Les boules unités pour les distances d_p, avec $p \in \{1, 1.15, 1.35, 1.5, 1.7, 2, 3, 4, 5, 10, +\infty\}$. En rouge, les boules pour $p < 2$, en bleu les boules pour $p \geq 2$.

Définition 3.3 (Voronoï - cellule, noyau)
Soit $\mathbf{E}$ *un espace muni de la distance* d*, et soit* $\mathbf{X} = \{f_i\}, i = 1, \ldots, N$ *un ensemble fini de fermés de* $\mathbf{E}$*. Le diagramme de* ***Voronoï*** *de* $\mathbf{X}$ *est la transformation qui à chaque élément* f_i *de* $\mathbf{X}$ *associe un sous-ensemble* $\mathcal{V}(f_i)$ *de* $\mathbf{E}$*, construit de la manière suivante:*

$$\mathcal{V}(f_i) = \{x \in \mathbf{E}, d(x, f_i) < d(x, f_j), j \neq i\}$$

Les $\mathcal{V}(f_i), i \in N$ *sont appelées les* ***cellules*** *de Voronoï, dont les* $f_i, i \in N$ *sont les* ***noyaux****.*

Chaque cellule est donc un sous-ensemble **ouvert** de $\mathbf{E}$ dont les points sont les plus proches de f_i que de n'importe quel autre point de l'ensemble $\mathbf{X}$. Nous rencontrons également dans la littérature (cf. [Aur91]) une définition selon laquelle les cellules sont des fermés (il suffit pour cela de transformer l'inégalité stricte en inégalité large). Notons toutefois que lorsqu'il existe un fermé f_i non convexe, la cellule correspondante ainsi que les cellules directement adjacentes (connexes) ne sont pas convexes (voir par exemple [AS95]). Par contre, si tous les f_i sont réduis à des singletons, et que la distance considérée est convexe, alors les cellules de Voronoï sont nécessairement convexes également.

Nous nous servirons de la notion de cellule de Voronoï dans cette section, ainsi que dans le chapitre 5 traitant des inondations. Nous ne considérerons pas de noyaux autre que les singletons: en effet, le principal problème du calcul de la fonction distance dans un espace discret réside précisément dans le passage du continu au discret. Le Voronoï des singletons dans l'espace discret va servir de support aux explications et illustrations.

3.1.2. Transformée en distance

Formulation dans le domaine continu

Considérons $\mathcal{I}$ comme étant une image à valeurs binaires. Pour reprendre les notations précédentes, l'ensemble image $\mathbf{F}$ est à valeurs dans $\{0, 1\}$ et l'ensemble support $\mathbf{E}$ est quelconque. $\mathcal{I}$ peut donc être considéré comme l'union de deux sous-ensembles d'intersection nulle: le fond $\mathcal{B}$ et la forme $\mathcal{F}$ (pour respectivement *background* et *foreground*) :

$$\mathbf{E} = \mathcal{B} \cup \mathcal{F}$$

Selon cette notation, nous avons:

$$\begin{cases} \mathcal{B} & = & \mathcal{I}^{-1}(0) \\ \mathcal{F} & = & \mathcal{I}^{-1}(1) \end{cases}$$

La transformation en distance de $\mathcal{I}$ est une image $\mathcal{D}$ de même support et dimension que $\mathcal{I}$: $\mathbf{E}_\mathcal{I} = \mathbf{E}_\mathcal{D} = \mathbf{E}$, qui à chaque point de $\mathbf{E}$ associe sa distance à $\mathcal{F}$. En d'autres termes, et en notant d une distance sur $\mathbf{E}$:

$$\mathcal{D} : \left\{ \begin{array}{ccc} \mathbf{E} & \rightarrow & \mathbb{R}_+ \\ p & \mapsto & d(p, \mathcal{F}) = \inf \{d(p, q), q \in \mathcal{F}\} \end{array} \right. \tag{3.3}$$

Le problème est exactement le même lorsque l'on souhaite obtenir la distance à $\mathcal{B}$, au lieu de la distance à $\mathcal{F}$. Nous considérerons donc dans la suite uniquement les points de $\mathcal{F}$.

Nous voyons bien que nous obtiendrons une transformée différente selon la distance choisie sur $\mathbf{E}$. La méthode naïve de calcul de $\mathcal{D}$ consiste à calculer pour tous les points de l'espace $\mathbf{E}$ la distance aux points de $\mathcal{F}$, et d'assigner aux points la distance minimale. Ce calcul est bien sûr extrêmement coûteux (en $O(\sharp\mathcal{F} \times \sharp\mathcal{B})$), et nous allons voir des méthodes plus astucieuses, en essayant autant que possible de ne pas nous restreindre à une quelconque dimension de $\mathbf{E}$ ni à une fonction distance particulière.

Discrétisation du problème

Lorsque nous travaillons sur des images numériques, nous avons une trame régulière identifiée à $\mathbb{Z}^n$. Cette trame n'est qu'une approximation de l'ensemble $\mathcal{F}$ considéré précédemment. Cependant cette approximation est également prise pour référence pour le calcul de la distance exacte : puisque nous n'avons pas accès à $\mathcal{F}$ mais à une restriction $\mathcal{F}_\mathbb{Z} = \mathcal{F} \cap \mathbb{Z}^n$ sur $\mathbb{Z}^n$, nous considérerons dans notre calcul $\mathcal{F}_\mathbb{Z}$ comme ensemble de référence.

La structure particulière de l'espace discret $\mathbb{Z}^n$ introduit deux points remarquables:

1. Les développements sur les distances ne peuvent reposer sur des hypothèses de continuité de l'espace de travail. Ceci aura pour conséquence l'apparition de certains problème, notamment celui de la déconnexion des cellules de Voronoï des sources dans l'espace discret [(i)]. Nous reviendrons sur ce point dans la partie suivante.
2. Chaque élément de $\mathbb{Z}^n$, et donc a fortiori de $\mathcal{F}_\mathbb{Z}$ est un singleton, donc fermé et compact.

La deuxième remarque nous amènera à considérer dans la suite un sous-ensemble de points de l'espace discret comme les sources à partir desquelles nous souhaitons calculer la transformée en distance. Ces points seront donc dans la transformée à une distance nulle [(ii)].

Enfin, nous parlerons de *distance exacte* si, en chaque point p de la trame discrète $\mathbb{Z}^n$, nous avons $\mathcal{D}(p) = d(p, \mathcal{F}_\mathbb{Z}$.

3.1.3. État de l'art et considérations algorithmiques

La littérature propose un grand nombre d'algorithmes pour la transformée en distance. La plupart d'entre eux se concentrent sur la rapidité de la transformation et se restreignent au cadre $2D$. Les méthodes se scindent principalement en deux classes : celles approchant la distance (méthodes biaisées) et celles fournissant une distance exacte. Nous ne nous intéressons qu'aux méthodes exactes.

Une revue de certaines méthodes de calcul est disponible dans [Cui99]. Une classe de ces méthodes utilise la connaissance *a priori* sur les voisinages utilisés lors de la propagation. La déduction de la distance sur un nouveau point est faite selon la configuration de ce voisinage, sur les points connus. C'est ce qui s'appelle le « chamfering ». Cette méthode ne garantit pas l'exactitude de la distance, mais l'erreur peut être bornée. Le principal inconvénient de la déduction locale provient de la propagation des erreurs. À partir de cette remarque, des méthodes dites « vectorielles », sous l'impulsion initiatique de [Dan80], propagent les coordonnées des points sources sur les points non connus, et permettent ainsi d'obtenir l'exactitude. C'est ce que nous allons étudier à présent.

[(i)] Une propriété des cellules de Voronoï est d'être connexes par arcs, notion qui perd son sens dans un espace discret.

[(ii)] Il est possible de modifier cette hypothèse comme nous le verrons par la suite. En effet, la discrétisation de $\mathcal{F}$ n'est pas toujours possible - si $\mathcal{F}$ décrit une droite parallèle à un axe de la trame et ayant une intersection nulle avec cette dernière ($\mathcal{F} \cap \mathbb{Z}^n = \varnothing$)

Algorithmes à propagation: L'approche vectorielle consiste à propager les coordonnées des points de source, c'est à dire les coordonnées des points de $\mathcal{F}$ ayant produit la distance minimale à $\mathcal{F}$. La propagation génère une suite d'enveloppes (ou fronts/lacets dans le cas $2D$) d'iso-distance par rapport à $\mathcal{F}$ (cf. figure 3.2). On qualifie cette approche de *vectorielle* puisque chaque point de l'enveloppe contient sa *source*, qui est une donnée vectorielle.

Dans l'algorithme initial de Danielsson [Dan80], chaque nouvelle enveloppe est construite à partir de la précédente, par simple voisinage. Pour cela, pour chaque point de la nouvelle enveloppe, on récupère toutes les sources des points voisins et de l'enveloppe précédente. Pour chacune de ces sources, on ne garde que celle produisant la distance minimale. L'algorithme produit donc une distance *exacte* pour la plupart des points, mais des biais existent.

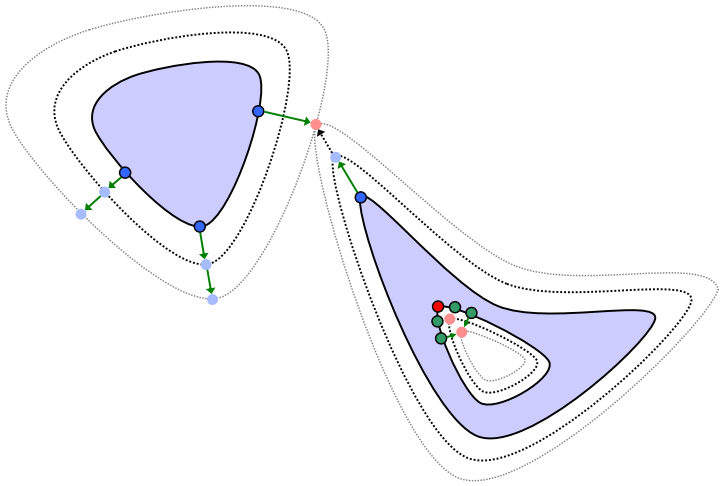
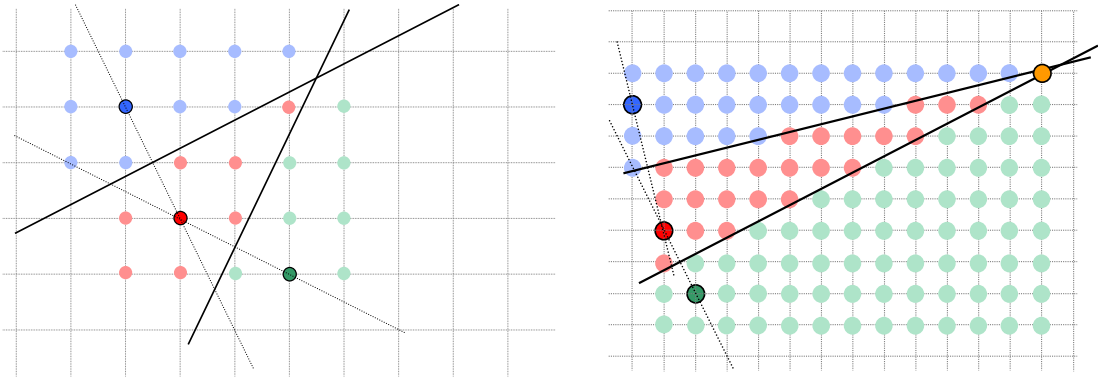


Fig. 3.2.: Illustration de la propagation pour le calcul de la fonction distance. L'ensemble d'intérêt $\mathcal{F}$ est représenté en bleu. Les lignes pointillées correspondent aux enveloppes de propagation lors des deux premières étapes. Les distances aux points bleu clair sont déduites d'une source unique (bleu plein), généralement le cas sur les convexités de $\mathcal{F}$. Les points à l'intérieur de concavités ou à la rencontre de deux Voronoï (en rouge clair) possèdent plusieurs sources. La déduction des distances de voisins de rouge nécessite la prise en compte de toutes les sources voisines de rouge.

Biais de la propagation: Le premier type de biais provient des sources des points à l'intérieur des concavités, qui ne sont pas uniques (cf. figure 3.2). Ceci conduit Vincent [Vin91] à d'abord considérer les points à l'intérieur des concavités, puis à leur appliquer une propagation différente des points issus des convexités. L'approche est efficace en $2D$ puisqu'énumérer les concavités par rapport au graphe de connexité est assez facile. Cependant son extension en nD est assez délicate.

Le second type de biais provient du passage du cadre continu au cadre discret. En effet, la discrétisation de $\mathbf{E}$ ne garantit plus la connexité de la cellule de Voronoï d'un point source et selon un graphe de connexité précis, et l'algorithme de Danielsson échoue. Ce problème est illustré sur la figure 3.3.

La restriction au cadre $2D$ permet de prendre en compte les éventuels biais connus de la fonction de distance à base de propagation. Ainsi, dans [Cui99], l'auteur effectue une première propagation, biaisée, mais déduit des biais une condition particulière sur les configurations du voisinage. Cela lui permet de détecter les zones de biais et de les corriger rapidement. L'inconvénient d'une telle approche est qu'elle est fortement dépendante à la fois de la distance utilisée, et du cadre bidimensionnel. Elle n'est d'ailleurs pas extensible de manière simple à des dimensions supérieures [Cui99, page 111]. Dans [Cui97], l'auteur utilise une contrainte géométrique pour la détection des points isolés des cellules de Voronoï, mais encore une fois l'application d'une telle méthode, géométrique, à des dimensions



(a) Biais en trame 4-connexe (b) Biais en trame 8-connexe

Fig. 3.3.: Déconnexion du diagramme de Voronoï en trame discrète. En rouge, vert et bleu clair respectivement les zones de Voronoï des pixels rouge, vert et bleu pleins. Le passage en trame discrète provoque la déconnexion du Voronoï de rouge selon le graphe 4-connexe (a). (b) : le point orange $(0,0)$ appartient normalement au Voronoï du point rouge plein $(12,5)$, mais est déconnecté du Voronoï de rouge à cause des Voronoï de bleu $(13,1)$ et vert $(11,7)$.

supérieures est très délicate.

Une autre approche empruntée par Vincent [Vin91] consiste à ne pas empêcher la propagation des points sources vérifiant une condition lipchitzienne par rapport à la distance exacte en un point. Nous voyons que d'une part l'algorithme recherche les concavités pour n'avoir à propager qu'une unique source, et d'autre part toutes les sources vérifiant une contrainte lipchitzienne doivent être propagés. Faisons donc fi de la recherche des concavités puisque le problème se résume finalement à la propagation d'un ensemble de points sources et non d'une source unique.

Nous allons baser nos développements sur la propriété lipchitzienne, que nous considérons beaucoup plus astucieuse que l'approche géométrique. Nous allons justifier dans un premier temps cette approche en démontrant l'équivalence du processus de propagation à la dilatation par un élément structurant précis. Nous verrons ensuite une contrainte lipchitzienne liée à la dimension de l'espace. L'originalité vient du fait que nous nous plaçons dans un cadre général, sans nous contraindre ni par la distance utilisée, ni par la dimension de l'espace image.

Nous avons besoin du lemme suivant, proposé par Matheron et que nous reprenons ci-après [(iii)].

Lemme 3.2 (Matheron [Mat69, page 19])
Soit E un espace, d une fonction distance sur E, et B_λ la boule fermée de taille λ selon la métrique d. Si B est convexe, nous avons $B_\lambda \oplus B_\mu = B_{\lambda+\mu}$.

Nous avons B convexe $\Rightarrow B_{\lambda+\mu}$ convexe .

$$x \in B_{\lambda+\mu} \Rightarrow \frac{\lambda}{\lambda+\mu} \cdot x + \frac{\mu}{\lambda+\mu} \cdot x \in B_{\lambda+\mu}$$

Or

$$d(\frac{\lambda}{\lambda+\mu} \cdot x, 0) = \frac{\lambda}{\lambda+\mu} d(x,0) \leq \frac{\lambda}{\lambda+\mu} \cdot (\lambda+\mu)$$

D'où $\frac{\lambda}{\lambda+\mu} \cdot x \in B_\lambda$. De la même manière $\frac{\mu}{\lambda+\mu} \cdot x \in B_\mu$.

Soit donc

$$\begin{array}{rcl} x \in B_{\lambda+\mu} & \Rightarrow & \exists (a,b) \in B_\lambda \times B_\mu \,/\, x = a+b \quad (\text{ avec } a = \frac{\lambda}{\lambda+\mu} \cdot x \quad \text{et} \quad b = \frac{\mu}{\lambda+\mu} \cdot x) \\ & \Rightarrow & x \in B_\lambda \oplus B_\mu \end{array}$$

[(iii)] uniquement pour des raisons pratiques, puisque la référence [Mat69] n'est pas évidente à trouver.

L'inclusion réciproque est immédiate.
■

Proposition 3.3
Soit $\mathbf{E}$ *un espace vectoriel de dimension finie,* $\mathbf{X} \subset \mathbf{E}$ *un sous-ensemble de* $\mathbf{E}$*,* B *une boule « unité » pour la distance* d *et de manière à ce que* B *soit convexe. Soit* $k \in \mathbb{N}^*$ *et* $\delta_B^{(k)}(\mathbf{X})$ *la dilatation de* $\mathbf{X}$ *de taille* k *par l'ensemble structurant* B*. Nous avons:*

$$B(\mathbf{X}, k) = \delta_B^{(k)}(\mathbf{X})$$

La convexité de B nous permet d'écrire l'égalité suivante [Mat69, page 19] (lemme 3.2) [(iv)]:

$$\lambda \geq 0, \mu \geq 0, B \text{ convexe } \Rightarrow B^\lambda \oplus B^\mu = B^{\lambda+\mu}$$

Par associativité de l'addition de Minkowski,

$$\mathbf{X} \oplus B \oplus \ldots \oplus B = \mathbf{X} \oplus_{i=1}^{N} B = \mathbf{X} \oplus B^N$$

Il faut donc prouver $B(\mathbf{X}, \lambda) = \mathbf{X} \oplus B^\lambda$:

$$\begin{array}{lcll} a \in \mathbf{X} \oplus B^\lambda & \Leftrightarrow & \exists (x,b) \in \mathbf{X} \times B^\lambda \,/\, a = x + b & \text{(définition de l'addition de Minkowski)} \\ & \Leftrightarrow & \exists (x,b) \in \mathbf{X} \times B^\lambda \,/\, a - x = b & \\ & \Leftrightarrow & \exists x \in \mathbf{X} \,/\, d(a-x, 0) \leq \lambda & \text{(définition de } B^\lambda\text{)} \\ & \Leftrightarrow & \exists x \in \mathbf{X} \,/\, d(a, x) \leq \lambda & (d \text{ invariant par translation}) \\ & \Leftrightarrow & a \in B(\mathbf{X}, \lambda) & \text{(définition de } B(\mathbf{X}, \lambda)\text{)} \end{array}$$

■

Rappelons que la convexité est une propriété assurée pour les boules définies sur les distances type $d_{p,p\geq 1}$ (voir figure 3.1).

Selon cette proposition et le lemme 3.1, un encadrement de toutes les distances d_p peut être obtenu par l'intermédiaire des dilatations de l'ensemble d'intérêt. La dilatation mentionnée ici est celle donnée par un élément structurant construit comme étant la boule unité de la distance considérée. Pour la distance d_∞, il s'agira de la boule B_∞, alors que la distance euclidienne à $\mathbf{X}$ sera donnée par le dilaté de $\mathbf{X}$ selon la boule unité euclidienne B_{d_2}. Ceci justifie l'utilisation de l'approche à base de propagation: toutes les sources potentielles d'un point sont contenues dans les sources du dilaté de l'ensemble d'intérêt et en ce point. Ceci est indépendant de la dimension de l'espace d'étude et de la distance utilisée, mais n'offre pas intrinsèquement un moyen de calcul en trame discrète.

En effet, un problème se pose assez rapidement lorsque nous travaillons en trame discrète: l'approximation de B_p sur cette trame sera soit B_1 (distance Manhattan), soit B_∞ (distance norme sup). Par ailleurs, nous ne pouvons nullement déduire du résultat selon une distance d_p un résultat selon une distance $d_{p'}$. Cette remarque nous a particulièrement induit en erreur, nous n'avons tout simplement pas l'implication suivante:

$$\forall (a,b,c) \in \mathbf{E}^3, p \geq 1,\ d_p(a,b) < d_p(a,c) \nRightarrow d_{p'}(a,b) < d_{p'}(a,c)$$

Ainsi, pour les deux sources $s_1 = (16, 14, 0, 0)$ et $s_2 = (12, 2, 0, 6)$, au point $x = (16, 6, 3, 0)$, nous avons $d_\infty(s_1, x) = 8$, $d_\infty(s_2, x) = 6$. Or $d_2(s_1, x) = \sqrt{73}$ et $d_2(s_2, x) = \sqrt{77}$. Ce qui signifie que si nous construisons une suite de dilatés de $\mathbf{X}$ avec la boule B_∞, nous ne pourrons nullement déduire

[(iv)] Remarque : Selon Matheron, B doit être également compact. La compacité est une propriété supposée acquise ici par les espaces images que nous nous sommes fixés dans la partie 2.2 - espaces $\mathbb{R}^n$ ou $\mathbb{Z}^n$ fermés et bornés -. La distance n'induit que la *pré-compacité*, il faut en plus que l'espace soit complet - propriété liée à la distance - pour être compact.

une suite de dilatés de $\mathbf{X}$ selon la distance euclidienne d_2. Nous insistons sur cette remarque car nous sommes restés longtemps dans cette erreur. Nous remarquons par ailleurs sur cet exemple que les boules issues de d_∞ avancent « trop vite » par rapport à celles de d_2.

L'idée est donc d'approcher $\mathbf{X} \oplus B^\lambda$ à l'aide d'une suite $(X_n)_n$ d'ensembles obtenus par des dilatations conditionnelles: la dilatation n'est pas appliquée sur l'ensemble dans sa totalité, mais sur un sous-ensemble de points vérifiant une propriété. Les dilatations sont effectuées à l'aide d'une boule convexe quelconque, qui n'est pas nécessairement la boule unité de la distance considérée.

La propriété suivante va nous permettre de travailler en trame discrète:

Lemme 3.3
Soit $\mathbf{E}$ un espace vectoriel et $\mathbf{X} \subset \mathbf{E}$ un sous-ensemble de $\mathbf{E}$, d une distance sur $\mathbf{E}$. La fonction

$$d_X : \left\{ \begin{array}{ccc} \mathbf{E} & \rightarrow & \mathbb{R}_+ \\ x & \mapsto & d(x, \mathbf{X}) \end{array} \right.$$

est une fonction 1-lipchitzienne.

Cette propriété découle simplement de l'inégalité triangulaire sur d: $\forall (x, x') \in \mathbf{E}^2, a \in \mathbf{X}, b \in \mathbf{X} / d(x', \mathbf{X}) = d(x', b)$:

$$\begin{array}{rcll} d(x, \mathbf{X}) - d(x', \mathbf{X}) & \leq & d(x, a) - d(x', \mathbf{X}) & \forall a \in \mathbf{X} \\ & \leq & d(x, a) - d(x', b) & \forall a \in \mathbf{X} \\ & \leq & d(x, b) - d(x', b) & \\ & \leq & (d(x, x') + d(x', b)) - d(x', b) & \end{array}$$

On déduit le résultat par symétrie des rôles de x et x'.■

Soit B une boule convexe, nous supposerons pour l'instant $|d(x, \mathbf{X}) - d(x', \mathbf{X})| \leq \kappa_B$: ceci nous permettra de poser une borne supérieure à l'inégalité précédente. Nous discuterons ensuite sur le choix de κ_B. Il est important de remarquer que B n'est pas nécessairement la boule unité associée à d.

Soit $x \in \mathbf{E}$ un point, $\mathbf{X} \subset \mathbf{E}$ un sous-ensemble, et $s_x \subset \mathbf{X}$ un sous-ensemble associé à x que nous appellerons l'ensemble des « sources associées » à x. Soit

$$X_n = X_{n-1} \cup (X_{n-1}^\wedge \oplus B)$$

avec

$$X_{n-1}^\wedge = \left\{ x \in X_{n-1}, d(x, s_x) = \wedge_{a \in X_{n-1}} d(a, s_a) \right\}$$

Enfin, construisons les sources s_x de la manière suivante:

$$s_x = \bigcup \{ s_{x'} \ / \ x' \in x \oplus B \text{ et } d(x', s_{x'}) < d(x, s_x) \leq d(x', s_{x'}) + \kappa_B \} \tag{3.4}$$

La suite (X_n) n'est rien d'autre que l'expression d'une dilatation conditionnelle de l'ensemble $\mathbf{X}$ de départ: nous dilatons $\mathbf{X}$ uniquement aux lieux où la distance à $\mathbf{X}$ est minimale. Bien entendu, nous avons $X_0 = \mathbf{X}$. Pour tous les points x sélectionnés pour cette dilatation, nous construisons un sous-ensemble s_x, que nous appelons « sources », et qui présente la particularité suivante:

Proposition 3.4
Selon la description précédente et pour tout $n \in \mathbb{N}^*$, soit $x \in X_{n-1}^\wedge$ et soit s_x l'ensemble des sources associées. Nous avons $d(x, s_x) = d(x, \mathbf{X})$.

Nous pouvons commencer la démonstration par la remarque suivante : soit

$$s_{x,n} = s_{x,n-1} \cup \left\{ s_{x'} \ / \ x \in x' \oplus B \text{ et } d(x', s_{x'}) < d(x, s_{x,n-1}) \right\}$$

Soit $(u_n)_n$ la suite définie pour tout $n \in \mathbb{N}^*$ par $u_n = d(x, s_{x,n})$. Nous avons $\forall n \in \mathbb{N}^*, u_n \geq 0$ par définition de la fonction distance. Par ailleurs, $\forall n \in \mathbb{N}^*, s_{x,n} \subset s_{x,n+1} \subset \mathbf{X}$, d'où $u_{n+1} \leq u_n$. (u_n) est donc une

suite décroissante et minorée dans $\mathbb{R}$, donc convergente. Il faut à présent démontrer que la limite de (u_n) est effectivement $d(x, \mathbf{X})$.
Nous allons démontrer la proposition par récurrence sur n.

- **Conditions initiales :** $X_1 = X_0 \cup (X_0^\wedge \oplus B)$. Or $\forall x \in X_0, d(x, s_x) = 0 = d(x, \mathbf{X})$.

- **Hypothèse de récurrence :** $\forall k \in \mathbb{N}^*, k \leq n,\ x \in X_k^\wedge \Rightarrow d(x, s_x) = d(x, \mathbf{X})$.
  Prenons $x \in X_{n+1}^\wedge$ et posons $A = \{a \in x \oplus B / d(a, s_a) < d(x, s_x)\}$. L'élément B étant supposé symétrique, l'ensemble A décrit les points utilisés lors de la construction de s_x. De plus, par définition de $X_{n+1}^\wedge$, il n'existe aucun point de $\mathbf{E}$ de distance inférieure, la suite $(s_{x,n})$ mentionnée a donc atteint sa limite s_x.
  Soit F l'ensemble des noyaux dont les cellules de Voronoï correspondante ont une intersection non vide avec $x \oplus B$: $F = \{f\ , \mathcal{V}(f) \cap (x \oplus B) \neq \varnothing\}$. Le problème se résout donc à déterminer les noyaux F_x de x, de manière à calculer la distance exacte $d(x, f_x), f_x \in F$. En effet, les noyaux de x ne sont pas nécessairement réduit à un singleton, du fait de la forme quelconque de l'ensemble $\mathbf{X}$ de départ (qui peut être considéré comme une union de singletons).
  Le problème se résume à savoir si l'ensemble des noyaux de x peut être déterminé par les voisins de x, selon la construction proposée. Soit donc $f_x \in F$ un noyaux de x, f_x étant un singleton, la cellule associée $\mathcal{V}(f_x)$ est convexe, et connexe par arc [(v)]. De même prenons f_a un noyau de $a \in A$. Soit b un point de $\mathcal{V}(f_x)$ à la frontière de $x \oplus B$. Nous avons:
  $$\begin{aligned} d(a, f_x) &\leq d(a, b) + d(b, f_x) \\ &\leq d(a, b) + d(b, f_a) \\ &\leq 2 \cdot d(a, b) + d(a, f_a) \end{aligned}$$
  Ceci est vrai pour tout b dans $\mathcal{V}(f_x)$. La cellule $\mathcal{V}(f_x)$ rencontre une des faces du polyèdre délimité par les points $\mathbb{Z}^n \cap (x \oplus B)$ (enveloppe convexe de ces points). Cette cellule est dans un cas extrême d'épaisseur nulle (c'est à dire un segment rencontrant x). Soit $\{a_i\} \in A$ les points de la face contenant b. Puisque cette face est convexe, nous pouvons écrire $b = \sum_i \lambda_i a_i$, avec $\sum_i \lambda_i = 1$. Nous avons alors, par convexité de d et pour tout k:
  $$\begin{aligned} d(b, a_k) &\leq \textstyle\sum_j \lambda_j d(a_j, a_k) && \text{(par convexité)} \\ &\leq (1 - \lambda_k) \vee_{k \neq j} d(a_j, a_k) \\ &\leq \tfrac{n_i - 1}{n_i} \vee_{k \neq j} d(a_j, a_k) && \text{(par symétrie du problème)} \end{aligned}$$
  avec $n_i = \sharp\{a_i\}$.
  Nous terminons donc la démonstration en soulignant d'une part que $d(a, f_x) \geq d(a, s_a)$, et d'autre part en choisissant $\kappa_B = 2 \cdot \vee_i \left(\frac{n_i - 1}{n_i} \vee_{k \neq j} d(a_j, a_k) \right)$, pour chaque face i du polyèdre, a_j, a_k appartenant à la même face. Nous avons alors $f_x \in s_x$.

■

Travailler sur une trame discrète implique une certaine « myopie » pour les sources potentielles de chaque points pour un algorithme à base de propagation de chaînes. Il ne faut plus considérer uniquement les sources des points (ie. les points de $\mathbf{X}$ produisant une distance minimale), mais les sources rectifiées selon la myopie induite par la trame.

La proposition précédente présente un moyen algorithmique d'obtenir une fonction distance exacte. Il s'agit d'une construction *suffisante* pour obtenir notre fonction distance, mais cette construction n'est pas *nécessaire*, c'est à dire *minimale*.

Remarque: L'isotropie du problème nous permet de dire que seules les boules unités B_1 et B_∞ nous permettent de travailler en trame discrète. Pour ces deux boules, la proposition 3.4 reste valide, la différence se situant dans le calcul de κ_B. Puisque $B_1 \in B_\infty$, l'union permettant de construire l'ensemble s_x fera intervenir plus de voisins pour B_∞ que pour B_1. Enfin, la formulation est tout à fait indépendante de la dimension de travail, ainsi que de la distance d utilisée pour le calcul de la transformée.

[(v)] propriété propre aux Voronoï de distance convexes

3.2 Distances exactes en n dimensions

Selon les développements et remarques qui précèdent, nous sommes à présent capables de calculer une transformation en distance exacte pour toute distance convexe. L'algorithme proposé ne fait que reprendre la proposition 3.4.

3.2.1. Algorithme proposé

D'après les développements de la partie précédente, nous avons vu que dans les cas simples en deux dimensions, les concavités entraînent pour un même point l'existence de plusieurs points sources produisant une distance minimale. Si l'unicité des points sources n'est pas vérifiée en dimension 2, elle ne l'est pas dans le cas général.

Par ailleurs, nous devons garder en mémoire non seulement les points sources de distance minimale par rapport à l'ensemble d'intérêt $\mathcal{F}$, mais également tout ceux vérifiant la condition lipchitzienne, et ce même s'ils ne produisent pas une distance minimale à l'ensemble $\mathcal{F}$.

Cette remarque résume la principale difficulté de l'algorithme, qui concerne la définition de la structure de données des points sources durant la propagation. La seule difficulté réside dans cette structure, les autres aspects de l'algorithme ayant été résolus.

La structure *informatique* de l'enveloppe est représentée en figure 3.4. À chaque point de l'enveloppe nous associons une liste de paire d'éléments: chaque paire contient la coordonnée réelle d'un point source, ainsi que sa distance au point courant. Étant donné que nous ne connaissons pas *a priori* les points sources de distance minimale, nous ne pouvons également pas connaître les points sources dont la distance vérifie la condition lipchitzienne. Il faut donc construire cet ensemble au cours de la découverte des voisins d'un point d'intérêt. Enfin, au cours de cette découverte, une réduction de la liste peut être effectuée à partir des nouvelles informations apportées par un voisin.

Puisque nous devons accéder rapidement à un point de l'enveloppe, nous nous servons d'un type « dictionnaire » pour l'indexage des points de l'enveloppe. Enfin, parallèlement à ce dictionnaire, nous nous servons d'une file d'attente hiérarchique pour accéder aux points de l'enveloppe dont la distance est minimale.

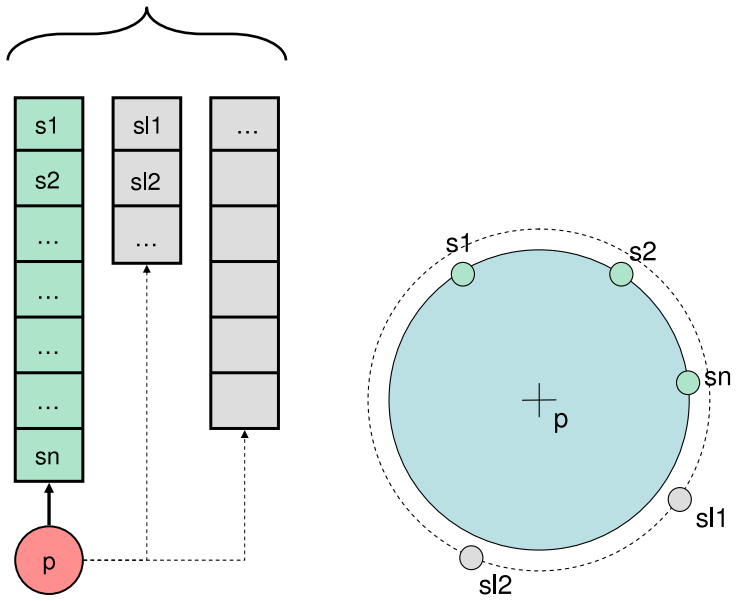
3.2.1.1. Description de l'algorithme

L'algorithme se décompose d'une étape d'initialisation 3.1 et de propagation 3.2. Dans la suite, nous noterons par s un point source, p un point de l'enveloppe en cours de construction, et enfin v un voisin.

Description de l'algorithme 3.1 : L'initialisation parcourt l'ensemble des points de l'image d'entrée $\mathcal{I}_{\mathbf{E}}$ et prépare la file d'attente en vue de la propagation. Les points d'intérêt de $\mathcal{I}_{\mathbf{E}}$ sont les points non nuls: ces points sont les points sources « s ». Pour chacun de ces points, l'algorithme parcours l'ensemble des voisins « v ». La distance « d_s » entre v et s est systématiquement calculée (ligne 10).

Deux cas se présentent alors: soit nous rencontrons v pour la première fois, soit nous avons déjà des informations à propos de ce point. Nous pouvons déduire cette information par l'état correspondant dans l'image de travail $\mathcal{I}_W$ (cf. test ligne 11). Dans le cas où nous avons déjà rencontré ce point, l'état correspondant dans l'image de travail est « dans−la−file », et « non−traité » dans le cas contraire.

- Si nous rencontrons ce point pour la première fois, nous initialisons une donnée v dans la file, avec l'information de distance d_s et la source s de ce point. À s nous associons également sa distance d_s pour effectuer des filtrages de points, comme nous le verrons par la suite. d_s est aussi dans ce cas la distance minimale connue pour l'ensemble des sources de v.
- Dans le cas où nous connaissons déjà ce point, une entrée v existe dans la file f. Nous récupérons cette entrée (ligne 13) sous forme de *distance minimale connue* « $d_{prev.}$ » et *liste de points source* « paires−sources ». Chaque élément de cette liste est composée de la paire: la coordonnée du point source s_i et sa distance d_i à v. Les cas qui se présentent sont les suivants:
  1. s est à une distance trop grande de v (ligne 14): cela se traduit par la condition $d_{prev.} + \kappa_B < d_s$ sur les distances, où κ_B est déterminé selon la proposition 3.4.



(a) Structure de donnée pour le stockage des points

(b) Une configuration correspondant à la file présenté en (a) : les points s_1 à s_n sont sur le même cercle à une distance $d_\wedge$ de p. Les points s_{l_1} et s_{l_2} sont également sur un cercle à une distance $d > d_\wedge$ mais vérifiant la condition lipchitzienne.

Fig. 3.4.: Structure de donnée pour le stockage des points. Le point de l'espace concerné est en rouge. Sont associés à ce point un certain nombre de liste de point source contenu dans $\mathcal{B}$. La liste en vert représente les points à la distance minimale $d_{prev.}$ de l'ensemble (point s_1 à s_n), alors que les deux autres listes en gris sont des points vérifiant la condition lipchitzienne par rapport à $d_{prev.}$.

2. s est à une distance de v telle que la condition lipchitzienne est vérifiée, et ne change pas l'ordre des distances de v (ligne 17): ceci est donné par la condition $d_{prev.} \leq d_s \leq d_{prev.} + \kappa_B$ sur les distances. Dans ce cas, nous ajoutons simplement la paire (d_s, s) à l'ensemble des paires sources de v, et ne changeons pas la distance minimale $d_{prev.}$ de v.
3. s est à une distance de v telle que les sources précédentes sont invalidées (ligne 22): ceci est donné par la condition $d_s + \kappa_B < d_{prev.}$ sur les distances (les sources précédentes ne vérifient plus la condition lipchitzienne). Dans ce cas, la totalité des sources précédentes est supprimée et les sources de v sont initialisées selon s.
4. s est à une distance de v telle que la condition lipchitzienne est vérifiée, et change l'ordre des distances de v (à partir de la ligne 27) : dans ce cas, la distance minimale d_s remplace la distance minimale $d_{prev.}$ précédente, et s est inséré dans une nouvelle file temporaire « paires−sources−temporaire ». Ensuite les points sources précédents (de « paires−sources »), vérifiant la condition lipchitzienne, cette fois selon d_s, sont insérés dans cette file temporaire (ligne 29). L'ancienne liste ne peut déjà contenir s. Enfin v est initialisé avec d_s et cette nouvelle liste de points.

Enfin, les distances des des points découverts sont ajoutés à la file d'attente hiérarchique « fah−distances » (ligne 43), pour pouvoir par la suite traiter les points séquentiellement selon les distances croissantes. Nous notons par $f[v].d_{prev.}$ la distance minimale pour le point v (connue directement grâce à la structure de donnée).

L'image de travail $\mathcal{I}_W$ nous permet de savoir si un point est déjà dans la file f. La raison de cette image supplémentaire est la suivante: les temps d'accès aux points de f sont de l'ordre de $O(\log N)$ alors que ceux de l'image sont en $O(1)$. Étant donné que cette condition est testée pour de nombreux points lors de la propagation, nous avons trouvé cette solution plus adéquate. Il est toutefois possible de n'utiliser que la file f pour tester cette condition.

Description de l'algorithme 3.2 : Cet algorithme construit la nouvelle enveloppe à partir de l'enveloppe précédente. On extrait dans un premier temps la distance minimale à partir de la file d'attente hiérarchique, puis on parcourt ce sous-ensemble de points de distance minimale. Ceci se fait simplement puisque notre file d'attente permet le parcours de plateaux.

Nous commençons par valider les distances de l'enveloppe précédente dans l'image de sortie $\mathcal{I}_\mathbf{D}$. Comme nous le verrons par la suite, il est possible de créer des doublons de distance dans cette *fah*. Cela survient lorsque l'on met à jour la distance d'un point. Étant donné l'éventualité de doublons dans la *fah*, et puisque les points sont traités dans l'ordre de leur distance, la validation de la distance dans l'image de sortie n'est pas effectuée sur les points déjà valides (ligne 7).

La suite est assez similaire à l'étape d'initialisation, la différence étant que les points de l'enveloppe contiennent un ensemble de points source, qui ne se réduisent pas à un point unique. Cet ensemble est extrait, et la distance de la source précédente au point courant est calculée (ligne 17). La distance minimale « $d_\wedge$ » est également calculée pendant cette étape. Il s'agit de la distance minimale entre v et les sources s_i issues de p. Aucun filtrage n'est effectué ici.

Les mêmes tests que pour l'étape d'initialisation sont ensuite effectués à partir de la distance minimale $d_\wedge$ issue du point p de l'enveloppe précédente. À noter que puisque la distance change lors d'une collision (point connu et nouvelle information), il faut ajouter de nouveau ce point dans la *fah* des distances; ce qui explique la ligne 41. Cette distance reste bien-sûr au dessus de la distance minimale associée à la *fah*. Puisque d'autre collision peuvent intervenir, l'ajout dans la *fah* n'est effectuée qu'après parcours total des points de distance courante (ligne 44). Enfin les points traités sont retirés de la file de points f.

Algorithme 3.1 : Distance euclidienne exacte nD - Initialisation

Data : image d'entrée $\mathcal{I}_{\mathbf{E}}$
Result : image des distance $\mathcal{I}_{\mathbf{D}}$

$\mathcal{I}_W \leftarrow$ non-traité
$f \leftarrow \varnothing$
f-candidat $\leftarrow \varnothing$
fah-distances $\leftarrow \varnothing$
forall $s \in \mathcal{I}_{\mathbf{E}}$ **do**
  if $\mathcal{I}_{\mathbf{E}}(s) \neq 0$ **then**
    $\mathcal{I}_W(s) =$ traité // les points de la forme sont de distance nulle et marqués traités
    forall $v \in \{\mathcal{N}_s \setminus s\}$ **do**
      if $\mathcal{I}_{\mathbf{E}}(v) = 0$ **then**
        $d_s = d(v, s)$
        if $\mathcal{I}_W(v) =$ *dans-la-file* **then**
          ▶ *nous connaissons ce point*
          $d_{prev.},$ paires-sources $= f[v]$
          if $d_{prev.} + \kappa_B < d_s$ **then**
            continue // passage au voisin suivant
          else if $d_{prev.} \leq d_s \leq d_{prev.} + \kappa_B$ **then**
            if $s \notin$ *paires-sources* **then**
              paires-sources $\leftarrow$ ajouter (d_s, s) // Ajout du couple dans la liste des sources
          else if $d_s + \kappa_B < d_{prev.}$ **then**
            $f[v] = (d_s, [(d_s, s)])$ // remplacement total des points précédents
          else
            ▶ *cas correspondant à $d_s < d_{prev.}$: filtrage de la liste de points*
            paires-sources-temporaire $\leftarrow (d_s, s)$
            forall $(d_i, s_i) \in$ *paires-sources* **do**
              if $d_i < d_s + \kappa_B$ **then**
                paires-sources-temporaire $\leftarrow$ ajouter (d_i, s_i)
            $f[v] = (d_s,$ paires-sources-temporaire $)$ // remplacement avec la nouvelle liste
        else
          ▶ *nous ne connaissons pas ce point*
          $\mathcal{I}_W(v) =$ dans-la-file
          $f[v] = (d_s, [(d_s, s)])$
          f-candidat $\leftarrow$ ajouter v
forall $v \in$ *f-candidat* **do**
  fah-distances $\leftarrow$ ajouter $f[v].d_{prev.}$

Algorithme 3.2 : Distance euclidienne exacte nD - Construction de la nouvelle enveloppe

```
1  while f ≠ ∅ do
2      f-candidat ← ∅
3      ▶ validation des distances minimales
4      d∧ ← clef minimale de fah-distances
5      PL ← points de distance minimale de fah-distances
6      forall p ∈ PL do
7          if I_W(p) = traité then continue                          // point déjà traité
8          I_W(p) = traité
9          I_D(p) = d∧
10     ▶ propagation
11     forall p ∈ PL do
12         forall v ∈ {N_p \ p} do
13             if I_W(v) = traité then continue
14             d∧ = +∞
15             f-file ← ∅                  // construction d'une nouvelle file de couples (distance, source)
16             forall (d_i, s_i) ∈ f[p].paires-sources do
17                 d_s = d(v, s_i)
18                 d∧ = d_s ∧ d∧
19                 f-file ← ajouter (d_s, s_i)
20             if I_W(v) ≠ dans-la-file then
21                 f-file ← suppression des points tels que d_s > d∧ + κ_B
22                 I_W(v) = dans-la-file
23                 f[v] = (d∧, [(d∧, f-file )])
24                 f-candidat ← ajouter v
25
26             else
27                 d_prev., paires-sources = f[v]          // récupération des informations précédentes
28                 if d_prev. + κ_B < d∧ then continue   // le nouveau point v ne satisfait pas la condition
                   lipschitzienne
29                 if d_prev. ≤ d∧ then
30                     f-file ← suppression des points tels que d_s > d_prev. + κ_B
31                     paires-sources ← ajouter f-file pour les couples ne se trouvant pas dans paires-sources
32                 else
33                     f-file ← suppression des points tels que d_s > d∧ + κ_B
34                     if d∧ + κ_B < d_prev. then
35                         f[v] = (d∧, [(d∧, f-file )])
36                     else
37                         paires-sources ← suppression des points tels que d_s > d∧ + κ_B
38                         paires-sources ← ajouter f-file
39
40
41                 f-candidat ← ajouter v
42
43     forall v ∈f-candidat do
44         fah-distances ← ajouter f[v].d_prev.
45     forall v ∈ PL do
46         f ← supprimer v
```

3.2.2. Résultats

Nous plaçons ici quelques résultats en dimension 4 de l'algorithme. Nous donnons pour cela des coupes $3D$ d'une image initialement $4D$. Nous avons essayé les distances euclidiennes et d_5, ainsi que quelques distances non isométriques. La suppression d'une dimension par projection dans l'espace $3D$ produit des boules équivalentes dans l'espace réduit, ou plus simplement, des boules euclidiennes $4D$ projetées en dimension 3 sont des boules euclidiennes $3D$. Dans les figures qui vont suivre, nous avons fixé pour toutes les coupes la même iso-distance, que le lecteur apercevra sous forme de « patatoïde » jaune. Il s'agit d'une coupure de l'image des distances à une valeur fixée, projetant ainsi la boule (de taille égale à la coupure) dans l'espace $3D$. Ensuite, la variation en taille selon deux coupes successives permet d'apprécier la 4 ème dimension, non visible directement.

Nous avons effectué deux séries d'expériences, la première suppose l'espace totalement isométrique avec les distances ℓ^2 et ℓ^5. Pour ces images, nous avons tiré 30 points aléatoirement et uniformément dans l'espace $4D$, dont la taille est $100 \times 100 \times 20 \times 10$ (soit $2 \cdot 10^5$ points). Nous avons ensuite testé des distances non-isométriques, afin de vérifier le fonctionnement de l'algorithme lorsque les différent axes n'ont pas la même résolution, ainsi que l'utilisation de rotation entre les différents axes. Pour ces expériences, nous avons tiré 10 points répartis dans une image de taille $100 \times 100 \times 50 \times 20$.

Pour toutes les expériences, nous avons vérifié s'il existait un quelconque biais pour tous les points de l'espace, afin de vérifier notre algorithme, et le résultat est heureusement totalement exact.

Résultat en ℓ^2

La figure 3.5 présente les résultats de l'algorithme pour la métrique euclidienne. Les résultats sont totalement conformes à nos attentes. Nous obtenons effectivement, sur chacune des coupes de la dimension 4, un volume contenant des boules euclidiennes $3D$. Ces boules varient en taille selon l'éloignement à leur centre sur la dimension 4 (ie. la coupe).

Résultat en ℓ^5

La diminution en taille des boules selon ℓ^5 suivant les coupes est moins rapide que pour la boule euclidienne. C'est pourquoi nous constatons une certaine rémanence des boules de la figure 3.6 sur les différentes coupes. Leur « disparition » est par contre assez « rapide », dans le sens où peu de coupes successives permettent de faire apparaître ou disparaître presque totalement les boules.

Distances ℓ^2 non-isométriques

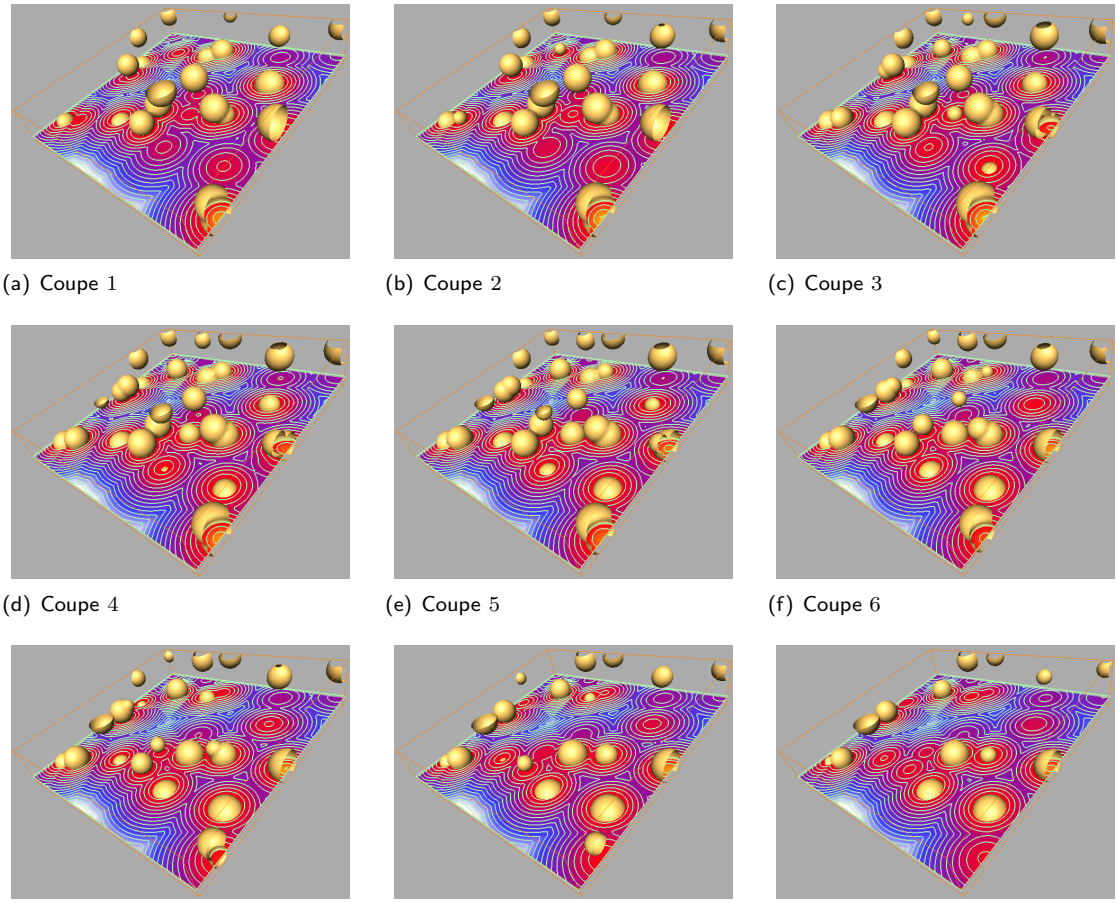
Nous voulons vérifier s'il est possible de calculer des distances non-isométriques, avec le même algorithme. Les figures 3.7 et 3.8 sont le résultat de l'application d'une transformation matricielle avant le calcul effectif de la distance euclidienne. La matrice de transformation comporte une rotation autour des axes, et une homothétie centrée dont les rapports sont différents sur chacun des axes :

$$\begin{aligned} v &= \mathcal{M}_r \cdot \mathcal{M}_\Theta \left(p - s\right) \\ d &= \|v\|_2 \end{aligned}$$

avec $\mathcal{M}_r$ matrice diagonale des rapports des axes, $\mathcal{M}_\Theta$ matrice de rotation, et enfin p et s respectivement le point courant, et la source considérée. Puisque le calcul des distances est plus complexe, le calcul dure plus longtemps.

Dans nos tests, nous avons décelé quelques biais [(vi)] de l'ordre de 10^{-2} . Nous attribuons néanmoins ces biais à une erreur numérique entre l'algorithme, écrit en *C++*, et la méthode utilisée dans la vérification, écrite en langage de script Python.

[(vi)] 28 points sur 10^7

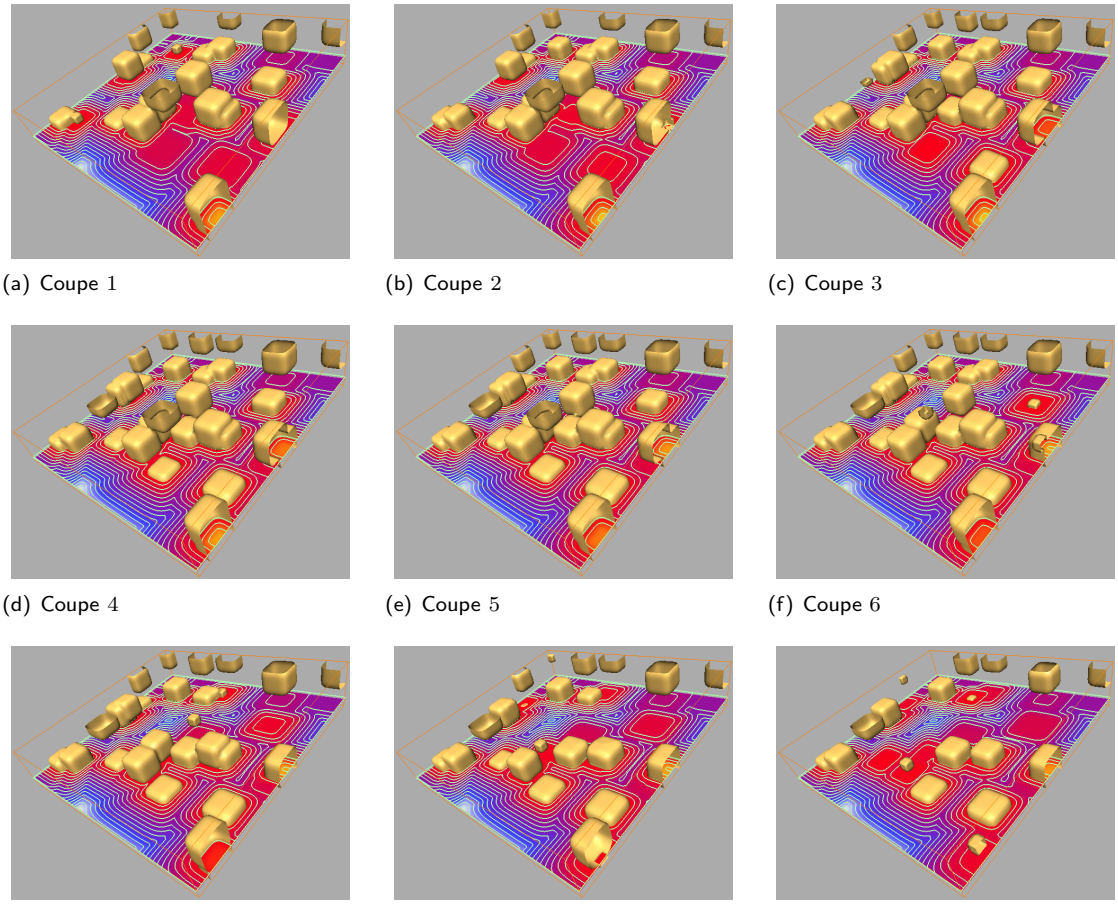


(a) Coupe 1 (b) Coupe 2 (c) Coupe 3

(d) Coupe 4 (e) Coupe 5 (f) Coupe 6

(g) Coupe 7 (h) Coupe 8 (i) Coupe 9

Fig. 3.5.: Distance euclidienne sur un ensemble de 30 points. Coupure à $d = 6$.

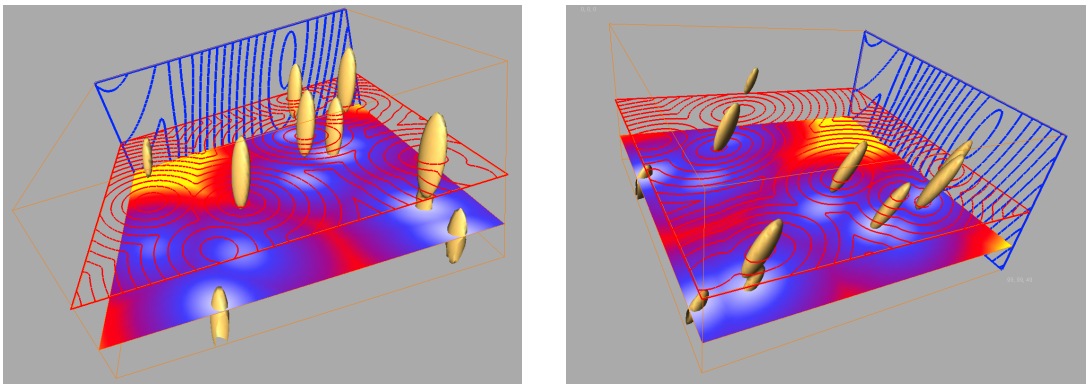


(a) Coupe 1 (b) Coupe 2 (c) Coupe 3

(d) Coupe 4 (e) Coupe 5 (f) Coupe 6

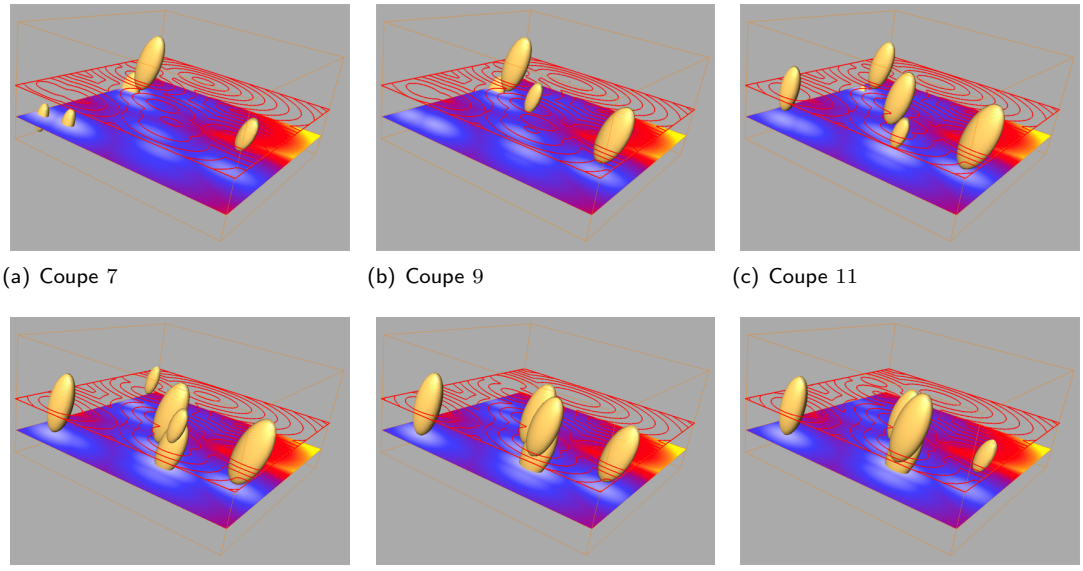
(g) Coupe 7 (h) Coupe 8 (i) Coupe 9

Fig. 3.6.: Distance ℓ^5 sur un ensemble de 30 points. Coupure à $d = 6$.



(a) Vue 1 (b) Vue 2

Fig. 3.7.: Distance non-isométrique. Les angles choisis ici sont, selon l'ordre croissant des axes et en degrés, $\{60, 20, 0\}$.



(a) Coupe 7 (b) Coupe 9 (c) Coupe 11

(d) Coupe 13 (e) Coupe 15 (f) Coupe 17

Fig. 3.8.: Distance non-isométrique. Les angles choisis ici sont, selon l'ordre croissant des axes et en degrés, $\{10, 20, 30\}$. L'angle suivant la dernière dimension donne un sens de « mouvement » apparent, le long des coupes.

3.2.3. Méta-programmation

Les structures introduites dans le chapitre 2 nous ont permis d'écrire cet algorithme de manière générique, indépendamment de la dimension de l'image et de la fonction distance. L'algorithme à base de propagation proposé repose sur les deux mécanismes suivants:

1. *les itérateurs* : comme nous l'avons mentionné dans le chapitre précédent, les itérateurs permettent un parcours de l'image sans manipulation directe des coordonnées. Ainsi, l'algorithme ne génère pas lui-même de coordonnées, mais utilise celles générées par l'itérateur. Les itérateurs dépendent de l'image à partir de laquelle ils sont créés. Par ailleurs, l'algorithme ne « fabrique » pas d'itérateurs, mais demande à l'image de les fabriquer pour lui. L'algorithme reste alors indépendant de la dimension de l'image.
2. *les voisinages (graphes de connexité)*: ces structures permettent de propager les informations d'une enveloppe à une autre, et ainsi de couvrir la totalité de l'espace de l'image. À l'instar des itérateurs d'image, les voisinages génèrent des itérateurs de voisinages qui évitent la manipulation directe de coordonnées dans l'algorithme. Par ailleurs, les méthodes de centrage de voisinage prennent en entrée des informations issues des itérateurs d'image.

L'algorithme sert ici de substrat dans lequel transitent des informations entre les différentes structures: itérateurs, voisinages et fonction distance. La file d'attente hiérarchique permet de créer un ordre de traitement entre les différents points, de manière à éviter un passage multiple sur les points. L'algorithme au final orchestre convenablement les opérations entre ces différentes structures de données et de traitement, mais ne génère pas directement des données. Pour conclure sur ce point, l'environnement de programmation que nous avons introduit a conduit à la définition réelle d'un algorithme nD. La modification de la dimension des images en entrée n'a aucun impact sur l'algorithme, et l'utilisateur n'a pas besoin d'y apporter une modification quelconque.

3.2.4. Conclusion

Dans cette partie, nous avons montré la faisabilité d'un algorithme générique de calcul de distance. La généricité se manifeste à la fois sur la dimension des images considérées, et sur la fonction distance

utilisée. Cet algorithme n'a pas été étudié pour être spécialement rapide, mais l'utilisation astucieuse des structures informatiques permet un temps de calcul assez réduit, même pour des distances complexes.

3.3 Quasi-distances

Nous abordons dans cette partie une classe particulière de fonction distance morphologique, les *quasi-distances*. Définies récemment par Beucher [Beu05, Beu03], les quasi-distances sont une extension des distances morphologiques binaires aux images numériques. Plus précisément, les quasi-distances font partie d'une classe plus générale de transformations appelées transformations résiduelles.

Les transformations résiduelles étaient jusqu'aux récents travaux cités, exclusivement envisageables sur des ensembles, et leur extensions aux fonctions numériques posaient quelques problèmes de définition. Grâce au nouveau formalisme, l'extension algébrique des transformations résiduelles est désormais possible.

Nous allons dans un premier temps rappeler les distances morphologiques ainsi que quelques unes de leurs applications. L'expression de ces distances sous forme ensembliste sera l'occasion d'introduire les transformations résiduelles. Nous verrons ensuite la stratégie de Beucher pour l'extension de ces transformations au cadre numérique. Nous présenterons un algorithme pour le calcul rapide des quasi-distances, et nous évaluerons ses performances.

Enfin, la stratégie de Beucher pour l'extension au cadre numérique peut être utilisée pour des images multi-valuées, à partir du moment où les espaces sont munis d'une structure d'ordre et d'une métrique. Nous terminerons ce chapitre par la présentation de résultats de la quasi-distance sur des images couleurs.

3.3.1. Introduction

Cette introduction s'articule de la manière suivante: premièrement nous allons définir la notion de distance morphologique. Nous définirons ensuite les transformations résiduelles et formulerons les distances morphologiques dans ce contexte. Nous verrons comment il est possible de formuler les transformations résiduelles dans un cadre numérique et enfin appliquerons cette formulation dans le cas particulier des distances morphologiques. Cette formulation aura pour conséquence la définition d'un nouvel opérateur: la quasi-distance.

3.3.1.1. Les distances morphologiques

La distance morphologique est une manière de bâtir une fonction distance directement à partir d'une opération élémentaire d'érosion ou de dilatation, selon l'ensemble à partir duquel on souhaite calculer la distance. Elle se définit de manière récurrente de la manière suivante:

Soit $\mathbf{X}$ l'ensemble d'intérêt, nous cherchons à obtenir une image $\mathcal{D}$ telle qu'en tout point x de $\mathcal{D}$, nous ayons $\mathcal{D}(x) = d_m(x, \mathbf{X})$, pour une certaine fonction distance d_m. Ainsi, une érosion ϵ est appliquée de manière récursive à $\mathbf{X}$, et d_m est simplement une valuation de l'étape de récurrence. Cela se résume de la manière suivante:

$$d_m : \left\{ \begin{array}{ccc} \mathbf{X} & \rightarrow & \mathbb{N} \\ x & \mapsto & \arg_i \max\{\epsilon^{(i)}(\mathbf{X}) \ni x\} \end{array} \right.$$

Cette distance a la bonne propriété d'être dans $\mathbb{N}$, ce qui rend son maniement extrêmement simple, et comme on peut le voir dans l'expression de d_m, son calcul est trivial.

Le champ d'application des fonctions distance est relativement lié à celui des distances de type l^p décrit dans la section §3.2 précédente. Une fonction distance est représentative de la taille des structures, ou de l'éloignement de structures que l'on souhaite séparer. Si l'expression des transformées en distance de type l^p de la section §3.2 pose un cadre d'espace simplement métrisable - la topologie sous-jacente résultant de la métrique -, les transformations en distance morphologique ensembliste ne font intervenir que des notions de voisinage dans l'espace, et donc des espaces aux propriétés moins fortes que celles des espaces métriques.

Les quelques développements de la section §3.2 mettent en évidence une analogie entre les deux approches, lorsque l'élément structurant est défini par la boule unité d'une métrique d_k: la dilatation

$\delta^{(i)}$ (et dualement l'érosion $\epsilon^{(i)}$ pour $i \in \mathbb{N}^*$) de $\mathbf{X}$ revient à calculer la distance à $\mathbf{X}$ selon d_k [(vii)].

Dans le cas de la distance morphologique, l'élément structurant sert davantage comme graphe de connexité que d'élément structurant valué. Il sert à définir les voisins à une distance unitaire d'un point, et établit ainsi la structure (topologique) de voisinage. Dans ce sens, il n'est pas aberrant de ne s'intéresser qu'aux érosions par un élément structurant plan.

3.3.1.2. Transformations résiduelles

Une transformation résiduelle Θ sur un ensemble $\mathbf{X}$ est définie comme la donnée d'une famille - ou suite - de paires de fonction $\left\{(\psi_i, \zeta_i)_{(i)}, i \in \mathbb{N}^*\right\}$ jointe à une famille de fonctions binaires de combinaison $(r_i)_{(i)}$, et tel que :

$$\Theta : \left\{ \begin{array}{lcl} \mathcal{P}(E) & \rightarrow & \mathcal{P}(E) \\ \mathbf{X} & \mapsto & \bigcup_i r_i\left(\zeta_i(\mathbf{X}), \psi_i(\mathbf{X})\right) \end{array} \right. \tag{3.5}$$

Les éléments de $(r_i)_{(i)}$ sont appelés les *résidus* et ceux de $(\psi_i, \zeta_i)_{(i)}$ les *primitives*. L'équation 3.5, définie ainsi, est cependant beaucoup trop générale pour être exploitable; nous allons fournir quelques exemples afin d'éclairer la discussion.

Érosion ultime : L'érosion ultime est une transformation résiduelle, telle que primitives et résidus sont définis comme suit:

$$\forall i, \left\{ \begin{array}{lcl} r_i & : (\cdot, \star) & \mapsto \cdot \setminus \star \\ \zeta_i & : \mathbf{X} & \mapsto \gamma^{\infty}_{\mathbf{X}} \circ \epsilon^{(i+1)}(\mathbf{X}) \\ \psi_i & : \mathbf{X} & \mapsto \epsilon^{(i)}(\mathbf{X}) \end{array} \right.$$

où $\gamma^{\infty}_{\mathbf{X}}$ est l'ouverture par reconstruction [(viii)] dans $\mathbf{X}$ et pour les résidus r_i, $A \setminus \varnothing = A$, $A \subset B \Rightarrow A \setminus B = \varnothing$, $\varnothing \setminus \varnothing = \varnothing$.

Fonction indicatrice : La fonction indicatrice associée à Θ informe l'occurrence de l'activité de Θ sur un point. Elle est donc restreinte au support de Θ. Elle est définie de la manière suivante:

$$q_\Theta : \left\{ \begin{array}{lcl} \mathbf{supp}(\Theta) & \rightarrow & \mathbb{N} \\ x & \mapsto & \arg_i\left(x \in r_i\right) + 1 \end{array} \right.$$

c'est à dire l'occurrence i du résidu r_i contenant un point x de $\mathbf{supp}(\Theta)$. Dans le cadre ensembliste, les primitives sont choisies de manière à ce que l'indicatrice soit unique pour chaque point de $\mathbf{supp}(\Theta)$. Nous verrons que ce choix s'avère difficile dans le cadre numérique.

Distance morphologique : L'expression de la distance morphologique selon la définition d'opération résiduelle est immédiate: les primitives sont des érosions de taille croissante, le résidu est toujours la soustraction ensembliste, et la fonction distance est l'indicatrice associée à Θ :

$$\Theta_{dm} : \forall i, \left\{ \begin{array}{lcl} r_i & : (\cdot, \star) & \mapsto \cdot \setminus \star \\ \zeta_i & : \mathbf{X} & \mapsto \epsilon^{(i)}(\mathbf{X}) \\ \psi_i & : \mathbf{X} & \mapsto \epsilon^{(i+1)}(\mathbf{X}) \end{array} \right. \tag{3.6}$$

et

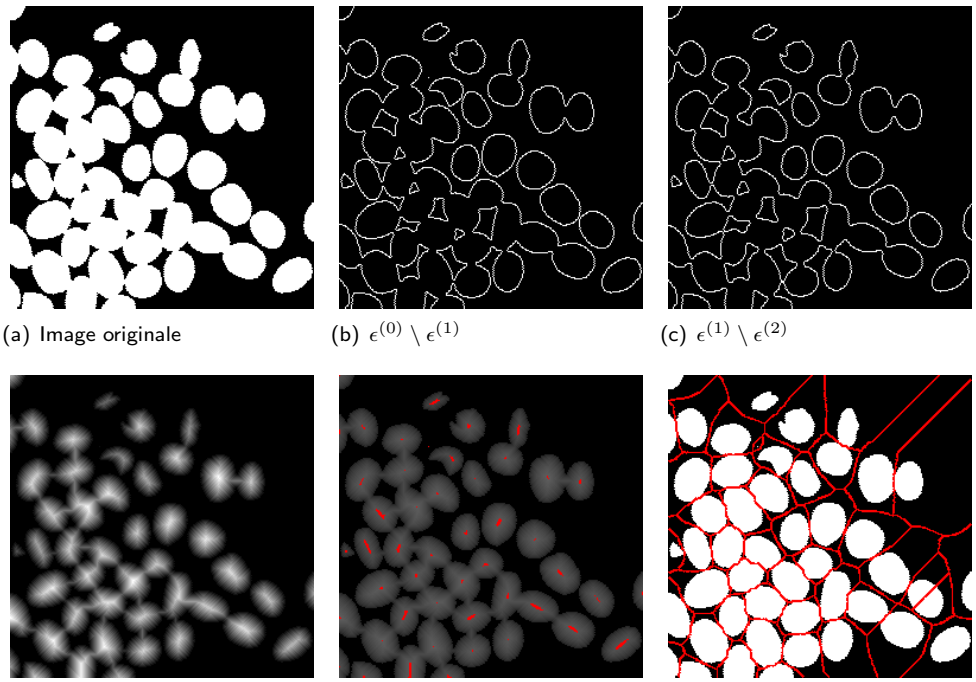
$$d_m \equiv q_{\Theta_{dm}}$$

[(vii)] il est aisé de voir que l'on passe de la première à la seconde représentation en s'intéressant à la distance par rapport à $\mathbf{X}$ ou à $\mathbf{X}^c$

[(viii)] la reconstruction est un opérateur morphologique classique qui n'est pas abordé ici. Le lecteur peut se référer au §4.3.1 page 130 où cet opérateur est abordé dans un contexte lexicographique.

Il s'agit ici d'une expression formelle dans laquelle aucune considération algorithme n'intervient. Il va de soi qu'il est plus efficace de ne considérer qu'un sous-ensemble du support (les contours intérieurs) pour le calcul des érodés successifs.

Les distances morphologiques sont utilisées depuis longtemps [LB81]. Une application très connue de cette distance est la « séparation de grains », illustrée en figure 3.9. La propriété utilisée est la convexité des grains: l'enveloppe de deux grains superposés présente dans la plupart des cas des zones de concavité. Par application de la fonction distance, une telle zone est jointe à une autre zone concave, la plus proche parmi toutes les zones concaves. La ligne de jonction a alors une distance inférieure à celle de l'intérieur des grains superposés, ce qui permet la séparation par *ligne de partage des eaux*.



(a) Image originale (b) $\epsilon^{(0)} \setminus \epsilon^{(1)}$ (c) $\epsilon^{(1)} \setminus \epsilon^{(2)}$

(d) Fonction distance obtenue (e) Maxima locaux de la fonction distance (f) Résultat de la séparation par watershed

Fig. 3.9.: Application de la distance morphologique binaire pour l'exemple de séparation de grains. Les érosions successives permettent de construire la fonction distance de manière itérative - seules les deux premières itérations sont montrées. La fonction distance obtenue caractérise le grain, supposé suffisamment « convexe » pour ne produire qu'un seul maximum. Les grains sont ensuite séparés, par exemple par application de la *lpe* sur l'inverse de l'image de distance.

Bien entendu pour le cas binaire, le cadre des opérations résiduelles est beaucoup trop général pour être d'une quelconque utilité à la définition de distance morphologique. Il devient par contre pertinent lorsque l'on s'intéresse à l'extension des opérateurs dans le cadre numérique. C'est ce que nous allons développer à présent.

3.3.1.3. Extension au cadre numérique

Partant de la définition précédente, l'application des transformations résiduelles au cas numérique nous expose à un certain nombre de problèmes.

Résidu : Le premier problème concerne la fonction résidu. Dans le cas des distances morphologiques, le résidu était défini comme étant la soustraction ensembliste, qui est une opération à deux paramètres,

croissante pour le premier et décroissante pour le second. La soustraction numérique semble être la candidate directe:

$$\forall i,\, r_i = |\zeta_i - \psi_i|$$

Non-unicité de la valeur du résidu : Le second problème est plus délicat et concerne la fonction indicatrice. Dans le cas binaire, on essayait tant bien que mal de définir les primitives de manière à ce que les résidus soient disjoints deux à deux (le cas de la distance morphologique binaire en est un exemple). Une telle contrainte est très difficile à obtenir dans le cas numérique.

Il est ainsi possible d'avoir plusieurs résidus pour chacun des points du support de l'ensemble $\mathbf{X}$ de départ. Retrouvons notre exemple de distance morphologique, et appliquons les mêmes primitives à une image numérique.

Beucher [Beu03] ne considère qu'une sous-famille de l'ensemble des résidus obtenus en un point $x \in \mathbf{X}$. Il s'intéresse particulièrement aux occurrences du résidu maximal. Ce sont en quelque sorte les occurrences de l'*activité maximale* du couple de primitives. La fonction indicatrice serait alors:

$$q_\Theta : \left\{ \begin{array}{lcl} \mathbf{supp}(\Theta) & \rightarrow & \mathbb{N} \\ x & \mapsto & \arg_i \left(x \in \bigvee_i r_i\right) + 1 \end{array} \right.$$

Selon cette définition, l'indice du résidu maximal n'est pas toujours unique non plus, comme le montre la figure 3.10. Beucher ne choisit alors de garder que la dernière occurrence du résidu maximal, l'indicatrice devient:

$$q_\Theta : \left\{ \begin{array}{lcl} \mathbf{supp}(\Theta) & \rightarrow & \mathbb{N} \\ x & \mapsto & \bigvee_i \arg_i \left(x \in \bigvee_i r_i\right) + 1 \end{array} \right. \tag{3.7}$$

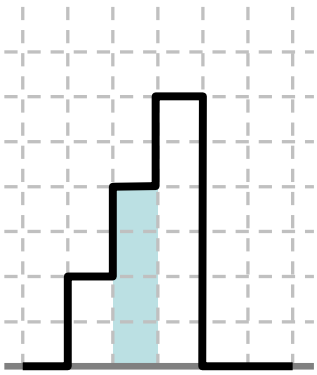
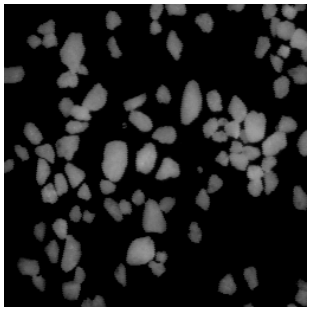


Fig. 3.10.: Non unicité de la valeur maximale du résidu. Dans cet exemple, des érosions successives du profil noir engendrent plusieurs fois la même valeur, maximale, du résidu dans la zone en bleu.

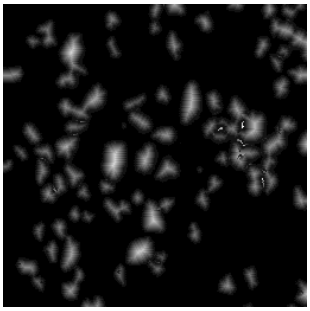
Il s'agit ici d'une convention, aussi valable qu'une autre. Aussi, il est tout à fait envisageable de s'intéresser à des quantiles différents du supremum de l'activité.

3.3.1.4. Retour sur la fonction distance : la « Quasi-Distance »

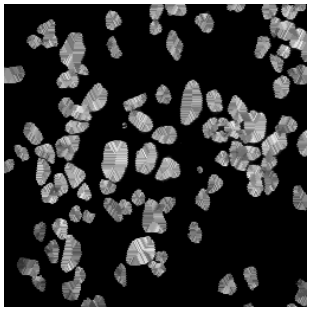
Quasi-distances : Nous avons à présent tous les éléments pour appliquer la distance morphologique dans le cadre numérique. Les primitives sont les mêmes que pour l'équation 3.6, et à l'instar de la distance morphologique classique, la quasi-distance est donnée par l'indicatrice de la transformation résiduelle. Les itérations sont arrêtées lorsque l'on arrive à idempotence, c'est à dire lorsqu'il n'y a plus d'activité du couple de primitives sur l'ensemble de départ. Nous reviendrons sur ce point par la suite. Un exemple de cette transformation est donné sur la figure 3.11.



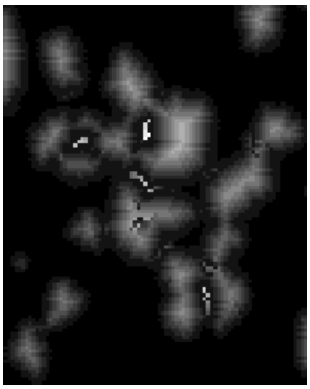
(a) Image originale



(b) Fonction quasi-distance (indicatrice)



(c) Fonction résidu



(d) Caractère non-lipchitzien de la distance

Fig. 3.11.: Quasi-distance sur l'image « Salt »

Problème de régularité : Comme nous pouvons le voir sur la figure 3.11(b) (dont un agrandissement est donné en 3.11(d)), la fonction de quasi-distance peut présenter des sauts importants entre points voisins. Or une fonction distance, comme explicité dans la partie précédente, possède une contrainte lipchitzienne. Ce problème n'apparaît pas dans le cadre binaire dans la mesure où les supports des différents résidus sont disjoints et connexes. Dans le cas numérique, les résidus successifs restent certes connexes, mais se chevauchent car leur support n'est pas disjoint. Le passage à la fonction indicatrice, supremum des résidus - donc non continue - provoque les sauts que l'on observe sur la figure précédente, et dont une illustration schématique est donnée sur la figure 3.12.

La violation de la contrainte lipchitzienne est exprimée par:

$$\exists (x, y) \in \mathbf{X}^2 / \, (y \in \mathcal{N}_x \ \wedge \ d_m(x, y) > 1) \tag{3.8}$$

Comme mentionné, la principale cause de ces irrégularités est due au choix de l'indicatrice: le supremum des résidus n'est pas une fonction continue sur les résidus. Ce n'est pas la seule explication, puisque la fonction $\arg$ n'a elle-même rien de continu. Concernant les occurrences de ces discontinuités, il semble selon le schéma 3.12 que l'une des causes soit l'érosion d'un plateau par un autre. S'il existe deux minima locaux à des hauteurs différentes dans l'image, l'indicatrice de ces minima reste nulle pendant un certain nombre d'itérations. Lorsque les plateaux engendrés par les érosions respectives de ces minima se chevauchent, au moins sur le minimum initial le plus haut, l'indicatrice passe brutalement de 0 à une certaine valeur correspondante à l'itération, strictement supérieure à 1 (sinon le minimum n'aurait pas été un minimum local).

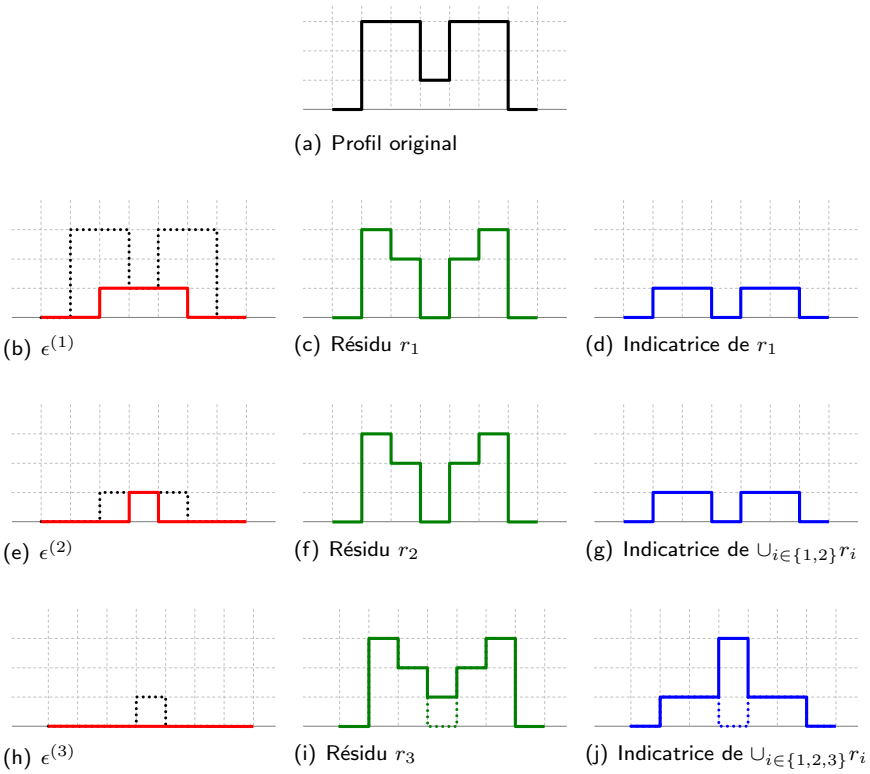


Fig. 3.12.: Violation de la contrainte lipchitzienne de la fonction distance par les érodés successifs dans le cas numérique.

Régularisation : À partir de ces remarques, Beucher a conçu une méthode de régularisation. Pour tous les points ne vérifiant pas la condition lipchitzienne, on attribue la valeur de l'érodé en ce point augmentée de 1. Ce processus est itéré jusqu'à ce qu'aucun point ne viole la condition lipchitzienne. Les algorithmes correspondants sont donnés en 3.3 et 3.4. Nous qualifierons cette approche de « classique ». Ces algorithmes reprennent exactement les définitions introduites et ne seront pas explicités d'avantage.

Utilisation : Cet opérateur est assez récent, et son domaine d'application n'est pas totalement maîtrisé. Sur l'image « Salt » (figure 3.11 précédente), nous observons que pour des images fortement bi-valuées - composante très sombre en fond contrastant avec les grains - cet opérateur permet de s'affranchir facilement d'une valeur de seuil de binarisation (pour succéder ensuite à l'application d'une distance morphologique classique). Dans le cas d'un fond non uniforme, l'intuition nous dicte que pour un tel fonctionnement, le contraste maximal dans le fond doit être inférieur au contraste minimal fond/forme. Ce cas est malheureusement marginal.

Hanbury & Marcotegui [HM06] utilisent avec succès cette propriété sur des images de gradient définis par Malik [MBLS01]. Ces gradients mesurent le contraste en se basant sur des critères locaux multiples (couleur, texture etc.); les lignes de gradients sont valués par cette mesure. Ils ont cependant la particularité de ne pas produire des contours fermés, et sont mal adaptés à des algorithmes de type *ligne de partage des eaux*. L'application de la quasi-distance sur l'inverse de l'image des gradients permet de créer des zones de distance croissante à l'intérieur des lignes de gradient. La distance est alors croissante depuis les contours, de manière analogue à la *séparation de grain* décrite page 69. La *lpe* est ensuite applicable sur l'inverse de ces derniers images (cf. figure 3.13). La quasi-distance per-

Algorithme 3.3 : Calcul de l'indicatrice of q - Algorithme de S. Beucher [Beu03]

Data : I
Result : Q, R
W1 $\leftarrow \mathcal{I}$
Q, R, W2 $\leftarrow 0$
$i \leftarrow 0$
repeat
  i $\leftarrow$ i + 1
  W2 $\leftarrow \epsilon$(W1)
  résidu $\leftarrow$W1$-$W2
  $E \leftarrow$ (résidu $\geq$ R $\wedge$ résidu $\neq 0$)
  Q[E] $\leftarrow \vee\{i$,Q[E]$\}$
  R $\leftarrow \vee\{$résidu,R$\}$
  W1 $\leftarrow$ W2
until *résidu* $= 0$

Algorithme 3.4 : Regularisation de l'indicatrice q - Algorithme de S. Beucher [Beu03]

Data : Q
Result : RQ
RQ $\leftarrow$ Q
$i \leftarrow 0$
repeat
  W $\leftarrow$ RQ$-\epsilon$(RQ)
  $E \leftarrow \neg$(W ≤ 1)
  RQ[E] $\leftarrow \epsilon$(RQ[E]) + 1
until $E = \emptyset$

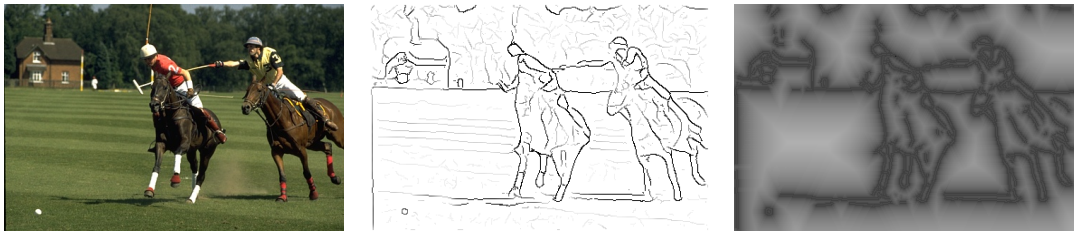
met encore une fois de s'affranchir d'un paramètre de seuil et de préserver l'automatisme du traitement.

Nous présentons sur les figures 3.14, 3.15, 3.16, 3.17 et 3.18 cinq exemples de transformation en quasi-distance pour des images réelles.

L'objet d'intérêt (personne) dans l'image 3.14 est fortement contrasté par rapport au fond. Dans ce genre de cas, la quasi-distance réagit de manière idéale. La légère texture du fond disparaît totalement grâce au contenu sombre de la personne. Par ailleurs, les zone fortement texturées sur la partie droite de l'image nous indiquent des « objets » de petite taille, que l'on peut facilement supprimer par un seuillage sur la distance. Par contre, la quasi-distance sur l'inverse de l'image fait apparaître de nombreux points sur le fond (partie blanche autour de la personne) et corrompt une partie du contour de celle-ci. La combinaison par supremum des deux quasi-distances n'est pas appropriée. La combinaison par infimum nous fera apparaître également de nombreuses zones sans intérêt.

Pour l'image « Fleur Budapest » (3.15), la fleur en avant scène - dont le ton original est rouge foncé - contraste très bien avec l'arrière plan de l'image. Dans ce cas précis, la quasi-distance est une transformation bien adaptée à la séparation des objets d'intérêt, l'image ne présente d'ailleurs pas de bruit ni de texture gênantes.

Sur l'image « Fifer », les habits du personnage présentent une légère texture finement pigmentée. La partie haute est très sombre et la distance reste faible dans l'image directe. Par contre elle ressort très bien dans l'image inversée, et ce même après régularisation. De même la partie basse du personnage possède une texture finement pigmentée, ce qui explique que la distance présente des valeurs basses après régularisation. La distance présente sur le pantalon est globalement due à la frontière très contrastée entre le pantalon et le fond. Le même pigment se retrouve dans le fond. La fonction distance ne



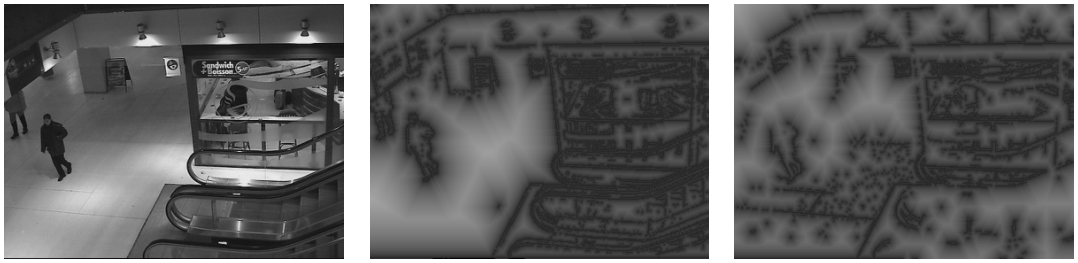
(a) Image originale (b) Inverse du gradient de Malik de (a) (c) Quasi-distance régularisée de (b)

Fig. 3.13.: Utilisation de la quasi-distance sur les gradients de Malik. Le contraste de (b) a été exagérément augmenté pour des raisons de visualisation. (c) l'application de la quasi-distance permet de supprimer un certain nombre de contours: les lignes entre et à gauche des chevaux, les contours de faible amplitude sur la moitié haute de l'image. Le premier avantage est l'absence de paramètre, le second est la transformation de l'image en « bassins » reliant les contours d'intérêt, image qui pourra ensuite être utilisé dans l'algorithme de ligne de partage des eaux. À noter également l'interaction entre certains contours: un contour fort à une distance élevée d'un autre contour moins fort provoque un saut sur ce dernier dans l'image non régularisée. Ceci a pour conséquence un dédoublement de certains contours après régularisation. Hanbury & Marcotegui corrigent ce rebond avant application de la *lpe* pour avoir des contours de région de faible épaisseur.

gardant que les sauts les plus importants du résidu, réagit avec le personnage et crée une grande zone de distances importantes autour de celui-ci pour la transformation de l'image directe. Cette réaction n'existe plus dans l'image inverse, et le pigment fera en sorte qu'après régularisation, la distance restera faible dans cette partie de l'image. Nous constatons par contre qu'il est difficile de prédire et contrôler les interactions entre les différentes parties de l'image: quel mécanisme provoque les irrégularités dans le fond de la transformation directe ? Serait-ce le bord ou la texture du fond ?

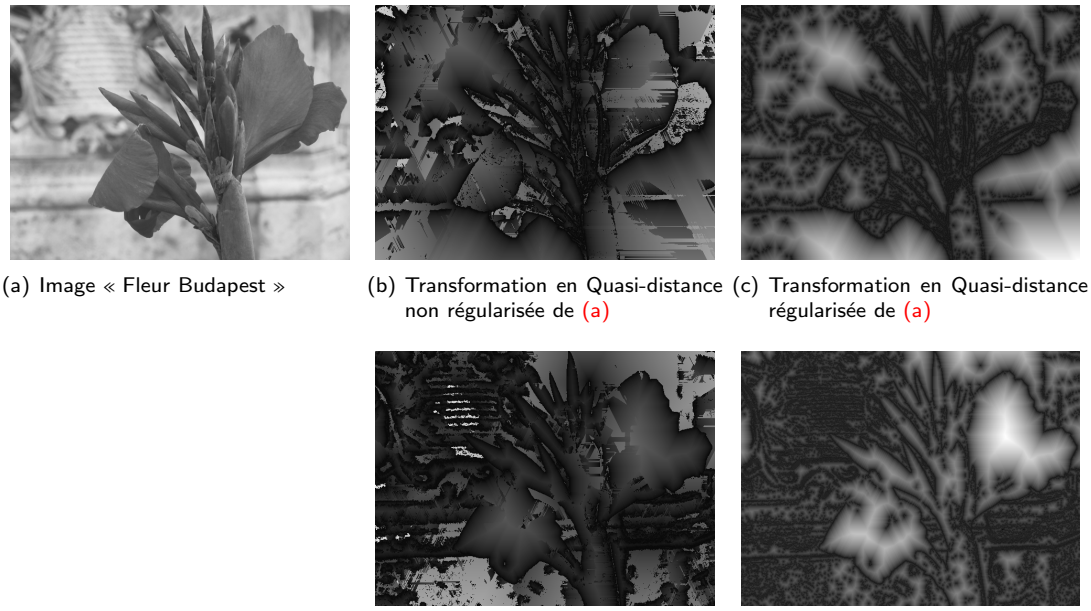
Sur l'image Retina 3.17, nous remarquons l'importance de la régularisation de la fonction distance. Avant régularisation, il est difficile de voir les vaisseaux alors que ceux-ci apparaissent très clairement après régularisation.

Enfin sur l'exemple « La danse des quatre bretonnes » de la figure 3.18, nous constatons que l'application directe de la transformation en Quasi-distance se heurte au problème des textures. Dans ce cas précis, que cela soit dans la transformation de l'image directe ou inverse, la fine texture de la



(a) Image originale (b) Quasi-distance régularisée de (a) (c) Quasi-distance régularisée de l'inverse de (a)

Fig. 3.14.: Utilisation de la quasi-distance pour la vidéosurveillance. (b) La quasi-distance produit de bons résultats lorsque les objets d'intérêt sont fortement contrasté par rapport à leur fond (ou leur voisinage). Cependant nous obtenons difficilement de bons résultats à la fois sur la quasi-distance de l'image et de son inverse ((c)).



(a) Image « Fleur Budapest »

(b) Transformation en Quasi-distance non régularisée de (a)

(c) Transformation en Quasi-distance régularisée de (a)

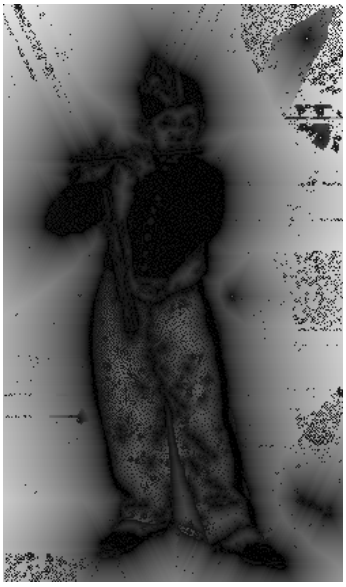
(d) Transformation en Quasi-distance non régularisée de l'inverse de (a)

(e) Transformation en Quasi-distance régularisée de l'inverse de (a)

Fig. 3.15.: Premier exemple de la transformation en Quasi-Distance pour l'image « Fleur Budapest » (dimensions : 648×486 pixels)

toile provoque des réactions, au niveau du résidu, qui restent locales. Ceci a pour conséquence une très faible distance générée sur la totalité de l'image, et ce malgré la présence d'éléments contrastants (personnages sombres sur fond clair).

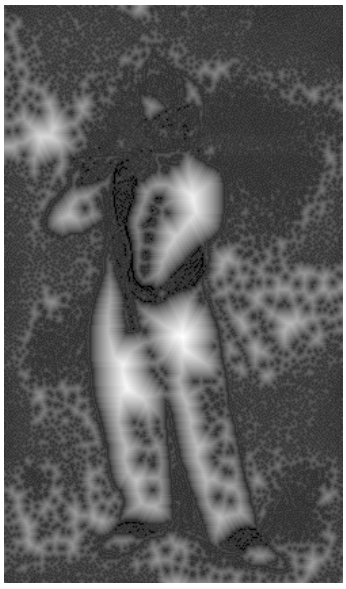
(a) Image « Fifer »



(b) Transformation en Quasi-distance non régularisée de (a)

(c) Transformation en Quasi-distance régularisée de (a)

(d) Transformation en Quasi-distance non régularisée de l'inverse de (a)



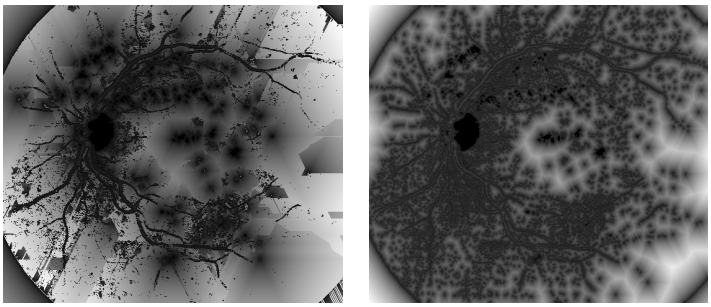
(e) Transformation en Quasi-distance régularisée de l'inverse de (a)

Fig. 3.16.: Deuxième exemple de la transformation en Quasi-Distance pour l'image « Fifer » (dimensions : 250×426 pixels)

(a) Image « Retina »

(b) Transformation en Quasi-distance non régularisée de (a)

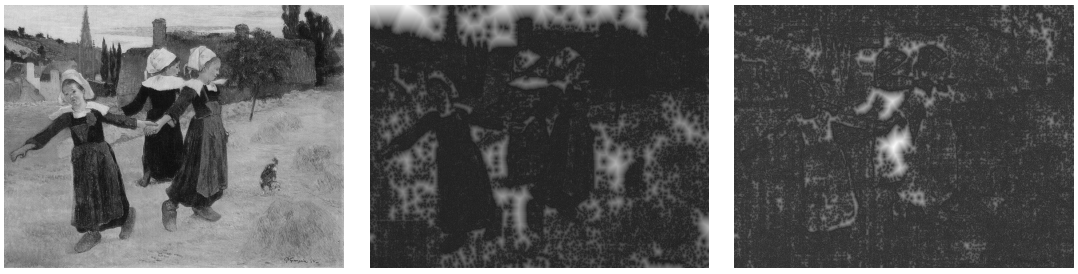
(c) Transformation en Quasi-distance régularisée de (a)



(d) Transformation en Quasi-distance non régularisée de l'inverse de (a)

(e) Transformation en Quasi-distance régularisée de l'inverse de (a)

Fig. 3.17.: Troisième exemple de la transformation en Quasi-Distance pour l'image « Retina » (dimensions : 548×480 pixels)



(a) Image « La danse des quatre bretonnes » de Gauguin

(b) Transformation en Quasi-distance régularisée de (a)

(c) Transformation en Quasi-distance régularisée de l'inverse de (a)

Fig. 3.18.: La transformation en Quasi-distance ne donne pas toujours des résultats appréciables. Ici, la fine texture de la toile « La danse des quatre bretonnes » perturbe la transformation et la fonction distance générée reste très faible sur presque la totalité de l'image. Pourtant le contraste entre les robes et le fond est assez important.

Nous cherchons à présent à améliorer les algorithmes existants en termes de temps de calcul. Pour cela nous allons nous intéresser au *noyau* de l'activité des primitives. Cela nous permettra de prouver le critère d'arrêt et également de voir quel type de redondance nous pouvons espérer entre deux itérations successives. Nous en déduirons une approche pour un nouvel algorithme.

Critère d'arrêt: Pour tout $i \in \mathbb{N}^*$, notons le noyau $Ker(r_i)$ l'ensemble suivant:

$$Ker(r_i) = \{a \in \mathbf{X} \,/\, r_i(a) = 0\}$$

Dans le cas de la quasi-distance, nous avons:

$$Ker(r_i) = \left\{a \in \mathbf{X} \,/\, (\epsilon^{(i)} - \epsilon^{(i+1)})(\mathbf{X})(a) = 0\right\} = \left\{a \in \mathbf{X} \,/\, \epsilon^{(i)}(\mathbf{X})(a) = \epsilon^{(i+1)}(\mathbf{X})(a)\right\}$$

Le couple de primitives n'est plus actif pour un indice $j > i$ si et seulement si le noyau est égal à $\mathbf{X}$ pour l'indice i, en effet:

$$\begin{array}{rcll} Ker(r_i) = \mathbf{X} & \Leftrightarrow & \forall a \in \mathbf{X}, r_i(a) = 0 & \\ & \Leftrightarrow & \forall a \in \mathbf{X}, \epsilon^{(i)}(\mathbf{X})(a) = \epsilon^{(i+1)}(\mathbf{X})(a) & \\ & \Leftrightarrow & \epsilon^{(i)}(\mathbf{X}) = \epsilon^{(i+1)}(\mathbf{X}) & \\ & \Leftrightarrow & \epsilon^{(j)}(\mathbf{X}) = \epsilon^{(i)}(\mathbf{X}) & , j > i \\ & \Leftrightarrow & r_j = 0 & , j > i \end{array}$$

Il est donc possible d'arrêter les itérations lorsque le résidu est nul partout - également équivalent à $\int_{\mathbf{X}} r_i = 0$ puisque le résidu est positif - ce qui est la condition utilisée par Beucher, mais ce n'est pas la seule puisqu'elle est équivalente à $\mathbf{X} \setminus Ker(r_i) = \varnothing$.

Est-il possible de déduire, lors de l'itération $i+1$, les éléments du noyau $Ker(r_{i+1})$ à partir des éléments de $Ker(r_i)$? Notons S l'élément structurant, et voyons l'intersection de ces deux ensembles:

$$\begin{array}{l} a \in Ker(r_{i+1}) \cap Ker(r_i) \\ \Rightarrow \left\{ \begin{array}{l} a \in Ker(r_i) \\ a \in Ker(r_{i+1}) \end{array} \right. \end{array}$$

$$\Rightarrow \left\{ \begin{array}{lll} \left(\epsilon^{(i)}(\mathbf{X}) \ominus S\right)(a) & = \epsilon^{(i)}(\mathbf{X})(a) & \\ \left(\epsilon^{(i+1)}(\mathbf{X}) \ominus S\right)(a) & = \epsilon^{(i+1)}(\mathbf{X})(a) & = \left(\epsilon^{(i)}(\mathbf{X}) \ominus S\right)(a) \end{array} \right.$$

d'où

$$\begin{array}{lll} a \in Ker(r_{i+1}) \cap Ker(r_i) & & \\ \Rightarrow \left(\epsilon^{(i)}(\mathbf{X}) \ominus S \ominus S\right)(a) & = \left(\epsilon^{(i)}(\mathbf{X}) \ominus S\right)(a) & = \epsilon^{(i)}(\mathbf{X})(a) \\ \Rightarrow \left(\epsilon^{(i)}(\mathbf{X}) \ominus S \circ S\right)(a) & = \left(\epsilon^{(i)}(\mathbf{X}) \circ S\right)(a) & = \left(\epsilon^{(i)}(\mathbf{X}) \oplus S\right)(a) \\ \Rightarrow \left(\epsilon^{(i)}(\mathbf{X}) \ominus S\right)(a) & = \left(\epsilon^{(i)}(\mathbf{X}) \oplus S\right)(a) & \end{array}$$

La deuxième implication est obtenue par la composition de $\oplus S$ à droite, la troisième provient du l'égalité suivante (A quelconque):

$$A \ominus S \circ S = A \circ S \ominus S = A \ominus S$$

La dernière égalité informe que le dilaté et l'érodé de l'image $\epsilon^{(i)}(\mathbf{X})$ au point a sont égaux. Si l'on considère l'élément structurant S plan - que l'on peut donc confondre avec un graphe de connexité - a est à la fois un minimum et un maximum du voisinage induit par S: l'image $\epsilon^{(i)}(\mathbf{X})$ est donc constante au voisinage de a. Ceci nous permet de dire que les éléments à l'intérieur des plateaux et non connexes au bord de ces plateaux sont invariants lors d'une application unique des primitives, ce qui va dans le sens de l'intuition. Les bords des plateaux mentionnés ne sont pas inclus dans ces invariants car il se peut qu'un plateau érode un autre plateau (et donc le bord varie). Nous n'avons pas réussi à extraire des informations supplémentaires.

Lors de l'application itérative des primitives à l'image, les minima locaux vont « creuser » l'image et étendre les plateaux autour de ces minima. Le nombre total de points à l'intérieur des plateaux ne peut qu'augmenter. En effet, deux cas de figure se présentent:

– soit un point n'appartient pas à un plateau issu d'un minimum local: dans ce cas il existe une itération j finie pour laquelle ce point sera érodé par un minimum.
– soit un point appartient déjà à un plateau issu d'un minimum: dans ce cas, soit il reste dans ce plateau, soit ce plateau est érodé par un plateau inférieur et ce point passe d'un plateau à un autre.

Dans les deux cas, le nombre de points contenus dans les plateaux augmente, et par connexité des plateaux, le nombre de points non connexes aux bords de ces plateaux augmente également.

Selon cette remarque, il parait donc judicieux de ne s'intéresser qu'aux points de $\mathbf{X} \setminus Ker(r_i)$ à chaque étape de l'itération, c'est à dire aux points susceptibles de varier lors de l'itération i. Il n'est pas aberrant de supposer que le nombre de ces points est susceptible de diminuer rapidement - puisque le $Ker(r_i)$ augmente - ce qui aura également pour conséquence de focaliser le traitement plus efficacement sur les points d'intérêt.

Nous allons à présent présenter l'algorithme issu de ces remarques. Comme précisé précédemment, il se compose de deux parties: le calcul des érodés successifs et la régularisation de la fonction distance. Il ne nous a pas apparu possible de combiner les deux de manière intéressante, sans créer un nombre important de redondances.

3.3.2. Algorithme pour le calcul des quasi-distances

Dans cette partie, nous présentons un nouvel algorithme pour le calcul des quasi-distances. Par rapport à l'algorithme original proposé par Beucher, celui-ci a l'avantage d'être moins coûteux en termes de calcul, en ne traitant qu'un nombre réduit de points à chaque pas d'itération. Nous décrivons les algorithmes dans une première partie, puis nous évaluons quantitativement les améliorations.

Calcul de la famille des érodés - algorithme 3.5

L'algorithme classique de Beucher traite tous les points à chaque pas d'itération. Cette observation triviale est le fil conducteur de nos développements. Selon ce qui précède, le fait de ne s'intéresser qu'aux points appartenant au complémentaire du noyau des primitives est suffisant pour le calcul de la quasi-distance. Rappelons toutefois que nous ne nous intéressons qu'aux quasi-distances définies par un élément structurant plan.

La construction du complémentaire du noyau est triviale: il s'agit des points pouvant être érodés, ie. des points possédant un voisin de hauteur inférieure.

$$\mathbf{X} \setminus Ker(r_i) = \{x \in \mathbf{X} \,/\, \exists\, v \in \mathcal{N}_x, v < x\}$$

La première partie de l'algorithme - lignes 4 à 15 - construit précisément l'ensemble $\mathbf{X} \setminus Ker(r_0)$, mais n'effectue aucun traitement. Cet ensemble est stocké dans une file d'attente normale, noté f_1. Le traitement est effectué dans une image de travail W1. Nous aurons besoin d'une deuxième image de travail W2 dans laquelle nous stockerons temporairement l'érosion de W1.

La deuxième partie de l'algorithme effectue le calcul de l'érosion de $\mathbf{X} \setminus Ker(r_i)$, et propage cet ensemble aux voisinages immédiats. Nous avons en effet vu que seuls les points non connexes aux bords de $Ker(r_i)$ sont invariants, et donc par dualité les points connexes à $\mathbf{X} \setminus Ker(r_i)$ sont variants.

L'érosion ainsi que le test des conditions de la quasi-distance sont effectuées aux lignes 19 à 28. L'érosion est effectuée sur les points de W1, et le résultat est placé dans W2. Ce calcul est fait pour tous les points de la file f_1 et donc pour les points de $\mathbf{X} \setminus Ker(r_i)$. Le résidu « R » et l'indicatrice « Q » sont également mis à jour lors de cette étape.

Enfin la deuxième partie de la propagation recherche les points de $\mathbf{X} \setminus Ker(r_{i+1})$. Nous savons que ces points sont connexes à $\mathbf{X} \setminus Ker(r_i)$, la recherche est donc effectuée au voisinage des points de f_1, et dans l'image nouvellement érodée W2 (égale à $\epsilon^{(i+1)}$). Les nouveaux points sont placés dans f_2. Les critères de recherche sont les mêmes que lors de l'initialisation. Puisqu'il est possible avec ces seules conditions de mettre plusieurs fois un même point dans f_2, une condition supplémentaire garantissant

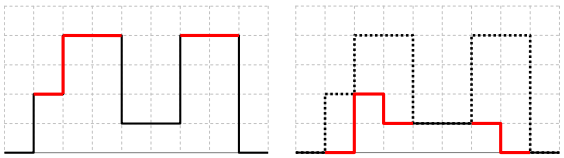
l'unicité des nouveaux points dans f_2 est nécessaire. Cette condition utilise une troisième image de travail C. C'est également pendant cette dernière boucle (ligne 31) que les points érodés de W2 (égale à $\epsilon^{(i+1)}$) sont placés dans W1 (égale à $\epsilon^{(i)}$), de manière à boucler correctement la propagation. Les files f_1 et f_2 sont ensuite échangées. La condition d'arrêt $\mathbf{X} \setminus Ker(r_i) = \varnothing$ est alors équivalente au test sur f_1.

Les étapes de l'algorithme sont illustrées sur la figure 3.19.

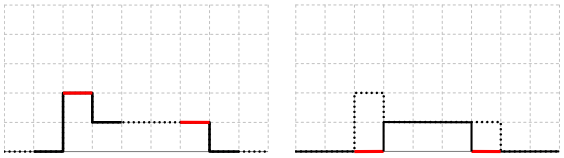
Algorithme 3.5 : Calcul de la famille des érodés successifs

Data : $\mathcal{I}$
▷$\mathcal{I}$: Image d'entrée scalaire
Result : Q, R
▷ Respectivement les images de distance et de résidu

▶ **Initialisation**
C ← non-traité
W1, W2 ← $\mathcal{I}$
R, Q ← 0
$f_1, f_2 \leftarrow \varnothing$
forall $p \in$ *W1* **do**
  forall $v \in \mathcal{N}_p \setminus p$ **do**
    if *W1*$(v <$*W1*(p) **then**
      $f_1 \leftarrow$ ajouter p
      break
  end
end
indicateur = 1
▶ **Propagation**
while $f_1 \neq \varnothing$ **do**
  Érosion
  forall $p \in f_1$ **do**
    C(p) = non-traité
    W2$(p) = \wedge\mathcal{N}_p$(W1)
    residu = W1(p) - W2(p)
    if *residu* $\geq$ *R*(p) **then**
      if *residu* $\neq$ *R*(p) **then** R$(p) \leftarrow$ residu
      Q$(p) \leftarrow$ indicateur
  end
  ▶ *Nouveaux points pouvant recevoir une érosion dans les voisins des points érodés*
  forall $p \in f_1$ **do**
    W1(p) = W2(p)
    forall $v \in \mathcal{N}_p \setminus p$ **do**
      if *W2*$(v >$*W2*(p) *et (*C$(v) \neq$ *dans-la-file)* **then**
        $f_2 \leftarrow$ ajouter v
        C(v) = dans-la-file
    end
  end
  $f_1 \leftarrow f_2$
  $f_2 \leftarrow \varnothing$
  indicateur = indicateur + 1
end



(a) Détection du sous-ensemble d'intérêt X_1 (b) $\epsilon(X_1)$



(c) Détection du sous-ensemble d'intérêt X_2 (d) $\epsilon(X_2)$

Fig. 3.19.: Calcul de la famille d'érosions successives. Seuls les points rouges sont présents dans la file d'attente : ils sont représentatifs de l'activité du couple de primitives.

Remarque : L'approche par files d'attente souffre néanmoins d'un inconvénient majeur: l'action de centrer le voisinage de recherche fait que nous ne pouvons pas utiliser la valeur de points déjà lus (qui seraient par exemple dans un voisinage connexe). Après dépilement d'un point de la file (qui représente le centre d'un voisinage), une lecture sur la totalité du voisinage s'impose. Il existe par ailleurs des approches de calcul d'érosion qui minimisent précisément le nombre de points à relire dans la mémoire lors d'un déplacement du voisinage vers un voisinage voisin. L'approche par files d'attente entraîne nécessairement du « gaspillage » de lecture dans la mémoire.
Une idée serait donc de combiner les approches rapides de calcul d'érosion, de manière à tirer le profit maximal jusqu'à ce que ce le nombre de points érodants soit suffisamment petit pour qu'une approche par files d'attente soit plus intéressante.

Régularisation de la fonction distance - algorithme 3.6

Nous présentons également un algorithme de régularisation des « distances perchées ». Bien que la détection des points ne satisfaisant pas la contrainte lipchitzienne soit triviale, il est difficile d'effectuer cette détection lors du calcul de la famille des érodés sans introduire de la redondance. Le test lipchitzien est effectivement simple lors d'une itération, mais complexe lorsque la totalité de la famille des érodés est considérée.

Cette remarque nous conduit à régulariser la fonction de distance après le calcul de la famille des érodés, sans chercher à transmettre des informations du premier au deuxième algorithme (lieu des points de conflit notamment). L'algorithme 3.6 proposé utilise une approche à base de files d'attente hiérarchiques. Il est composé de deux parties: la première est une initialisation des structures sur les ensembles d'intérêt, qui sont ici les points ne vérifiant pas la contrainte lipchitzienne. La deuxième partie propage les points nouvellement régularisés aux voisins de distance plus importante et violant également la contrainte lipchitzienne.

Notons NL les points « non lipschitziens »: ils sont donnés par l'équation 3.8 (page 71). L'initialisation présente ici deux étapes: la première marque les points de NL mais ne les traite pas directement et ne les introduit pas dans la file de traitement. Le marquage s'effectue dans une image de travail C. Nous avons choisi de les marquer comme étant « dans−la−file », il faudra comprendre ici « potentiellement » dans une file d'attente.

La deuxième étape de l'initialisation est la plus importante; elle sépare l'ensemble NL en deux sous-ensembles: les points connexes au bord de NL et les autres. La raison de ce choix est la suivante: les

points connexes au bord sont ceux ayant au moins un voisin régulier, au sens d'un voisinage unitaire. L'hypothèse ici est que ce voisin soit un appui pour la régularisation de la distance: les distances déduites de ce point d'appui seront correctes si ce dernier ne nécessite pas de régularisation à une étape ultérieure, ce qui est une condition difficile à déterminer lorsqu'on ne s'intéresse qu'aux voisinages unitaires. Il est d'ailleurs inutile de mettre un point contenu entièrement dans NL (ie. non connexe à son bord) dans la file, car il n'est pas possible de déduire sa distance - corrigée - à partir de ses voisins - car nécessitant eux-mêmes correction -. Si un tel point entre dans la file de traitement, il devra alors être traité plusieurs fois. Cette séparation est également motivée par la structure de files d'attente hiérarchiques, dont les performances sont fortement dépendantes du nombre de clefs qu'elle contient (ici la clef correspond à une valeur de la distance) .

Ces points sont ensuite placés dans la structure de files d'attente hiérarchiques, à une priorité correspondante à la hauteur qu'ils doivent avoir selon la contrainte lipchitzienne : cette valeur est l'infimum des voisins augmentée de 1. Enfin, ces points sont également régularisés pendant cette étape.

La deuxième étape de l'algorithme propage la régularisation sur les points de NL, en considérant des valeurs de distance croissantes. Puisque l'ensemble des points de distance inférieure à la distance courante p_1 sont réguliers, il est inutile de revenir sur ces points. L'ensemble des nouveaux points non lipchitziens sont voisins des points nouvellement régularisés, ce qui assure le traitement des l'ensemble des points de NL de manière itérative. Il s'agit ensuite d'une propagation tout à fait classique.

Les étapes de l'algorithme sont illustrées sur la figure 3.20.

Évaluation de l'amélioration

Nous allons à présent évaluer les performances de ces deux nouveaux algorithmes, comparativement à l'approche classique de Beucher. Plusieurs points attirent notre attention: nous souhaitons particulièrement montrer le fait que $\sharp(\mathbf{X} \setminus Ker(r_i))$ décroît rapidement, ce qui justifierait notre approche. Ceci se fait simplement en traçant la taille de la file d'attente f_1 de l'algorithme 3.5. De même, nous souhaitons également montrer que le nombre de points traités à chaque pas d'itération dans l'algorithme de régularisation 3.6 reste faible, et justifie encore notre approche par rapport à celle considérant la totalité de l'image. Nous montrerons enfin les gains en termes de temps d'exécution sur une machine de bureau standard.

Taille de la file d'attente: La figure 3.21 trace les tailles des files d'attente dans les deux algorithmes, les transformations sont présentées en figure 3.15. Pour la *fah* de la régularisation, nous considérons la totalité des points de toutes les priorités, mais il faut tenir compte du fait que seule une priorité est traitée à chaque étape.

À la vue de la figure 3.21 (calcul sur l'image « Fleur Budapest » de la transformation page 75), le nombre de points dans chacune des files d'attente présentent une décroissance logarithmique. Ceci se confirme pour la transformation des autres images d'exemple, données en figure 3.22 et 3.23.

Une autre remarque importante réside dans le fait que le nombre de points dans f_1 est toujours inférieur au nombre de points dans l'image. En traitant moins de points, on peut aisément envisager des améliorations en termes de temps de calcul, ce que nous allons voir à présent.

Temps de calcul : Nous présentons à présent l'amélioration en termes de temps d'exécution de nos algorithmes [(ix)]. Pour cela, nous avons calculé plusieurs graphiques.

Le graphique 3.24 reprend les temps associés à quelques images type. La légende est la suivante: en vert sont représentés les temps associés à la transformation pour les nouveaux algorithmes, en violet les temps associés à la transformation selon l'approche classique. Pour chacune de ces couleurs, une barre correspond à la transformation de l'image directe, l'autre à celle de l'image inverse. Les temps présentés sont la moyenne de dix réalisations, pour éviter l'influence du système d'exploitation.

[(ix)] Tous les temps ont été mesurés sur un ordinateur de bureau, de type Pentium 4 double cache, à 2.8Ghz avec 512Mo de mémoire, et un bus cadencé à 400MHz

Algorithme 3.6 : Régularisation des distances perchées

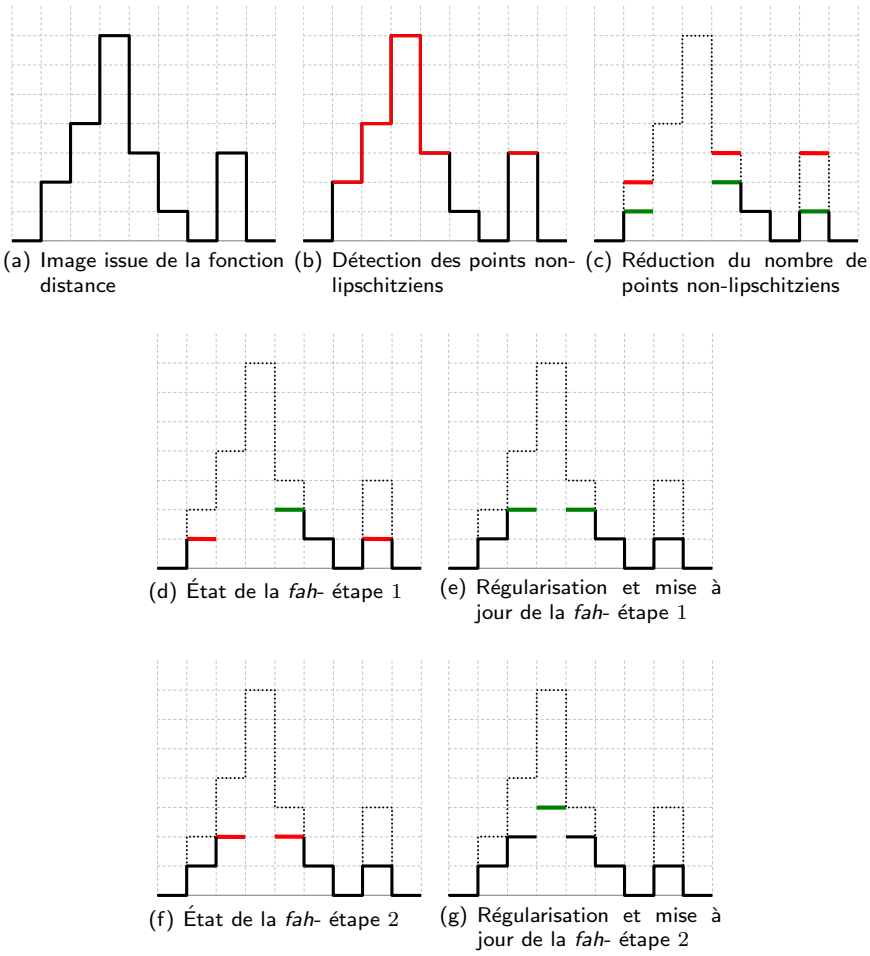
```
   Data : Q
1    ▷ Image issue de l'algorithme 3.5 de quasi-distance non régularisée
   Result : D
2    ▷ Quasi-distance régularisée
3  D ← Q
4  C ← non-traité
5  fah ← ∅
6  forall p ∈ D do
7      forall v ∈ N_p \ p do
8          if D(v) >D(p) + 1 then C(v) = dans-la-file
9      end
10 end
11 forall p ∈ D do
12     if C(p) = dans-la-file then
13         forall v ∈ N_p \ p do
14             val-voisin = Q(v) + 1
15             if C(v) = non-traité et D(p) > val-voisin then
16                 D(p) = val-voisin
17                 fah ← ajouter p à la priorité val-voisin
18
19         end
20
21 end
22 while fah ≠ ∅ do
23     p1 ← priorité la plus grande de fah
24     p2 ← p1 + 1
25     forall p ∈ fah(p1) do
26         forall v ∈ N_p \ p do
27             if D(v) > p2 then
28                 fah ← ajouter v à la priorité p2
29                 D(v) = p2
30
31         end
32     end
33     fah ← suppression des points de la priorité p1
34 end
```

L'amélioration en secondes est très fortement dépendante du contenu de l'image. Selon ce graphique, la différence est fortement remarquable sur les images de dimension importante, elle n'est pourtant pas minime sur les images de faible dimension. Les images « Road », « Pepper », « Foreman » dont les dimensions sont entre le QCIF et 256^2 présentent des différences de temps de calcul de $2.8s$ à $6.5s$.

Puisque les images sont très différentes en dimension, nous avons ramené ces temps à un rapport logarithmique par pixel. Ces rapports sont présentés sur la figure 3.25.

Nous constatons une nette amélioration des temps de calcul. Premièrement, les logarithmes des rapports sont tous positifs, ce qui confirme l'amélioration des temps d'exécution. Ensuite, il est courant de voir une valeur de 0.8. Rappelons qu'une valeur de 1 signifie que l'on passe dix fois moins de temps sur chaque pixel dans le nouvel algorithme par rapport au classique: $10^{0.8} \approx 6.3$.

Les améliorations sont encore plus importantes pour l'algorithme de régularisation. Les deux graphiques similaires aux précédentes sont représentés en figure 3.26 et 3.27. Le plus mauvais rapport est de 0.68, soit un rapport de temps de $10^{0.68} \approx 4.8$ par pixel. Le meilleur est de 1.4, soit un rapport de $10^{1.4} \approx 25.1$. Ces grandes différences s'expliquent essentiellement par le contenu très différent des



(a) Image issue de la fonction distance (b) Détection des points non-lipschitziens (c) Réduction du nombre de points non-lipschitziens

(d) État de la *fah*- étape 1 (e) Régularisation et mise à jour de la *fah*- étape 1

(f) État de la *fah*- étape 2 (g) Régularisation et mise à jour de la *fah*- étape 2

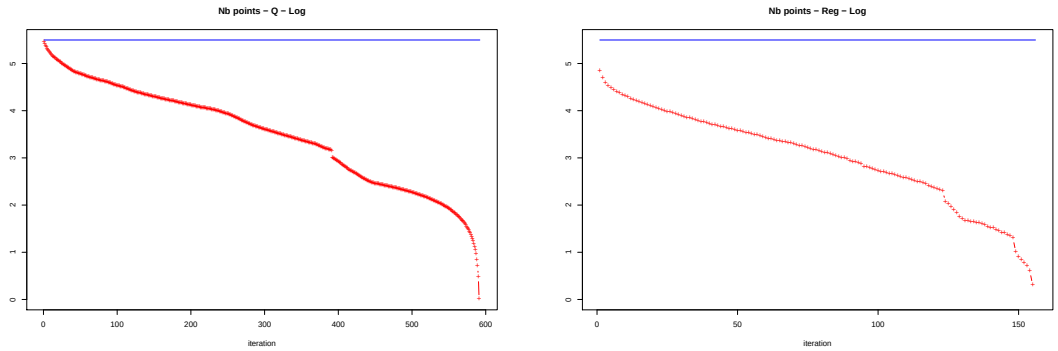
Fig. 3.20.: Régularisation de la fonction distance.

images. D'une part, le rapport est sensiblement meilleur sur les images de grande dimension, ce qui est normal puisque sur les images de petites tailles, le surcoût lié au traitement de la totalité des points n'est pas aussi important [(×)]. Ensuite plusieurs fronts de régularisation peuvent se joindre dans les grandes images, alors que ce phénomène risque d'être plus rare dans les petites images (deux fronts qui se joignent réduisent le surcoût lié au parcours de voisinage).

Amélioration: Nous nous sommes rapidement demandé s'il n'était pas possible d'améliorer les résultats en adoptant un algorithme hybride, exploitant les avantages des deux approches décrites. Nous avons mentionné le fait que l'approche classique ne présente pas de surcoût lié à l'utilisation de files d'attente. De plus, il est possible d'exploiter la redondance entre voisinages connexes pour le calcul des érosions, alors qu'une approche par files d'attente nous fait perdre la connexité de voisinage: des hypothèses simples évitant la lecture en mémoire de points de voisinage connexes ne peuvent plus être émises. Nous n'avons malheureusement pas pu essayer l'implémentation optimisée de l'érosion dans ce sens, et les comparaisons que nous allons présenter utilisent toutes deux une structure de voisinage similaire, n'exploitant pas la redondance de voisinage connexe.

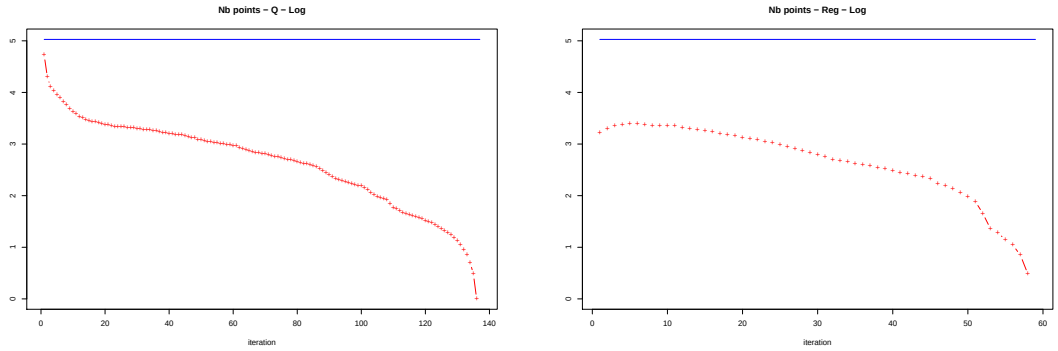
Le principe de l'algorithme hybride est le suivant: pendant les k premières itérations, l'approche

[(×)] moins de points à traiter: le rapport entre l'inutile et l'utile est moins flagrant.



(a) $\log_{10}(\sharp(\mathbf{X} \setminus Ker(r_i)))$ en fonction de l'itération, lors du calcul de la famille des érodés (b) $\log_{10}(\sharp fah)$ en fonction de l'itération, lors de la régularisation de la fonction distance

Fig. 3.21.: Tailles des files d'attente dans les deux algorithmes proposés pour l'image « Fleur Budapest ». Nous remarquons une décroissance linéaire du logarithme du cardinal, ce qui est un indicateur pertinent des performances globales de l'approche. La ligne bleue au dessus représente le nombre total de points de l'image.



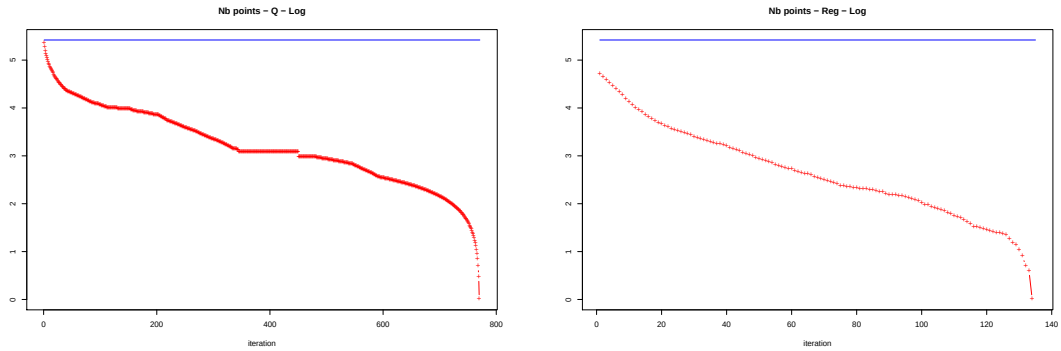
(a) $\log_{10}(\sharp(\mathbf{X} \setminus Ker(r_i)))$ en fonction de l'itération, lors du calcul de la famille des érodés (b) $\log_{10}(\sharp fah)$ en fonction de l'itération, lors de la régularisation de la fonction distance

Fig. 3.22.: Taille des files d'attente dans les deux algorithmes proposés pour l'image « Fifer ». La légende est la même que sur la figure 3.21.

classique est utilisée. L'algorithme bascule ensuite sur l'approche nouvelle, à base de files d'attente. Cet algorithme est tellement simple qu'il est absolument inutile de le décrire d'avantage. Il semble par contre intéressant d'évaluer les temps de calcul de la transformée en fonction du paramètre k qui fixe le nombre d'itérations de l'approche classique. Il est évident que k trop grand détériore les temps globaux compte tenu des améliorations décrites: nous nous sommes donc restreint aux 40 premières itérations.

Les résultats sont présentés sur la figure 3.28 (image directe) et 3.29 (image inverse). L'axe des abscisses présente l'index à partir duquel nous basculons sur l'algorithme que nous venons de présenter. Les temps sont encore une fois calculés pour une moyenne de 10 réalisations. Ce qu'il est intéressant d'observer sur ce genre de graphique est la présence ou l'absence de point d'inflexion, qui serait un indicateur de temps optimal pour le traitement. À la vue des profils, il semble qu'un point d'inflexion existe autour de $k = 8$ itérations, bien que l'inflexion ne soit pas partagée par toutes les images. Ces résultats rejoignent ceux concernant la taille de la file d'attente f_1: il apparaît sur un certain nombre d'images une décroissance très importante du nombre de points dans la file d'attente au cours des quelques premières itérations. Pour un nombre faible de points dans la file, la nouvelle approche est toujours plus performante. Ce qui explique ce point d'inflexion assez proche de l'origine.

Enfin en dernier point, nous souhaitons évaluer à quel gain nous pouvons nous attendre par l'utilisation de l'approche hybride. Les figures 3.30 et 3.31 présentent les gains maximaux. La légende est



(a) $\log_{10}(\sharp(\mathbf{X} \setminus Ker(r_i)))$ en fonction de l'itération, lors du calcul de la famille des érodés (b) $\log_{10}(\sharp fah)$ en fonction de l'itération, lors de la régularisation de la fonction distance

Fig. 3.23.: Taille des files d'attente dans les deux algorithmes proposés pour l'image « Retina ». La légende est la même que sur la figure 3.21.

la suivante: les images sont représentées en abscisse, chaque barre représente le logarithme du rapport temps maximal sur temps minimal. Cette mesure du gain doit toutefois être prise avec précaution: elle a tendance à privilégier l'approche hybride - au détriment de l'approche par files d'attente - car le temps maximal est donné dans la plupart des cas lorsque le nombre d'érosions initiales est grand (c'est à dire lorsque l'apport de l'approche par files d'attente est pratiquement nul).

Les gains les plus importants sont de l'ordre de $10^{0.8} \approx 6.3$, ce qui est assez important. Ce gains se manifestent sur des images « Salt », « Chambre » et « Fifer ». L'image « Salt » représente quelques petits grains (cf. figure 3.11 page 71), et l'image disparaît après un nombre faible d'érosions. Le résultat n'est donc pas très significatif pour cette image. L'image « Chambre » présente un pigment très fin: le résultat de la quasi-distance après régularisation présente par ailleurs des distances très faibles, représentatives des interactions de voisinage. Le fait d'appliquer quelques érosions « classiques » avant l'approche par files d'attente réduit considérablement le nombre de pigments, et indirectement le nombre de points qui seront ensuite placés dans la file d'attente de notre algorithme. C'est un cas d'exemple pour lequel appliquer un algorithme tirant profit de la redondance de voisinage, préalablement à l'utilisation de l'algorithme par files d'attente, s'avère très pertinent. L'image « Fifer » (cf. figure 3.16 page 76) présente le même type de pigment, ce qui se traduit également dans le graphique par un gain important. Pour les autres images par contre, le gain existe, mais n'est pas tellement significatif. Nous pouvons simplement dire que l'application pour quelques itérations de l'algorithme classique améliore un peu les temps de calcul globaux, sans toutefois apporter une grande amélioration.

Conclusion: Comme nous l'avons mentionné, les comparaisons affichées ici n'utilisent pas d'implémentation rapide des érosions, ce qui est un défaut non négligeable. Nous nous attendons en effet à ce que le point d'inflexion de l'approche hybride soit reporté à un nombre d'itérations beaucoup plus élevé.

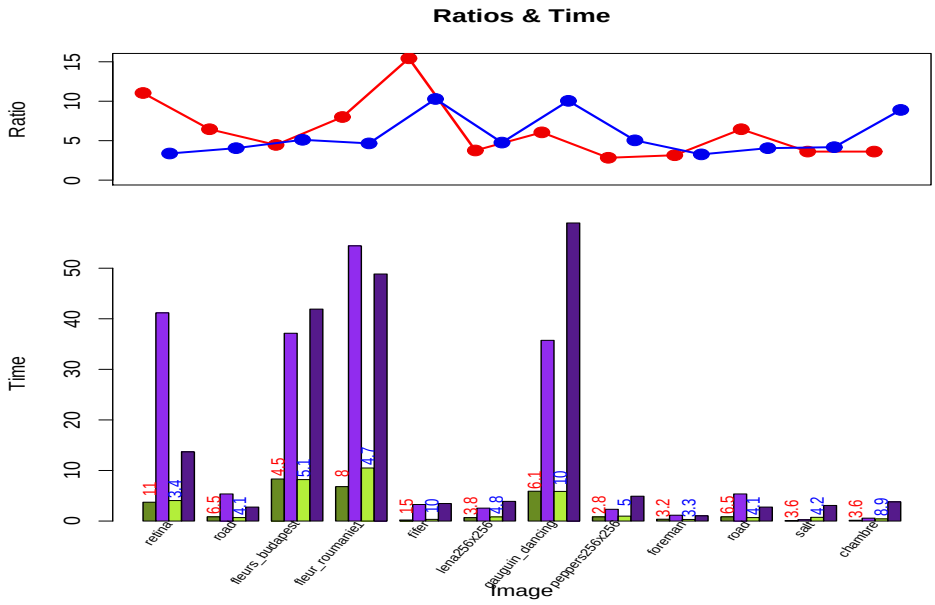


Fig. 3.24.: Temps de calcul des érodés successifs. Respectivement en vert et violet: les temps correspondant au nouvel algorithme proposé et ceux correspondant à l'algorithme classique. La différence en secondes entre les deux algorithmes est reportée à l'extrémité des barres et reporté sur la partie haute du graphique.

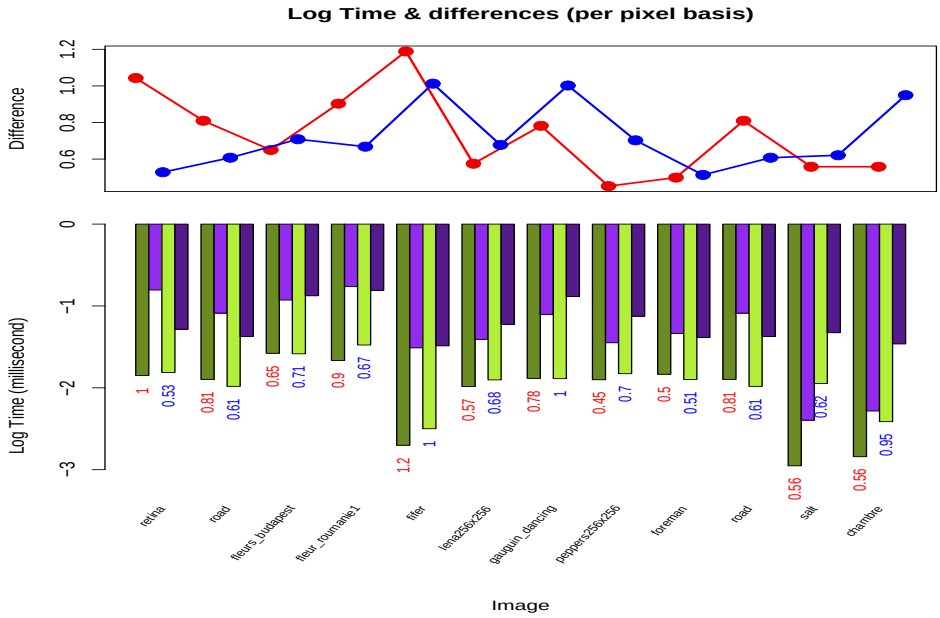


Fig. 3.25.: Logarithme du rapport des temps de calcul des érodés successifs, entre l'algorithme de Beucher et celui par files d'attente, par pixel. La légende est la même que pour le graphique 3.24. Le logarithme du rapport est reporté à l'extrémité des barres et reporté sur la partie haute du graphique.

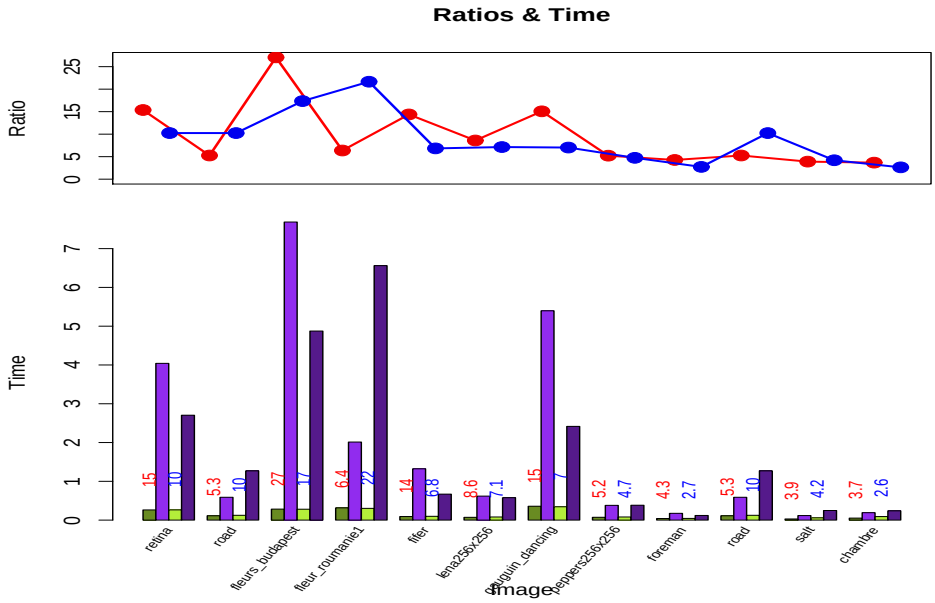


Fig. 3.26.: Temps de calcul de la régularisation. La légende est la même que pour le graphique 3.24.

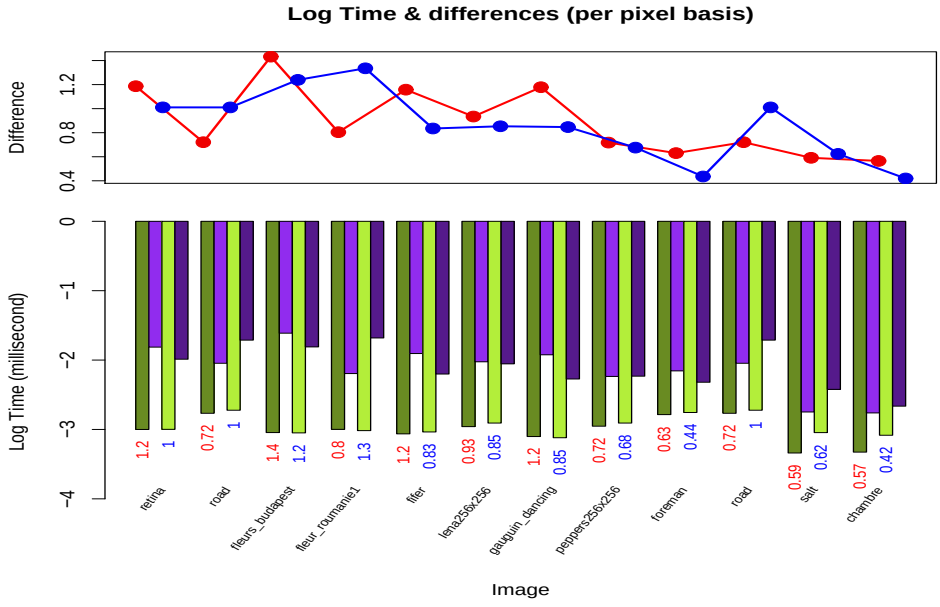


Fig. 3.27.: Logarithme du rapport des temps de calcul de la régularisation, entre l'algorithme de Beucher et celui par files d'attente hiérarchiques, par pixel. La légende est la même que pour le graphique 3.24.

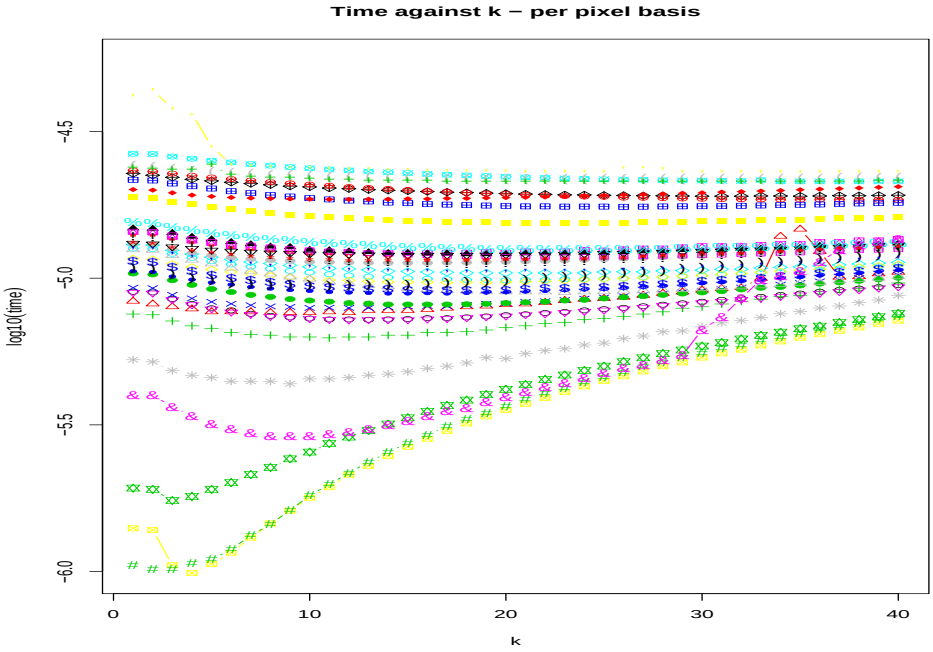


Fig. 3.28.: Temps de calcul de l'approche hybride en fonction de k. Chaque courbe représente les temps moyens mesurés sur une image. Nous remarquons un point d'inflexion global autour de 8 itérations. Les temps sont donnés par pixel en coordonnée logarithmique.

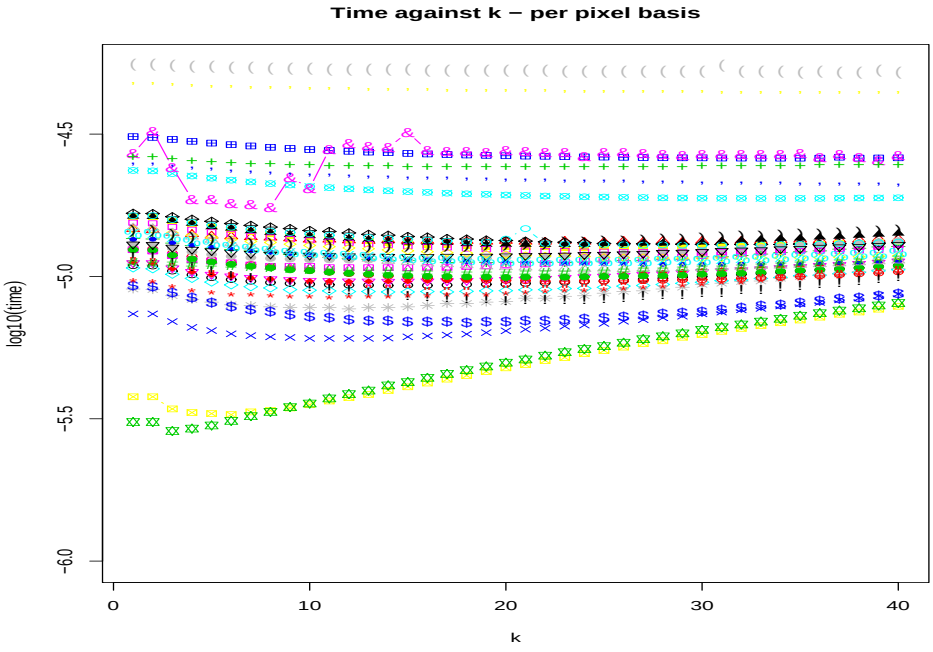


Fig. 3.29.: Temps de calcul de l'approche hybride en fonction de k - images inverses. Les temps sont donnés par pixel en coordonnée logarithmique.

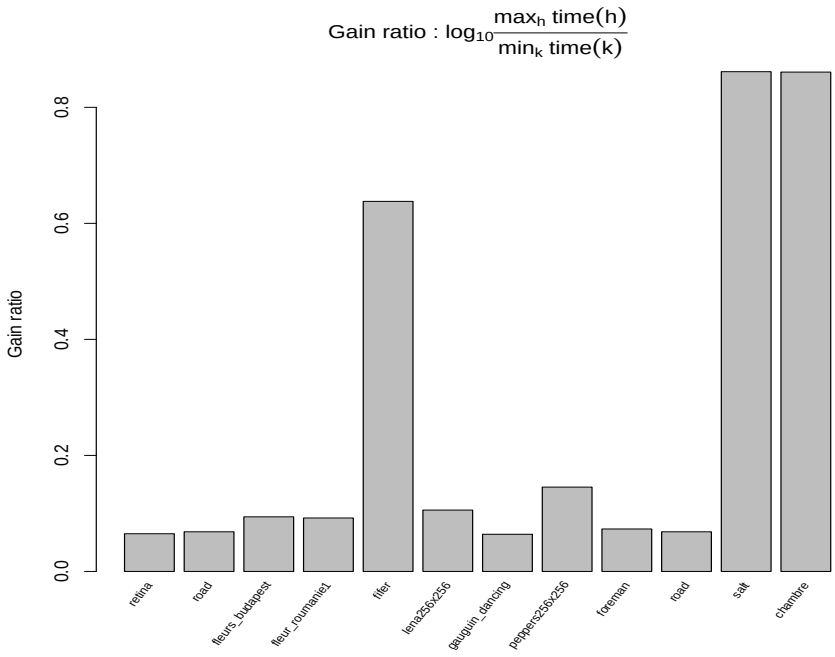


Fig. 3.30.: Gains maximaux de l'approche hybride. Les images sont représentées en abscisse. Chaque barre représente le logarithme du rapport du temps maximal sur le temps minimal de calcul; c'est à dire le gain maximal le plus pessimiste de l'approche hybride. Nous remarquons que les gains ne sont pas négligeables puisqu'il peuvent aller jusqu'à un facteur proche de $10^{0.8} \approx 6.3$.

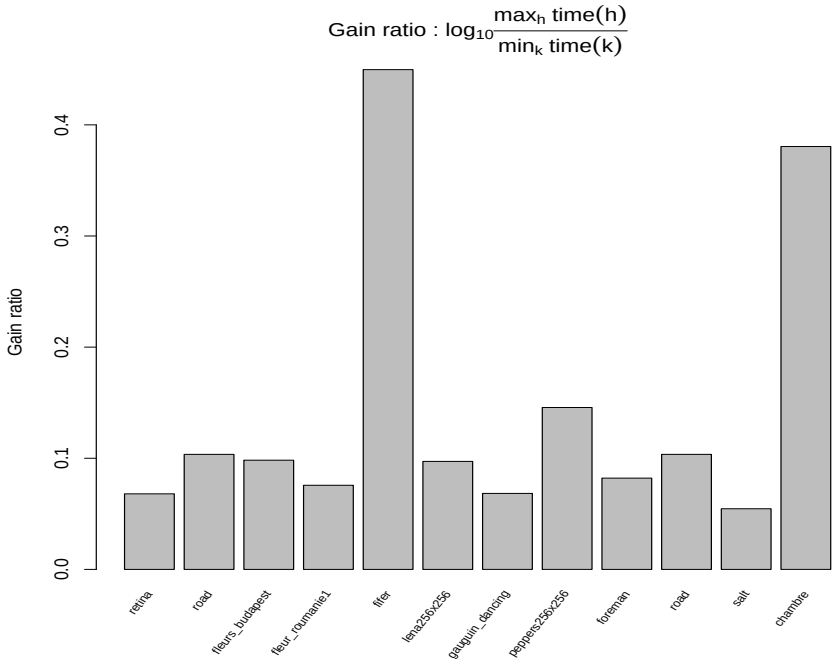


Fig. 3.31.: Gains maximaux de l'approche hybride - images inverses. Voir 3.30 pour la légende.

3.3.3. Extension des opérations résiduelles à la couleur

L'extension numérique des opérateurs résiduels a donné à ces derniers une dimension dont le potentiel n'a pas encore été exploité en totalité. Ainsi les quasi-distances, que nous avons étudiées en détail, ont acquis une précision meilleure que les distances morphologiques classiques [(xi)] Par simple analogie à cette démarche, il semble naturel de penser que l'application de ces opérateurs sur des espaces plus riches comme les espaces couleurs, permettrait d'améliorer davantage la précision d'un tel algorithme.

Il peut paraître hâtif de s'aventurer dans une telle extension alors que le potentiel de ces opérateurs est encore à peine discernable. Cette démarche s'inscrit cependant dans la démarche plus générale de ce manuscrit, à savoir abstraire les notions au possible de manière à tirer profit des outils informatiques présentés au chapitre 2. Comme nous allons le voir dans la suite, la transposition des notions acquises du cadre numérique vers le cadre couleur est presque immédiate.

Le passage de la définition des opérateurs résiduels du binaire vers le numérique est intéressant dans la mesure où il répond à des interrogations naturelles, comme le choix de l'indice de résidu définitif, le choix du résidu lors de conflit, etc..

Il est parfaitement envisageable de se servir du même schéma d'extension lorsque qu'il s'agit de définir les opérateurs résiduels sur des domaines multidimensionnels. Dans cette partie, nous allons donc étudier les moyens dont nous disposons pour appliquer les opérations résiduelles à la couleur. À cette fin, nous allons reprendre les notions intervenant dans la définition des opérations résiduelles, puis proposer ces mêmes notions dans les espaces multidimensionnels.

Nous avons vu que la définition des primitives des opérateurs résiduels impliquent une structure d'ordre sur l'espace de travail. Sur ce point, nous avons introduit les ordres lexicographiques (cf. 2.10 page 26) permettant d'étendre très simplement une structure d'ordre existante à des espaces produits.

Les ordres lexicographiques sont des candidats de choix dans l'extension des ordres naturels sur les espaces multidimensionnels. Nous nous servirons de ces ordres dans ce qui suit.

En ce qui concerne l'opérateur de résidu, l'opérateur de différence intervenant dans le cadre numérique est utilisé conjointement à l'opérateur de valeur absolue: il s'agit donc d'un opérateur de distance de type d_1. Certains espaces couleur définissent également des fonctions de distance. À l'instar des relations d'ordre lexicographiques, les distances couleur seront un moyen de définir des transformations résiduelles dans les espaces couleur.

Plusieurs schémas d'application sont possibles à partir de ces simples outils. Nous allons dans ce qui suit détailler ces outils, les combinaisons possibles ainsi que quelques résultats en prenant comme support la quasi-distance.

3.3.3.1. Transformations résiduelles lexicographiques

Selon les équations générales des transformations résiduelles (équations 3.5 et 3.6 page 68), la structure du treillis utilisé, c'est à dire les opérateurs $\underline{\vee}$ et $\underline{\wedge}$ ainsi que l'ensemble de définition, n'interviennent pas dans la définition de Θ. La définition de l'opérateur Θ reste parfaitement générale.

Toutefois, lors de la définition de la transformation résiduelle, la structure du treillis intervient sur les deux aspects suivants:

- choix des primitives et des fonctions de résidu (le choix de l'un est intrinsèquement lié à l'autre).
- choix du quantile du résidu pour l'indicatrice.

Rappelons que, dans le cas des quasi-distances, le quantile était choisi comme étant le supremum sur l'ensemble des résidus (dans le cas où ce supremum n'était pas unique, seule la dernière occurrence était sélectionnée).

D'après les développements généraux que nous avons décrits jusqu'ici, il n'y a techniquement aucune contrainte dans l'extension couleur des opérations résiduelles décrites ici. Nous ne nous attarderons donc

[(xi)] En effet, bien que l'indicatrice de la quasi-distance n'est pas un sur-ensemble de la distance morphologique classique (à cause du choix du résidu), les algorithmes se comportent néanmoins de manière similaire lorsqu'ils sont restreints à des images binaires.

pas à décrire les techniques mises en œuvre pour l'implémentation des treillis lexicographiques: cela dérive naturellement des développements du chapitre 2 (page 25).

Nous voyons rapidement une extension simple des transformations résiduelles à un cadre plus général encore, à travers l'utilisation de fonction distance sur l'espace $\mathbf{F}$ des valeurs. Les espaces couleur sont un exemple intuitif dont nous allons nous servir. Une présentation des espaces couleurs est donnée en annexe A. Nous introduirons plus en détail, dans la partie §4.2.1 (page 105) les différentes métriques couleur. Supposons pour l'instant l'existence de telles métriques.

Ordres lexicographiques : Partons d'un espace $\mathbf{F}$. Un treillis sur $\mathbf{F}$ nécessite une structure d'ordre donné par la relation $\preceq_{\mathbf{F}}$. Si $\langle \mathcal{L}_{\mathbf{F}}, \preceq_{\mathbf{F}} \rangle$ est un treillis, alors $\langle \mathcal{L}_{\mathbf{F}}, \succeq_{\mathbf{F}} \rangle$ est le treillis dual. Ainsi à partir d'un espace et d'un ordre, nous avons deux treillis possibles. De même, pour un espace produit $\mathbf{F} = \mathbf{F}_1 \times \mathbf{F}_2 \times \ldots \mathbf{F}_n$ nous avons $N_{lex} = 2^n$ définitions d'ordre lexicographique possibles, soit une complexité combinatoire difficile à manipuler pour les espaces de grande dimension. Les remarques suivantes peuvent toutefois être émises:

- bien que l'espace de travail soit de dimension n, il est tout à fait possible de définir un ordre lexicographique sur seulement $p < n$ de ces espaces. Pour des applications hyperspectrales, si par exemple les indices I des axes contenant le plus d'information ont été préalablement identifiés, la définition de l'ordre lexicographique sur la totalité des espaces n'est plus nécessaire. Ceci réduit considérablement la complexité du problème. L'ordre n'est plus total sur l'ensemble des axes des vecteurs, mais reste cependant total sur le sous-espace de dimension p et déterminé par les indices I.

- le point précédent est également vrai si $\mathbf{F}$ est un espace produit contenant un sous-espace ne possédant pas d'ordre naturel. C'est le cas par exemple pour les espaces couleurs cylindrique de type HLS: l'espace de teinte, H, du fait de sa circularité, ne possède pas de définition immédiate d'ordre. Deux solutions sont alors envisageables: soit l'espace posant difficulté est écarté de la relation d'ordre lexicographique (l'ordre ne se fait plus que sur les espaces de luminance L et de saturation S), soit un ordre presque quelconque est défini sur la teinte H, et H est moins prioritaire que les autres espaces au sens de la relation d'ordre lexicographique. Cette dernière solution nécessite toutefois la définition d'une origine pour l'espace H [(xii)].

- dans les deux cas précédents, l'intérêt d'une relation d'ordre lexicographique ne se borne pas qu'à l'extension simple des ordres à des espaces produits. Le fait de travailler sur des treillis lexicographiques, avec des éléments structurant plans, évite l'introduction de nouvelles couleurs dans un voisinage. C'est en effet un point problématique lorsque l'on souhaite visualiser le résultat de l'application des primitives: les transformations inverses étant fortement non linéaires, l'introduction de nouvelle couleur conduit à des aberrations visuelles. Cela n'est toutefois pas un réel problème si nous nous contentons de la visualisation des fonctions indicatrice et résiduelles.

Remarque : Le point majeur à l'approche est le suivant: lorsque l'on se borne à la Morphologie Mathématique à base d'éléments structurants « plans », c'est à dire non-valués [(xiii)], les opérateurs d'érosion et de dilatation correspondent à la collection respectivement de l'infimum et du supremum d'un sous-ensemble particulier. Dans ce cas précis il n'y a pas d'introduction de nouvelle valeur.

Lorsque l'on effectue une érosion sur un treillis lexicographique, le treillis obtenu étant en général complet (soit parce qu'il est fini, soit parce qu'il est produit de chaînes), les valeurs de l'image transformée sont prises à l'intérieur de l'ensemble des valeurs de l'image originale. Cette approche est à opposer à celle plus classique, où le traitement se fait de manière marginale dans chacun des canaux, approche introduisant de nouvelles valeurs. Mais nous reviendrons plus en détail sur ce point dans le

[(xii)] Si l'origine est un angle $\alpha \in [0, 2\pi[$, une transformation préalable de l'espace des teintes par la transformation $x \mapsto x - \alpha[2\pi]$ permet de se ramener au cas classique de l'origine sur le rouge $\alpha = 0$. Soulignons toutefois que l'opération modulo $x \mapsto x[2\pi]$ peut être assez coûteuse en temps de calcul si on travaille en précision à virgule flottante.

[(xiii)] ce qu'on nomme également des graphes de connexité dans le chapitre 2

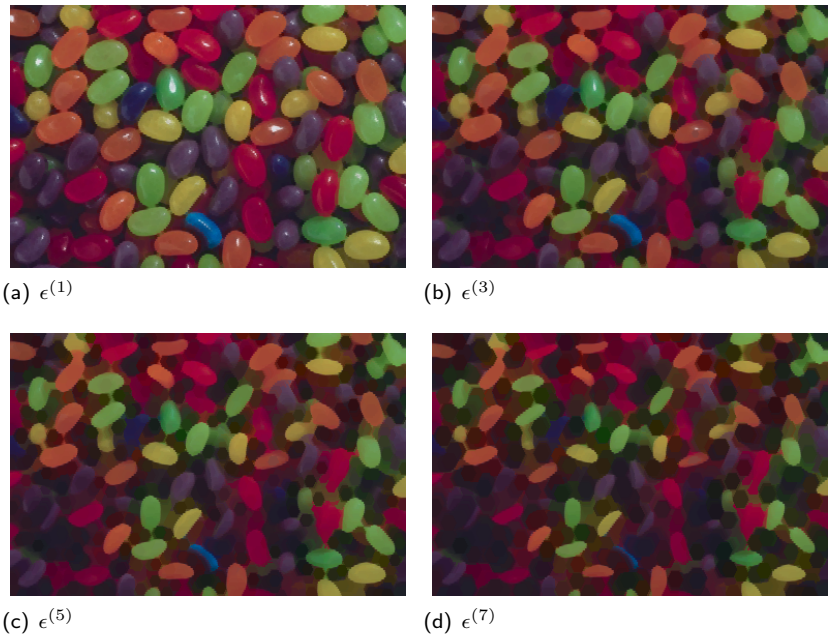
chapitre 4. Pour le moment, considérons simplement une érosion comme un opérateur infimum sur une collection de points.

Opérateur de distance : De la même manière que pour les ordres lexicographiques, il est possible de définir une infinité de fonctions distance sur les espaces choisis. Les espaces couleurs définissent toutefois certaines fonctions distance qui se sont avérées utiles dans certains cas. Ainsi, l'espace Lab a été défini de manière à appliquer la distance euclidienne sur chacune des composantes. Bien que moins rigoureusement défini que l'espace Lab, nous nous sommes particulièrement intéressés à l'espace HLS dans la mesure où la représentation de la couleur présente une approche beaucoup plus intuitive. Nous connaissons deux fonctions de distance sur cet espace : la distance luminance/teinte pondérée par la saturation (équation 4.6), ainsi que la distance angulaire en teinte uniquement (équation 4.5).

3.3.3.2. Quasi-distance lexicographique

Partant des précisions ci-dessus, les points sur lesquels nous pourrons définir une quasi-distance « lexicographique » apparaissent plus clairement. Nous bénéficions d'une structure de treillis lexicographique, et ainsi des primitives de Θ intervenant dans la définition de la quasi-distance. Reste à définir les résidus des érosions.

Érosions lexicographiques : Nous allons observer les résultats de l'érosion lexicographiques dans les espaces mentionnés. L'érosion utilise un élément structurant « plan », c'est à dire n'introduisant pas de nouvelle valeur [(xiv)]. Les résultats de quelques unes de ces érosions sont données sur les figures 3.32 à 3.35. L'image très colorée servant à l'illustration est un bon exemple pour appréhender le comportement de l'érosion lexicographique dans l'espace HLS.

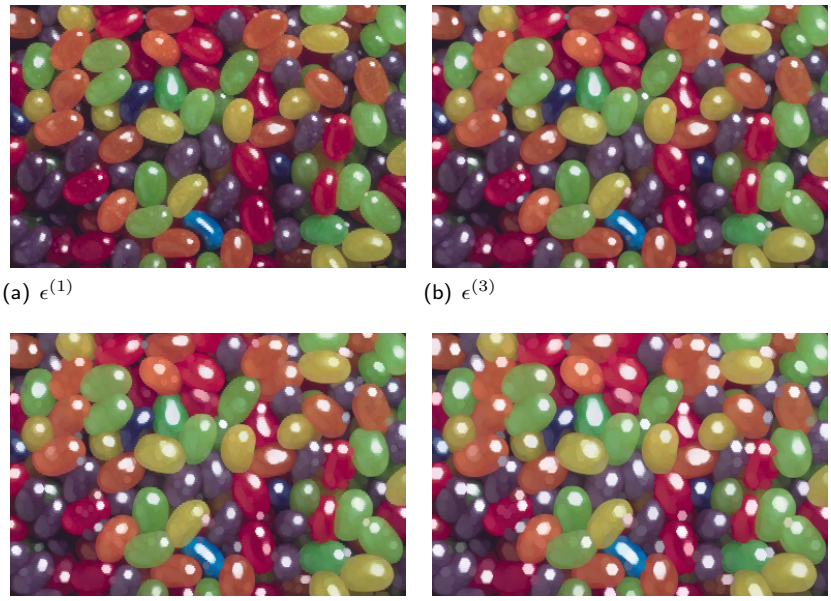


(a) $\epsilon^{(1)}$ (b) $\epsilon^{(3)}$

(c) $\epsilon^{(5)}$ (d) $\epsilon^{(7)}$

Fig. 3.32.: Érosions lexicographiques dans le treillis L, S, H (ordres naturels). Les images sont d'abord transformées dans l'espace HLS. L'érosion est ensuite calculée, enfin la transformation couleur inverse est appliquée de manière à apprécier les résultats.

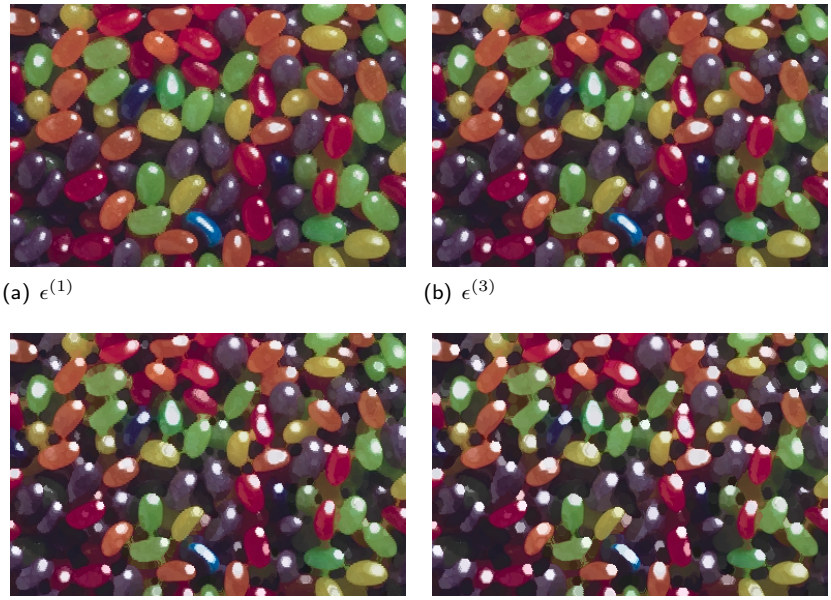
[(xiv)] C'est d'ailleurs la raison pour laquelle nous pouvons appliquer la transformation couleur inverse, la transformation HLS bien que bijective, ne donne pas de bons résultats visuels lorsque de nouvelles valeurs sont introduites



(a) $\epsilon^{(1)}$ (b) $\epsilon^{(3)}$

(c) $\epsilon^{(5)}$ (d) $\epsilon^{(7)}$

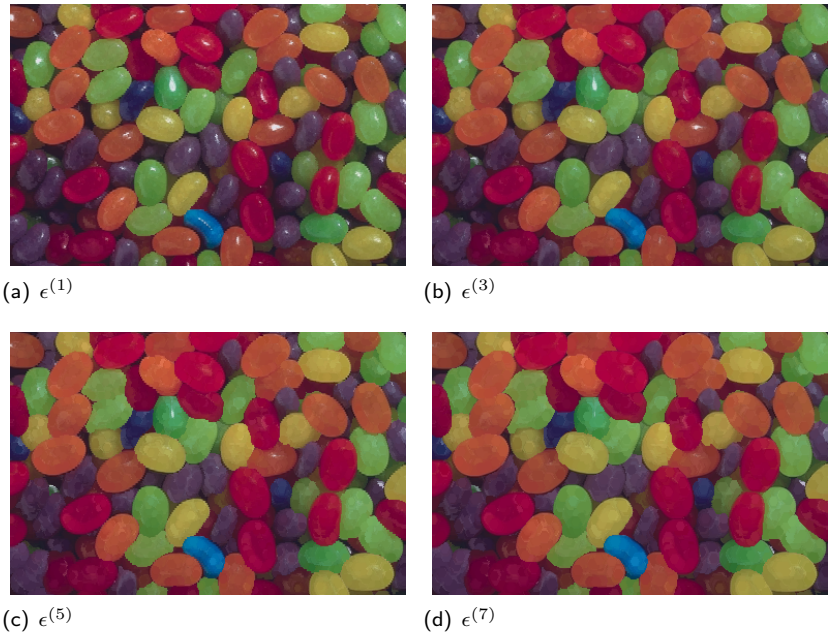
Fig. 3.33.: Érosions lexicographiques dans le treillis L, S, H (ordres inverses sur chaque canal). L'ordre inverse sur la luminance revient à dilater les zones de reflet important sur les bonbons. L'image devient ensuite rapidement très lumineuse.



(a) $\epsilon^{(1)}$ (b) $\epsilon^{(3)}$

(c) $\epsilon^{(5)}$ (d) $\epsilon^{(7)}$

Fig. 3.34.: Érosions lexicographiques dans le treillis S, L, H (ordres naturel). Les zones les moins saturées prennent le « dessus » et les bonbons disparaissent à cause du reflet à leur centre, et de leur bord moins saturé.



(a) $\epsilon^{(1)}$ (b) $\epsilon^{(3)}$

(c) $\epsilon^{(5)}$ (d) $\epsilon^{(7)}$

Fig. 3.35.: Érosions lexicographiques dans le treillis S, L, H (ordres inverses).

Fonction de distance : comme décrit précédemment, l'application d'une fonction distance couleur est une première méthode pour l'obtention d'un résidu, lorsque le résultat des primitives est également couleur. C'est par ailleurs le cas pour les érosions lexicographiques. Le résidu étant alors scalaire, l'algorithme des quasi-distances s'effectue ensuite de la même manière que dans le cas « numérique »: par sélection de l'indice du supremum des résidus. Des exemples de résidu, pour des relations d'ordre différentes et une même métrique couleur, sont donnés sur les figures 3.36 et 3.37

Il n'est cependant pas très pratique de comparer ces fonctions distances.

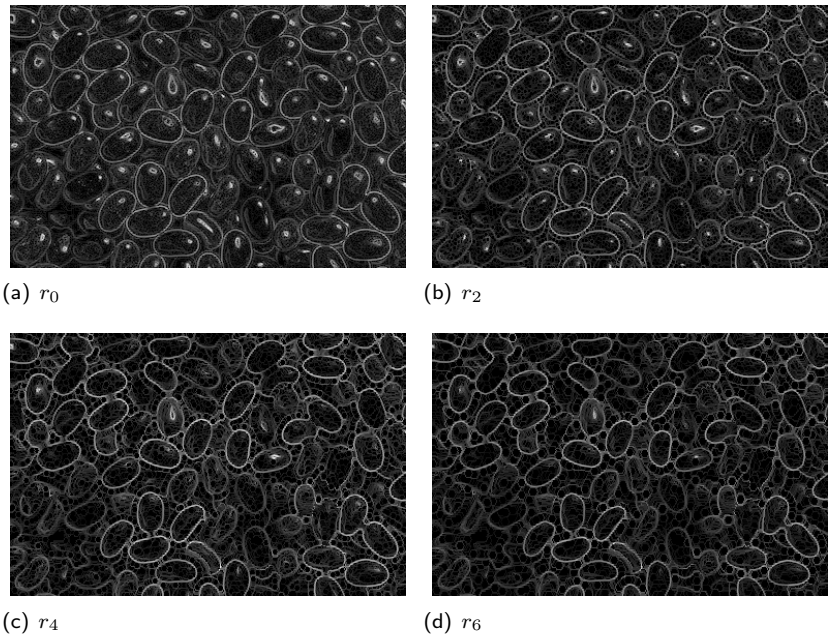
Résultat de la quasi-distance : Selon ces expériences, le résultat des quasi-distances n'est pas très pertinent sur les treillis L, S, H et S, L, H par rapport à une approche marginale ou monochrome. Ces résultats sont repris sur la figures 3.38.

D'après nos tests, dans l'espace HLS, la métrique basée sur la distance en teinte ne donne pas de résultat intéressant, dans la mesure où trop d'éléments sont perdus (dernière ligne de la figure). Il semblerait que l'approche lexicographique en priorité de luminance apporte un élément de détail dans les zones où la luminance semble constante (zone gauche basse à comparer entre les figures 3.38(b) et 3.38(f)). Le treillis lexicographique en priorité de saturation semble moins sensible aux reflets sur les bonbons (les trous creusés par les reflets en 3.38(d) n'apparaissent pas 3.38(h), bien que la distance soit au final moins précise).

La figure 3.39 montre des résultats assez similaires entre les versions marginales et lexicographiques de la quasi-distance. Dans cet exemple, nous avons inversé les canaux de saturation et de luminance : la perte de luminance s'accompagnera d'augmentation de saturation. La différence en teinte ne donne pas des résultats satisfaisants dans la mesure où de nombreux grains fusionnent dans les treillis de saturation prioritaire. Pour les quatre colonnes, nous remarquons qu'en général l'approche marginale (luminance seule ou saturation seule) produit de meilleurs résultats.

3.3.3.3. Conclusion et perspective

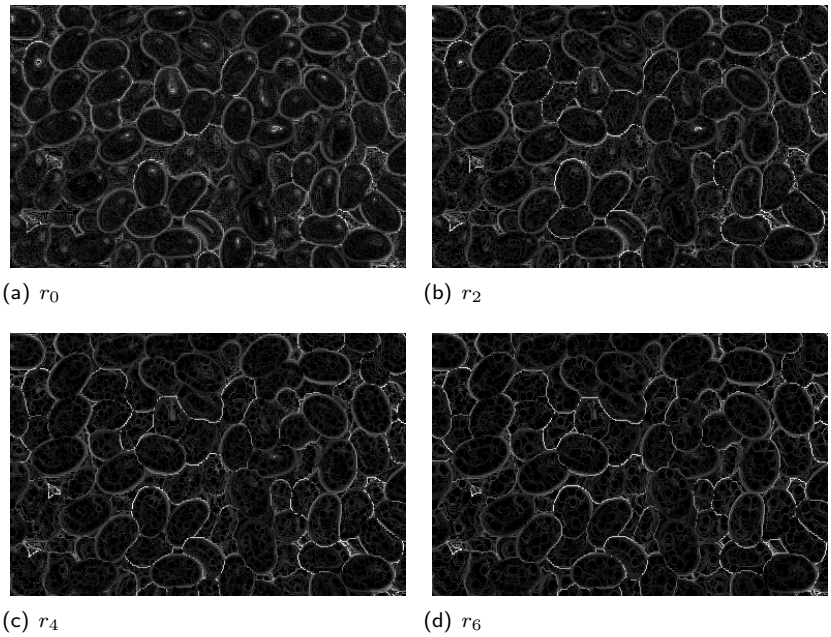
Cette conclusion peut sembler hâtive. Comme nous l'avons précisé, le nombre de combinaisons devient rapidement extrêmement élevé: d'une part par le nombre de treillis lexicographiques possibles,



(a) r_0 (b) r_2

(c) r_4 (d) r_6

Fig. 3.36.: Résidus des érosions lexicographiques dans le treillis L, S, H par la métrique « HL - saturation pondéré ».

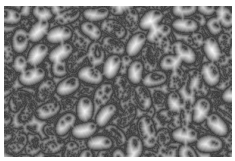


(a) r_0 (b) r_2

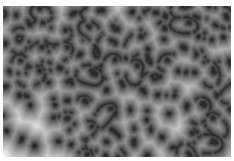
(c) r_4 (d) r_6

Fig. 3.37.: Résidus des érosions lexicographiques dans le treillis S, L, H (ordres inverses) de la figure 3.35 par la métrique « HL - saturation pondéré »

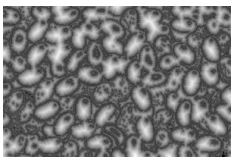
(a) Image originale



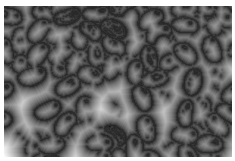
(b) QD - luminance



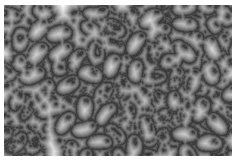
(c) QD - luminance inverse



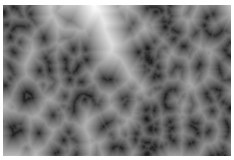
(d) QD - saturation



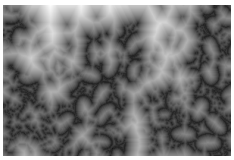
(e) QD - saturation inverse



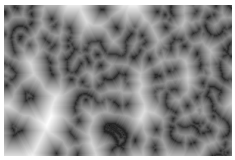
(f) QD - L, S, H - métrique « HL - saturation pondéré »



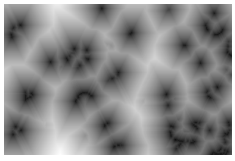
(g) QD - L, S, H inversés - métrique « HL - saturation pondéré »



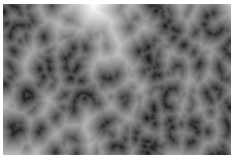
(h) QD - S, L, H - métrique « HL - saturation pondéré »



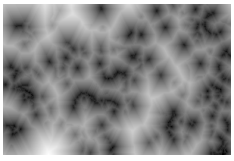
(i) QD - S, L, H inversés - métrique « HL - saturation pondéré »



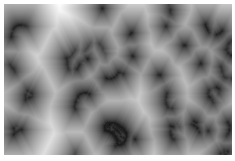
(j) QD - L, S, H - métrique teinte



(k) QD - L, S, H inversé - métrique teinte



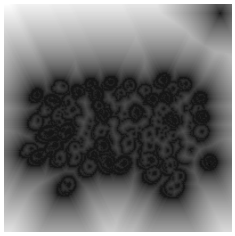
(l) QD - S, L, H - métrique teinte



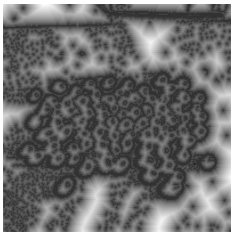
(m) QD - S, L, H inversé - métrique teinte

Fig. 3.38.: Quasi-distance sur le treillis L, S, H, et S, L, H

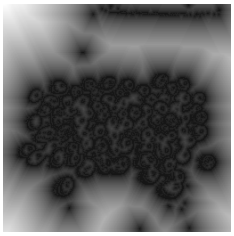
(a) Image originale



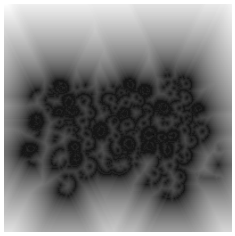
(b) QD - luminance



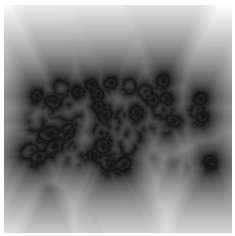
(c) QD - luminance inverse



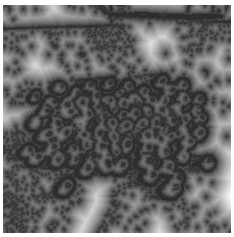
(d) QD - saturation



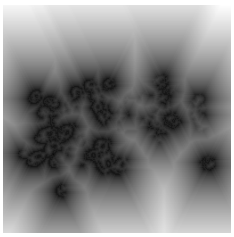
(e) QD - saturation inverse



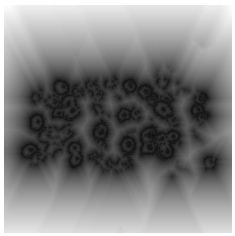
(f) QD - L, S inversé, H - métrique « HL - saturation pondéré »



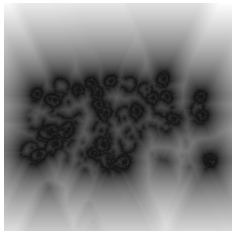
(g) QD - L inversé, S, H inversé - métrique « HL - saturation pondéré »



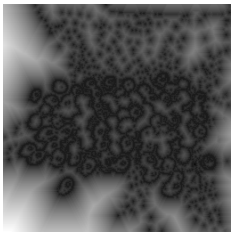
(h) QD - S, L inversé, H inversé - métrique « HL - saturation pondéré »



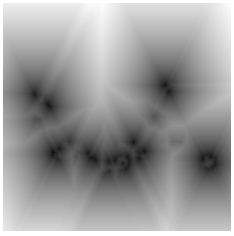
(i) QD - S inversé, L, H - métrique « HL - saturation pondéré »



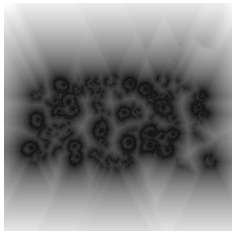
(j) QD - L, S inversé, H - métrique teinte



(k) QD - L inversé, S, H inversé - métrique teinte



(l) QD - S, L inversé, H inversé - métrique teinte



(m) QD - S inversé, L, H - métrique teinte

Fig. 3.39.: Quasi-distance sur le treillis L, S inversé, H, et S inversé, L, H

d'autre part par le nombre de fonctions de distance couleur utilisables. Nous avons cependant appliqué ici le même schéma qu'avait proposé Beucher lors de l'introduction des quasi-distances, en utilisant certains des outils mis à disposition par notre structure logicielle. Malgré la difficulté évidente de l'appréhension de la quasi-distance couleur définie comme telle - cet opérateur n'était certes pas évident à comprendre dans le cas numérique - nous ouvrons une voie qui semble intéressante pour un certain nombre d'applications. Les images en figure 3.39 montrent par exemple que la quasi-distance, marginale numérique ou lexicographique, montre de bonnes propriétés pour séparer les zones de couleurs homogènes des graines de couleurs.

Dans une plus lointaine perspective, il est également possible de donner une autre définition à l'approche couleur. En effet, il est possible de munir l'espace des résidus d'une structure de treillis lexicographique, puisque seul le supremum nous intéresse. Les résidus seraient vectoriels (par application d'une différence marginale par exemple), et l'opérateur supremum serait défini sur cet espace vectoriel. Nous laissons ces travaux à l'appréciation du lecteur.

3.4 Conclusion

Résumé

Dans ce chapitre, nous avons présenté deux approches complètement différentes autour du calcul de distances dans une image. La première approche était consacrée au calcul de distances exactes à un sous-ensemble, quelles que soient la dimension de l'image et la distance considérée, pourvu que sa boule unité soit convexe. La deuxième approche était consacrée au calcul de distances morphologiques récentes appelées les quasi-distances, à l'amélioration des temps de calcul et leur extension à la couleur.

• Le calcul de la fonction distance exacte a nécessité un développement théorique qui nous a permis de construire et valider l'algorithme proposé. Pour cela, nous avons tout d'abord établi le lien entre la dilatation et la convexité de la boule unité de la distance utilisée. Ce lien nous a ensuite permis de proposer un algorithme à base de propagations. Nous avons enfin énoncé la condition pour qu'un point de l'espace reçoive toutes les informations suffisantes au calcul de la distance exacte au sous-ensemble. Enfin, l'analyse des résultats confirme les développements théoriques ayant permis la construction de l'algorithme.

L'approche que nous avons proposée ne posait pas comme contrainte principale le temps de calcul, mais plutôt la souplesse et la généricité. En ce sens, les hypothèses nécessaires au fonctionnement de l'algorithme, que cela soit en termes de dimension des images ou de la forme de la fonction distance, semblent parfaitement adaptées à une librairie de traitement d'image orientée recherche. Les temps de calcul ne seront certainement pas comparables à des algorithmes dédiés à une fonction de distance et une dimension d'image particuliers et il ne nous parait pas possible, par exemple, de faire fonctionner un tel algorithme en moins d'une seconde (sur les images considérées). Par contre, ces mêmes algorithmes optimisés ne fonctionneront certainement pas lors d'un changement de dimension, comme l'a souligné Cuisenaire [Cui99] [(xv)]. Enfin, la preuve de convergence et d'exactitude dans la trame discrète est également un aspect de l'algorithme qui nous paraissait particulièrement important, ce qui ne semble pas être fait à notre connaissance dans la littérature pour le cadre proposé. Nous insistons donc sur ces deux points pour mettre en relief l'originalité de notre approche.

• Dans la seconde partie, nous avons présenté des développements autour des « Quasi-distances » introduites par Beucher. Nous avons illustré et commenté l'approche initiale. Nos développements nous ont ensuite permis de justifier l'approche de deux nouveaux algorithmes : le calcul des érodés successifs et la régularisation de la fonction distance. Nous avons pour cela étudié le critère d'arrêt de l'algorithme, ainsi que les lieux d'activité des primitives à chaque itération.

L'évaluation des deux algorithmes proposés à la fois en termes de réduction de calcul et de temps d'exécution permet de dire que les résultats sont satisfaisants. La transformation reste néanmoins trop coûteuse pour être exploitée en temps réel. Nous n'avons pas particulièrement cherché à proposer des applications potentielles de cette transformation. Nous renvoyons pour cela le lecteur aux travaux présentés par Beucher [Beu05]; d'autres travaux sont en cours. Nous avons toutefois mis en parallèle cette transformation avec celle de distance morphologique classique. Sachant que cette dernière peut être utilisée à des fins nombreuses (dont le filtrage d'image en vue d'une segmentation), il ne nous a pas paru utile de présenter des applications potentielles, car le point de vue aurait été finalement réducteur. Nous avons préféré axer les développements sur le calcul d'une telle transformation, ce qui fait le lien avec le chapitre 2 des développements informatiques.

Nous avons enfin proposé une extension des quasi-distances, ou plus généralement des opérateurs résiduels, aux images couleurs. Cette extension s'opère par l'intervention de métriques couleurs, et de treillis lexicographiques. La conclusion du paragraphe §3.3.3.3 (page 95) souligne la difficulté de l'interprétation des résultats. L'exemple de la quasi-distance, bien que trivial, reste relativement complexe

[(xv)] L'auteur propose un premier schéma algorithmique pour le calcul de la distance euclidienne valable en $2D$ mais souligne que ce schéma n'est pas valide en $3D$ et donc nécessairement à des dimensions supérieures.

car dépendant d'un certain nombre de choix en amont: ordre lexicographique, distance couleur, etc..

Perspectives

Les perspectives sont nombreuses pour les deux approches. Pour le calcul des distances exactes, il serait aisé d'étendre l'algorithme initial au calcul de la distance exacte à des formes géométriques définies uniquement par leur équation (et non par leur projection sur la trame discrète): distance à une droite dans l'espace, à une conique.

Concernant les quasi-distances, de nombreuses idées s'ajoutent également à celles proposées. L'amélioration des temps de calcul reste encore une fois un axe de développement important, mais nous a paru loin d'être évidente. C'est pourquoi nous pensons qu'il faudrait certainement privilégier l'utilisation de variantes de la définition initiale de quasi-distance, et découvrir au mieux leur application. Les différentes variations de l'algorithme concerneraient la définition du résidu maximal qui pourrait par exemple être fonction du numéro d'itération (et qui permettrait de définir d'autres critères d'arrêt), des critères d'arrêt dépendant de la dérivée d'une fonction du résidu total dans l'image, etc.. Enfin pour l'extension lexicographique que nous avons proposée, il s'agit bien sûr d'une ébauche qui peut être grandement complétée, comme nous l'avons souligné précédemment.

Cette dernière partie nous sert d'introduction pour le chapitre suivant, dans lequel nous allons étudier différentes approches dans le traitement de la couleur, dont l'utilisation d'ordres lexicographiques.

CHAPITRE 4

Vers le traitement multispectral

« Les choses prennent la couleur de nos contrariétés. »
Les demi-civilisés, Jean-Charles Harvey

Le cadre informatique introduit au chapitre 2 permet de manipuler des images dites « multispectrales », c'est à dire des images dont les valeurs des pixels (espace d'arrivée $\mathbf{F}$) sont vectorielles. Nous explorons dans ce chapitre quelques traitements liés à ces images. Dans ce sens, les images couleur serviront de support d'étude.

Le chapitre est composé de deux parties: une première partie - composée par les sections §4.2 et §4.3 - décrit les approches multispectrales envisagées dans l'étude. La seconde partie composée par les sections §4.4 et §4.5 décrit deux applications liées aux développements de la première partie. Ces applications concernent la détection de zones de peau dans le cadre de sécurité automobile, et la détection d'objets en mouvement dans le cadre de la vidéosurveillance.

Les traitements multispectraux seront envisagés selon trois approches: métrique, statistique et algébrique. L'approche **métrique** - §4.2.1 - utilisera une fonction de distance dans l'espace des valeurs $\mathbf{F}$. L'approche dite **statistique** - §4.2.2 - utilisera une mesure *statistiques* calculée localement.Enfin, nous aborderons le point de vue **algébrique**, par l'utilisation de relation d'ordre lexicographiques. Pour ces ordres, nous reportons le lecteur à la desciption §2.2.4 page 25.

L'approche métrique permettra de formuler le module du gradient morphologique dans les espaces multispectraux. L'application d'un tel opérateur effectue le passage du domaine multispectral vers le domaine scalaire, dont le traitement est généralement plus facile. La formulation métrique du gradient ne sera toutefois pas adaptée pour les voisinages de grande taille. Nous utiliserons alors l'approche statistique. Nous avons porté nos efforts principalement sur l'espace HLS, qui présente une difficulté dûe à la circularité de la teinte. Les gradients morphologiques seront également envisagés d'un point de vue algébrique à l'aide des relations d'ordre lexicographiques.

Les relations d'ordre lexicographiques ont cependant une portée beaucoup plus grande que la simple définition de gradients. Ils permettent en effet de définir des treillis complets sur des espaces multispectraux, ce qui permet ensuite d'utiliser des algorithmes « *classiques* » de morphologie mathématique sur ces espaces. Grâce au cadre introduit au chapitre 2, la plupart des transformations morphologiques peuvent « immédiatement » être définies sur ces treillis. Nous allons voir dans quelle mesure le cadre informatique établi au chapitre 2 nous permet de transposer les algorithmes existants sur ces treillis. Nous illustrerons ceci par les algorithmes d'extraction d'extrema, de reconstruction et de granulométries

lexicographiques.

Enfin nous illustrerons les développements présentés sur deux applications concrètes. La première est la caractérisation de la chrominance de peau de manière robuste aux changements d'illuminant. La seconde application porte sur la transposition des techniques classiques de détection de mouvement à la couleur, dans le cadre de la vidéosurveillance.

Pour le lecteur non initié à la couleur, nous avons placé en annexe A une description de la couleur selon une approche à la fois physique et informatique. Le lecteur y trouvera également les principaux espaces utilisés dans ce chapitre, ainsi que leur transformation.

4.1 Avant-propos : Approches connues du traitement multispectral

La richesse de l'information couleur par rapport à l'information scalaire ouvre un certain nombre d'alternatives dans le choix des traitements. Les approches connues se réfèrent souvent à leur homologue scalaire. Parmi l'ensemble des approches, les trois présentées ci-dessous sont soit utilisées couramment, soit présentant un intérêt particulier dans ce chapitre.

Dans la suite, nous nous intéressons à une fonction de traitement f des données de l'espace image $\mathbf{F}$. L'espace d'arrivée de f est considéré quelconque. En effet, il est possible d'avoir des fonctions de transformation couleur définies sur un espace de dimension 3 et à valeur dans un espace de dimension 4 (par exemple RGB vers CMYK).

Traitement marginal

Par traitement marginal, nous entendons la possibilité de projeter l'espace des données $\mathbf{F}$ sur une base de dimension p, disons $\{\mathbf{F}_i\}_{i\in\mathbb{N}_p^*}$ et d'appliquer un traitement sur chacun de ces sous-espaces de manière indépendante. Soit f la fonction de traitement mentionnée:

$$\forall p \in \mathbf{F},\ f(p) = (f_i(p_i))_{i\in\mathbb{N}_p^*} \tag{4.1}$$

où chaque f_i représente le traitement de f sur le sous-espace F_i. Si mathématiquement, cette équation propose des choix très nombreux de fonction f applicable; ce choix est réduit si le point de vue visuel est pris en compte.

1. Une fois les traitements effectués sur chacune des composantes, nous devons recombiner ces informations de manière correcte et pertinente. Se pose alors le problème de la méthode de recombinaison.
2. Bien que liée à la recombinaison, l'approche marginale a pour conséquence presque inévitable l'introduction de nouvelles couleurs

Le deuxième point est assez problématique dans certains espaces pour la raison suivante : malgré la bijectivité de la transformation de RGB de et vers HLS, la modification ne serait-ce que faible d'une des composantes d'un vecteur dans HLS conduit à de grande différences dans RGB. Ceci est explicable par le fait que cette transformation n'est absolument pas linéaire: une variation linéaire dans HLS ne se transforme pas en une variation linéaire dans RGB, et conduit à des artefacts visuels assez importants. Pour cette transformation, les filtrages marginaux sont très difficiles [(i)] et seule une approche préservant les vecteurs dans HLS est possible.

L'approche marginale a toutefois l'avantage de permettre le portage immédiat des traitements scalaires vers des données multispectral.

[(i)] si on souhaite un retour sur RGB

Traitement vectoriel

Les traitements vectoriels ne peuvent être décomposés de la manière évoquée par l'équation 4.1. Ils agissent sur la totalité des composantes du vecteur en entrée. En fait, le traitement marginal est un cas particulier du traitement vectoriel. La plupart des fonctions de transformation d'espace couleur sont des exemples de traitement vectoriel.

L'approche vectorielle est difficile, et assez éloignée de la démarche algébrique propre à la Morphologie Mathématique. Elle sera préférée dans certaines disciplines de filtrage linéaire [TFMB04]. Nous n'utiliserons ce type de fonction que pour des transformations d'espace.

Traitement algébrique

La morphologie mathématique définit un cadre algébrique induisant la *stabilité* des données. Par stabilité, nous faisons référence à la non-introduction de nouvelles valeurs, par opposition aux approches marginales et vectorielles. Ceci est dû à la structure de treillis présentée en §2.2.4 (page 25). En effet, un treillis est une structure algébrique stable pour ses opérateurs $\overline{\wedge}$ et $\underline{\vee}$.

En définissant les opérateurs $\overline{\wedge}$ et $\underline{\vee}$ sur un ensemble $\mathbf{F}$ de manière à ce que la structure $\mathcal{L}_{\mathbf{F}} = \langle \mathbf{F}, \underline{\vee}, \overline{\wedge} \rangle$ soit un treillis, il est possible de définir des opérateurs de morphologie mathématique par une transposition simple du cadre scalaire. En effet, la plupart des opérateurs morphologiques sont définis de manière algébrique, ce qui les rend indépendants de la structure du treillis sous-jacent.

Nous voyons donc que le travail principal pour une morphologie mathématique multispectrale réside dans le choix des opérateurs $\overline{\wedge}$ et $\underline{\vee}$. Pour cela, l'approche par relation d'ordre lexicographique (cf. définition 2.10 page 26) est suffisamment simple pour être exploitée rapidement. Cela suppose que l'espace $\mathbf{F}$ soit décomposable sur une base $\mathbf{F}_i$, et que chaque $\mathbf{F}_i$ puisse être muni d'une relation d'ordre « $\preceq_i$ ». Cette hypothèse n'est cependant pas une limitation en soit: nous avons en effet supposé au §2.2.1 que l'espace $\mathbf{F}$ est assimilable à un espace vectoriel.

Enfin, mentionnons le fait que les ordres lexicographiques ne sont pas l'unique solution pour définir des treillis sur des espaces couleurs. Hanbury [Han02a] a proposé une relation d'ordre sur des données circulaires, ce qui lui a permit de définir des opérateurs morphologiques dans la dimension en teintes de l'espace HLS.

4.2 Approches métriques et statistiques

Nous étudions dans cette section les approches métriques et statistiques, pour la définition d'opérateurs morphologiques de voisinage. Selon la philosophie des chapitres précédents, ainsi que du projet fédérateur de bibliothèque de traitement morphologique par des algorithmes **génériques** [(ii)], nous nous intéressons uniquement à des méthodes indépendantes des dimensions impliquées. Les approches que nous avons nommées *métriques* et *statistiques* s'inscrivent parfaitement dans cette démarche.

4.2.1. Espaces couleurs et métriques

Les espaces couleurs sont une représentation numérique particulière de l'information couleur. Dans le domaine du visible, cette information se réduit généralement à une donnée en 3 dimensions. Le lecteur intéressé trouvera en annexe A (page 249) une introduction assez exhaustive à la couleur.

Bien que la notion d'origine dans un espace couleur soit mathématiquement plausible, le sens qui lui est donné n'est pas aussi évident que pour des dimensions physiques représentant une amplitude; s'il est concevable que la luminance ait une origine - nulle pour une quantité nulle de photons - la couleur, c'est à dire la distribution des photons sur le domaine spectral, peut avoir une infinité d'origines différentes (par exemple une longueur d'onde λ précise).

[(ii)] c'est-à-dire indépendant de la dimension du support $\mathbf{E}$ et de l'espace image $\mathbf{F}$ de l'image

À défaut d'origine, certains espaces sont munis d'une notion de distance: il est alors possible de quantifier la différence entre deux couleurs. Bien que la notion de distance soit riche, elle n'est pas suffisante pour définir des opérateurs morphologiques. Il est cependant possible de définir certaines opérations intéressantes. Nous allons dans un premier temps expliciter quelques métriques couleurs, que nous appliquerons ensuite à la définition de gradient morphologique.

4.2.1.1. Métriques couleur

Comme nous l'avons mentionné, la notion d'origine est paradoxale dans les espaces couleurs. Si en effet il est possible de considérer le noir absolu comme l'origine de l'espace, le noir concerne principalement l'absence de puissance spectrale. Cette quantité est donnée dans la plupart des espaces par le canal de luminance.

Tous les espaces ne sont pas naturellement munis de métrique. Théoriquement cette notion ne présente aucun problème: en effet, tous les espaces couleurs sont de type vectoriels [(iii)] et de dimension finie; l'expression d'une norme sur ces espaces peut être faite de manière triviale.

Cependant, une distance doit refléter une certaine réalité par rapport aux couleurs. La réalité est prise au sens de la comparaison de couleurs: la distance sert à quantifier la *différence* entre deux couleurs. Deux espaces attirent particulièrement notre attention: Lab et HLS. Lab fait partie d'une famille d'espaces dite *colorimétrique*, alors que HLS fait partie d'une famille dite *circulaire* [(iv)].

Distances dans les espaces colorimétriques : Tous les espaces colorimétriques ne bénéficient pas de la notion de distance. La finalité première de XYZ était par exemple la représentation des couleurs de manière absolue, c'est à dire en prenant en compte la température de l'illuminant, et en donnant la possibilité d'exprimer toutes les couleurs [(v)]. L'absence de notion de mesure cependant ne nous permet pas d'utiliser cet espace.

Les espaces colorimétriques de type Lab sont par construction basés sur la perception humaine. Ceci signifie entre autre qu'ils ont été définis de telle manière à ce qu'ils soient naturellement munis d'une distance permettant de quantifier la différence entre deux couleurs. Cette différence fut déduite empiriquement sur des tests humains. Lab a ensuite été définit de manière à ce que l'expression de la distance soit simple. Dans la suite, $\mathbf{F}$ définit l'espace couleur Lab, et chaque point de $\mathbf{F}$ possède une composante en luminance et deux composantes chromatiques rouge-vert (a) et jaune-bleu (b): $\mathbf{x} = (L, a, b)$.

$$d^e_{\text{Lab}} : \begin{cases} \mathbf{F} \times \mathbf{F} & \rightarrow & \mathbb{R}_+ \\ (\mathbf{x}_1, \mathbf{x}_2) & \mapsto & \|(L_1, a_1, b_1) - (L_2, a_2, b_2)\|_2 \end{cases} \tag{4.2}$$

Lab offre également une distance chromatique (en teinte) de la manière suivante:

$$d^h_{\text{Lab}} : \begin{cases} \mathbf{F} \times \mathbf{F} & \rightarrow & \mathbb{R}_+ \\ (\mathbf{x}_1, \mathbf{x}_2) & \mapsto & \left((a_2 - a_1)^2 + (b_2 - b_1)^2 - \left(\sqrt{a_1^2 + b_1^2} - \sqrt{a_2^2 + b_2^2} \right)^2 \right)^{\frac{1}{2}} \end{cases} \tag{4.3}$$

Quelques précautions sont à évoquer quant à l'utilisation de ces distances : comparer la différence de couleur entre deux couleurs très différentes n'a pas de sens: un bleu n'est pas plus « loin » qu'un rouge ou qu'un vert. La distance ici est un outils de mesure de ressemblance des couleurs, mais non de *dissemblance*. Les mesures de distance ne sont alors valides que dans une échelle limitée, décrite par le paramètre *JND*[(vi)]. La distance indique des différences de couleur à peine ou pas du tout perceptible pour un *JND* inférieur à 3, mais n'apporte plus aucun élément d'information pour un *JND* supérieur à 6.

[(iii)] la circularité de certains espaces ne concerne que la représentation des vecteurs dans cet espace
[(iv)] parfois appelée *interface*, famille caractérisée par la simplicité quasi-intuitive de la représentation.
[(v)] ce qui n'est pas le cas avec l'espace RGB par exemple; cf. annexe A.2
[(vi)] pour « Just Noticeable Difference ».

Distances dans des espaces circulaires : Les espaces couleurs circulaires permettent une manipulation plus intuitive, en termes de description, de l'information couleur. La construction de ces espaces n'a pas de fondement colorimétrique, alors que les espaces types Lab ont été construits de manière à pouvoir quantifier des différences entre couleurs proches. La plupart du temps, la transformation à partir de l'espace RGB est géométrique : le cube est réorienté selon l'axe achromatique, le long duquel la couleur n'existe pas. La projection d'un vecteur dans RGB sur le plan perpendiculaire à l'axe achromatique - ie. le plan chromatique - sépare la composante chromatique du vecteur couleur. La caractéristique des espaces circulaires concerne la manière dont sont représentés les projetés sur le plan chromatique : les coordonnées dans ce plan sont exprimées sous forme *polaire*.

L'absence de fondement colorimétrique n'empêche pas l'existence de métriques sur ces espaces, qui produisent par ailleurs des résultats tout à fait acceptables. Dans la suite $\mathbf{F}$ définit l'espace couleur HLS, et chaque point de $\mathbf{F}$ possède les trois composantes teinte, luminance et saturation : $\mathbf{x} = (h, l, s)$.

Nous noterons également $(\mathbf{x}_1, \mathbf{x}_2) \mapsto |h_1 \angle h_2|$ la différence angulaire normalisée, donnant la mesure de l'angle aigu entre h_1 et h_2 sur le cercle unité. Plus précisément, il s'agit de l'application suivante:

$$\angle : \left\{ \begin{array}{lcl} \mathbf{F} \times \mathbf{F} & \rightarrow & [0,1] \\ (\mathbf{x}_1, \mathbf{x}_2) & \mapsto & \frac{1}{\pi} \cdot \left\{ \begin{array}{ll} |h_1 - h_2| \,[2\pi] & \text{si } |h_1 - h_2| \,[2\pi] \leq \pi \\ 2\pi - |h_1 - h_2| \,[2\pi] & \text{sinon} \end{array} \right. \end{array} \right. \tag{4.4}$$

La normalisation de la mesure d'angle sur $[0, 1]$ permet d'effectuer des sommes pondérées de distances sur différents axes, en uniformisant le poids de chacun des axes dans la somme (une telle distance sera donnée par la suite par l'équation 4.6). Si on omet cette normalisation, la différence en teinte se trouve sur le domaine $[0, \pi]$ et risque de « capturer » l'essentiel de la distance. Cet espace bénéficie alors directement de trois métriques définies de manière marginale sur les trois composantes prises indépendamment. La distance en teinte est donnée par :

$$d^h_{\text{HLS}} : \left\{ \begin{array}{lcl} \mathbf{F} \times \mathbf{F} & \rightarrow & \mathbb{R}_+ \\ (\mathbf{x}_1, \mathbf{x}_2) & \mapsto & |h_1 \angle h_2| \end{array} \right. \tag{4.5}$$

Angulo [AL03] définit également des combinaisons de gradients marginaux par supremum des trois canaux. Le supremum risque naturellement de propager les défauts des différents gradients, défauts qui peuvent être assez importants.

La distance angulaire donnée par l'équation 4.5 est malheureusement très sensible dans les zones sombres, ou plus généralement dans les zones achromatiques. Cela s'explique par le fait que lorsque la luminance est proche de 0 ou de 1, la dynamique de la saturation diminue, et la teinte contient une information de moins en moins pertinente. Ceci s'explique par la géométrie en « double cône » de l'espace HLS (cf. figure 4.1). Dès lors que la saturation s'approche de 0, la teinte n'est pas une information fiable - non définie par exemple pour une saturation nulle - ; dans les zones de luminance faible (et donc de saturation faible), la teinte fluctue généralement beaucoup.

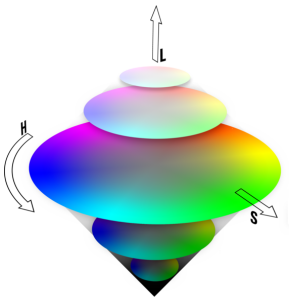


Fig. 4.1.: Géométrie de l'espace HLS. Pour des valeurs de luminance proche de 0 ou 1, la dynamique de la saturation diminue, et la pertinence de l'information de teinte s'appauvrie (source www.wikipedia.fr, sous licence « GNU Free Documentation License »).

Cette mauvaise propriété conduit Angulo [AL03] à définir une distance utilisant la saturation comme facteur pondérant. Ce facteur est linéaire et répartit l'influence entre le gradient en teinte et en luminance sur le gradient final. À l'origine non symétrique, la « distance » que nous proposons ici est la suivante:

$$d^{hw}_{\mathsf{HLS}} : \begin{cases} \mathbf{F} \times \mathbf{F} & \rightarrow & \mathbb{R}_+ \\ (\mathbf{x}_1, \mathbf{x}_2) & \mapsto & \frac{s_1+s_2}{2} |h_1 \angle h_2| + \left(1 - \frac{s_1+s_2}{2}\right) |l_1 - l_2| \end{cases} \tag{4.6}$$

La fonction donnée par l'équation 4.6 n'est cependant pas une distance, car elle ne vérifie par la propriété de séparation $d^{hw}_{\mathsf{HLS}}(\mathbf{x}, \mathbf{y}) = 0 \Rightarrow \mathbf{x} = \mathbf{y}$.

En effet, l'équation $d^{hw}_{\mathsf{HLS}}(\mathbf{x}, \mathbf{y}) = 0$ entraîne les trois systèmes suivants:

1. $$\begin{cases} s_\mathbf{x} + s_\mathbf{y} & = 0 \\ l_\mathbf{x} & = l_\mathbf{y} \\ h_\mathbf{x} \text{ et } h_\mathbf{y} & \text{ quelconques} \end{cases} \Rightarrow \begin{cases} s_\mathbf{x} = s_\mathbf{y} & = 0 \\ l_\mathbf{x} & = l_\mathbf{y} \\ h_\mathbf{x} \text{ et } h_\mathbf{y} & \text{ quelconques} \end{cases}$$
   car $s \in [0, 1]$.
2. $$\begin{cases} s_\mathbf{x} + s_\mathbf{y} & = 2 \\ h_\mathbf{x} \angle h_\mathbf{y} & = 0 \\ l_\mathbf{x} \text{ et } l_\mathbf{y} & \text{ quelconques} \end{cases} \Rightarrow \begin{cases} s_\mathbf{x} = s_\mathbf{y} & = 1 \\ h_\mathbf{x} & = h_\mathbf{y} \\ l_\mathbf{x} \text{ et } l_\mathbf{y} & \text{ quelconques} \end{cases}$$
   car $(\star, \bullet) \mapsto |\star \angle \bullet|$ est une distance sur $[0, 2\pi[^2$.
3. $$\begin{cases} s_\mathbf{x} + s_\mathbf{y} & \in]0, 2[\\ h_\mathbf{x} & = h_\mathbf{y} \\ l_\mathbf{x} & = l_\mathbf{y} \end{cases}$$

■

d^{hw}_{HLS} vérifie par contre l'inégalité triangulaire. En effet, si nous posons $D : (\mathbf{x}, \mathbf{y}) \mapsto \sum_i \alpha_i \cdot d_i(\mathbf{x}, \mathbf{y})$, avec pour tout i, $\alpha_i \in [0, 1]$ et d_i fonction distance, nous avons immédiatement pour tout $\mathbf{x}, \mathbf{y}, \mathbf{z}$:

$$\begin{array}{rcll} D(\mathbf{x}, \mathbf{y}) & \leq & \left(\sum_{i \neq j} \alpha_i \cdot d_i(\mathbf{x}, \mathbf{y})\right) + \alpha_j \cdot (d_j(\mathbf{x}, \mathbf{z}) + d_j(\mathbf{z}, \mathbf{y})) & \text{car } \alpha_j \geq 0 \text{ et } d_j \text{ distance} \\ & \vdots & & \\ & \leq & \left(\sum_i \alpha_i \cdot (d_i(\mathbf{x}, \mathbf{z}) + d_i(\mathbf{z}, \mathbf{y}))\right) & \\ & \leq & D(\mathbf{x}, \mathbf{z}) + D(\mathbf{z}, \mathbf{y}) & \end{array}$$

■

Trivialement, d^{hw}_{HLS} est symétrique. Par rapport à ce qui précède, d^{hw}_{HLS} est donc non une distance mais un **écart** (parfois appelé « *semi-distance* »).

4.2.1.2. Application à la détection d'objets en mouvement

Voyons à présent l'intérêt d'utiliser la couleur dans des applications courantes, comme par exemple la vidéosurveillance. L'une des partie majeure de la vidéosurveillance consiste à détecter les objets en mouvement dans une scène. Nous nous plaçons dans le contexte de caméras fixes.

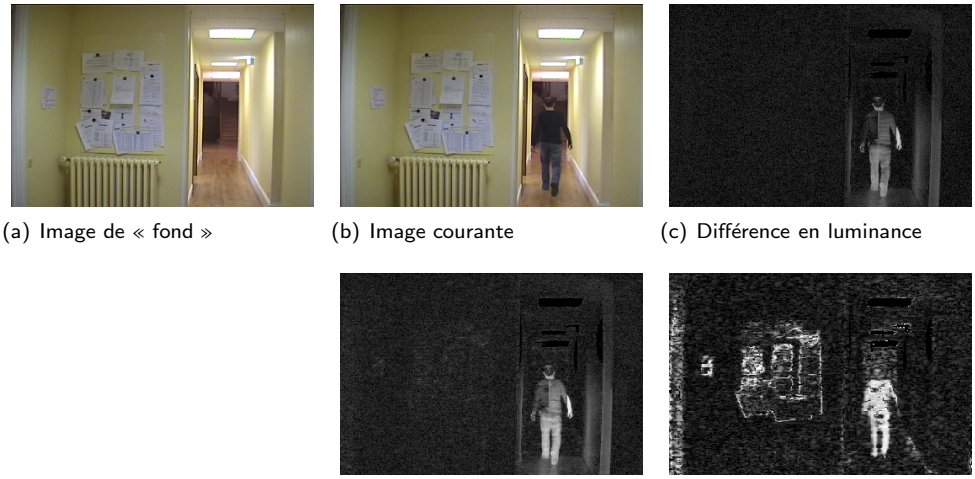
Le principe est le suivant : nous disposons d'une image dite de « *fond* », et de l'image « *courante* » au temps t. Nous noterons ces images respectivement $\mathcal{I}_f$ et $\mathcal{I}^t$. L'image de fond $\mathcal{I}_f$ représente les éléments immobiles de la scène; ainsi par différence d'image avec l'image courante $\mathcal{I}^t$, nous obtenons simplement les objets « ajoutés » sur la partie immobile au temps t. La qualité de l'image de fond conditionne bien sûr grandement les résultats de cette méthode. Sa construction est parfois délicate et peut être coûteuse en mémoire: nous reviendrons sur ce point par la suite et nous supposons pour le moment disposer d'une telle image. Puisque nous avons une notion de distance couleur, nous pouvons effectuer les différences d'images non uniquement dans le canal de luminance comme cela est fait habituellement,

mais directement dans les espaces couleur. Ainsi, si nous utilisons la distance teinte-luminance pondérée dans HLS, l'image des différences $\mathcal{I}^t_{\neq}$ au temps t est définie par :

$$\mathcal{I}^t_{\neq} : \mathbf{x} \mapsto d^{hw}_{\mathsf{HLS}} \left(\mathcal{I}_f(\mathbf{x}), \mathcal{I}^t(\mathbf{x})\right)$$

Cela suppose naturellement que toutes les images sont exprimées dans HLS, ce qui implique une transformation préalable de toutes les images puisque les caméras produisent des images exprimées dans des espaces vidéos RGB ou YUV. Soulignons que la transformation de RGB vers HLS est bijective.

La figure 4.2 illustre ces propos. Dans le canal de luminance, certaines zones « trop » sombres ou claires ne présentent pas suffisamment de contraste. Cette absence se répercute dans l'image de différence. Dans l'espace couleur HLS, bien que le contraste soit faible, il y a néanmoins une information de teinte. La saturation peut être faible également, ce qui entraîne une information dégradée de la teinte: la métrique utilisant uniquement la teinte présente alors beaucoup de bruit et est difficilement exploitable seule (figure 4.2(e)). Si l'on utilise par contre le canal de saturation comme facteur pondérant entre la teinte et la luminance (métrique de l'équation 4.5), ce bruit est fortement réduit comme le montre la figure 4.2(d). Certaines zones sombres possèdent alors un contraste plus élevé par l'utilisation de la métrique en teinte pondérée que par celle en en luminance uniquement. Le résultat de l'image des différences est alors amélioré.



(a) Image de « fond » (b) Image courante (c) Différence en luminance

(d) Différence HLS teinte pondérée (e) Différence HLS en teinte

Fig. 4.2.: Intérêt de la couleur dans le cadre de la détection d'objets en mouvement. (c) est la différence classique en luminance. Le faible contraste entre le pull noir et le fond sombre entraîne la perte de la partie gauche du haut du corps. (d) est la même opération, dans HLS et selon l'équation 4.6. Le contraste entre le pull et le fond est rehaussé (le fond est de couleur beige). Cette métrique réagit aux effets d'ombre portée, puisque la luminance intervient dans l'équation. (e) est l'analogue en teinte uniquement, selon l'équation 4.5. Le contraste est très bon pour la personne et l'impact des ombres portés beaucoup moindre que pour les deux métriques précédentes. Cependant cette métrique est difficile à traiter compte tenu du bruit important dans le canal de teinte.

Nous reviendrons plus en détail sur l'amélioration de la détection par la couleur en §4.5.

4.2.1.3. Gradients et couleur

Le gradient est un outil simple dont nous allons nous servir régulièrement dans la suite. Sa principale application est de mettre en évidence des contours, que l'on considère souvent comme étant la frontière des objets. Les gradients sont représentatifs de la variation d'une grandeur, qui est la plupart du temps en

imagerie scalaire l'intensité lumineuse. On suppose que plus cette variation est élevée, plus la probabilité de changer de zone ou d'objet est grande, ce qui évidement est loin d'être toujours vrai, mais apporte toutefois une approximation très intéressante sur les contours des objets. Le système de vision humain se base d'ailleurs essentiellement sur cette information - la détection des zones de transition - qui couplé à la connaissance et l'expérience, permet de séparer les objets.

Plusieurs types de gradient existent dans la littérature. La définition classique est la variation de la fonction image selon une direction précise, direction souvent confondue avec l'horizontale ou la verticale. Le gradient morphologique introduit par Beucher [Beu91] ne s'appuie non pas sur la variation selon une direction mais dans un voisinage.

Gradient morphologique : La définition du gradient morphologique est très simple, la voici:

Définition 4.1 (Gradient morphologique - Beucher [Beu91])
Le gradient morphologique est le résidu entre le dilaté et l'érodé de l'image. Les demi-gradients morphologiques supérieurs et inférieurs sont respectivement les résidus entre le dilaté et l'identité d'une part, et l'identité et l'érodé d'autre part.

Cette définition suppose bien sûr l'existence de l'érosion et de la dilatation dans l'espace image. Les espaces vectoriels ne bénéficient pas naturellement d'une structure ordonnée, il est toutefois possible de s'astreindre de l'ordre par la remarque suivante. Le dilaté par un élément structurant plan value le centre par le supremum dans le voisinage, l'érodé par l'infimum. Le gradient morphologique est la différence entre ce supremum et cet infimum, donc la différence maximale parmi tous les couples de points présents dans l'élément structurant; les notions d'origine et d'ordre disparaissent. Le gradient morphologique est donc la valeur maximale des modules des différences parmi tous les couples de points présents dans le voisinage. Le module est pris ici en temps que valeur absolue $\cdot \mapsto |\cdot|$. Ceci nous permet de définir le gradient morphologique généralisé de la manière suivante:

Définition 4.2 (Gradient morphologique généralisé)
Soit $\mathcal{I}$ une image de $\mathbf{E}$ dans $\mathbf{F}$ et « d » une distance sur $\mathbf{F}\times\mathbf{F}$. Le gradient morphologique généralisé un point $x \in \mathbf{E}$ est l'extension maximale selon d de $\mathcal{I}$ dans le voisinage de x:

$$|\nabla_g \mathcal{I}| : \left\{ \begin{array}{ccl} \mathbf{E} & \mapsto & \mathbb{R}_+ \\ x & \rightarrow & \vee\{d(\mathcal{I}(x_i),\mathcal{I}(x_j)),\ (x_i,x_j) \in \mathcal{N}_x^2, x_i \neq x_j\} \end{array} \right. \tag{4.7}$$

Un voisinage de taille unitaire donne le gradient morphologique classique, un voisinage de taille plus importante donne l'équivalent d'un gradient morphologique « épais ». La définition 4.2 suppose le calcul des distances entre chaque combinaison de points dans le voisinage, soit une complexité en $O(N \times (N-1))$, N étant la taille du voisinage. La complexité du calcul augmente rapidement avec l'épaisseur désirée: pour un voisinage de taille 5×5, nous avons 600 calculs de distance, pour une boule euclidienne de rayon 10, nous avons ≈ 99000 calculs [(vii)], etc..

L'intérêt des gradients morphologiques généralisés tient dans le fait que l'image gradient obtenue est scalaire: les traitements qui suivent le calcul du gradient sont classiquement ceux issus du cadre scalaire.

La démarche est moins évidente concernant les demi-gradients supérieur et inférieur puisque la notion d'ordre intervient. Si l'on fait abstraction des mots « supérieur » et « inférieur », et selon une démarche analogue à la précédente, le demi-gradient pourrait être défini comme suit:

Définition 4.3 (Demi-gradient morphologique généralisé)
Soit $\mathcal{I}$ une image de $\mathbf{E}$ dans $\mathbf{F}$ et d une distance sur $\mathbf{F}\times\mathbf{F}$. Le demi-gradient morphologique généralisé en un point $x \in \mathbf{E}$ est l'extension maximale selon d de $\mathcal{I}$ entre x et son voisinage:

$$|\nabla_g^{1/2} \mathcal{I}| : \left\{ \begin{array}{ccl} \mathbf{E} & \mapsto & \mathbb{R}_+ \\ x & \rightarrow & \vee\{d(\mathcal{I}(x),\mathcal{I}(x_i)),\ x_i \in \mathcal{N}_x^2\} \end{array} \right. \tag{4.8}$$

[(vii)] À titre indicatif, le calcul d'un tel gradient sur une image 308×238, sur une machine type *Pentium 4*, $2.8GHz$, avec un bus cadencé à $533MHz$, pour une boule euclidienne de 10 pixels de rayon, dure environ 16 mn.

Pour être plus exact, l'analogue scalaire du demi-gradient morphologique généralisé serait le supremum des deux demi-gradients morphologiques. Cette définition rejoint celle d'Angulo [AL03, page 138] pour les gradients en teinte. Il s'agit cependant, pour les raisons exposées, d'un demi-gradient et dont les propriétés sont moins bonnes que le gradient de la définition 4.2. L'intérêt principal du demi-gradient par rapport au gradient complet est le faible nombre de calcul en $O(N-1)$, N étant la taille du voisinage.

Gradients lexicographiques Des structures de treillis ont récemment été définies sur des espaces couleurs: Hanbury [Han02a] propose sur la composante circulaire de HLS les opérateurs élémentaires d'érosion et de dilatation, à partir desquels sont déduits des opérateurs haut niveau; Angulo [AL03] travaille sur des relations d'ordre lexicographique. Enfin, attirons l'attention sur le fait que les deux précédents travaux ont été initiés sous l'impulsion de Jean Serra [IS99, HS01a, HS01b, HS02, ALS03, Ser88].

Les ordres lexicographiques (cf. définition 2.10 page 26) nous permettent de définir trivialement des opérateurs d'érosion et de dilatation. Puisque nous disposons également de fonctions distances couleurs, il est également possible de définir une opération de « module ». En revenant sur la définition originale du (module du) gradient morphologique, la définition de gradients lexicographiques est immédiate.

Le résultat d'un supremum ou d'un infimum selon un ordre lexicographique équivaut à respectivement la dilatation et l'érosion selon un élément structurant plan. Rappelons que la structure algébrique utilisée est un *treillis complet*, ainsi tous les sous-ensembles, ce qui inclut les voisinages, héritent de la structure de treillis complets. Puisque l'infimum et le supremum font partis de l'ensemble de départ, l'introduction de nouvelles couleurs est évité. Ceci est un atout important de ce type de traitement en comparaison à des transformations introduisant de nouvelles valeurs. Les équations de transformation des espaces HLS et Lab n'étant pas linéaire, l'introduction de nouvelles valeurs provoque certains effets très désagréables lors de la transformation inverse vers RGB[(viii)]. Enfin, les gradients lexicographiques ont une complexité équivalente aux gradients classiques, et sont donc *informatiquement* adaptés à des calculs de gradient épais.

4.2.1.4. Méthode d'implémentation des gradients morphologiques.

L'implémentation des gradients et demi-gradients morphologiques généralisés décrits dans la partie précédente est assez simple. Selon l'approche proposée dans le chapitre 2, nous n'avons pas besoin d'écrire la totalité de la fonction de gradient. En effet, il s'agit d'un opérateur de voisinage: sur chaque site, on centre un voisinage et on effectue un calcul ne dépendant que des points du voisinage.

Nous avons déjà proposé un opérateur de voisinage au §2.2.5.2 page 31, par opposition aux *opérateurs de points*. Nous avions alors d'une part précisé l'opérateur du voisinage, d'autre part la fonction appelant cet opérateur pour chaque point de l'image traitée. La fonction appelante possédait alors la particularité d'avoir une arité (nombre de structures en paramètre de la fonction) de 4: en entrées une image, un voisinage et un opérateur, et une image en sortie. L'opérateur par contre possède une arité de 3 : en entrée le voisinage sous forme de paire d'itérateurs, et en sortie le résultat de l'opération.

Nous constatons que cette approche suffit pour le gradient morphologique généralisé. L'algorithme réalisé sur chaque voisinage est présenté en 4.1.

Cet algorithme calcule séquentiellement la distance maximale entre tous les couples de points, ce qui correspond bien à la définition 4.2 du gradient morphologique généralisé. Nous nous servons ici de la propriété de symétrie de la fonction distance, pour éviter un certain nombre de calcul. Ainsi, nous pouvons restreindre le calcul des distances pour les couples tels que $\forall i \in \mathbb{N}_n^*, \forall j \in \mathbb{N}_n^* / j > i$. L'algorithme précédent est implémenté de la manière suivante:

[(viii)] Bien que ceci n'est pas un inconvénient majeur dans le cadre des gradients, puisque l'application de l'opérateur de module résout les éventuelles nouvelles couleurs,

Algorithme 4.1 : Opérateur de gradient morphologique généralisé

Data : $V = \{v_1, \ldots, v_n\}$: ensemble de voisins
Data : d : fonction de distance
$d_M = -\infty$
forall $i \in \mathbb{N}_n^*$ **do**
  forall $j \in \mathbb{N}_n^* / j > i$ **do**
    $d_{current} = d(v_i, v_j)$
    $d_M = d_M \vee d_{current}$
return d_M

```
1  template <class It_, class op_distance> struct s_gradient_morphologique_generalise
2  {
3   typedef typename op_distance::value_type result_type;
4   op_distance func_dist;
5
6   result_type operator()(It_ it , It_ end){
7   result_type d_M = std::numeric_limits<result_type>::min();
8
9   for(; it != end; ++it){
10    const typename It_::value_type pix = *it;
11
12    It_ it2 = it ;
13    ++it2;
14
15    for(; it2 != end ; ++it2){
16        const double dist = func_dist(*it2 , pix);
17        if (d_M < dist) d_M = dist;
18      }
19    }
20    return d_M;
21  }
22 };
```

L'opérateur de voisinage dépend de la classe d'itérateur (comme tous les opérateurs de voisinage) mais également d'une fonction distance (sur l'espace image $\mathbf{F}$), donnée par la classe « op_distance » et qui sera appliquée sur chaque couple de points.

Ici, le domaine de retour de l'opérateur dépend de la fonction distance. Étant donné qu'il s'agit d'une fonction de distance, le domaine est toujours $\mathbb{R}_+$. Nous avons préféré déléguer le type de retour à la classe de distance, à travers l'instruction « typedef typename op_distance:: value_type result_type ; ». Rappelons que le champ « result_type » de cet opérateur sera utilisé par les clients de l'opérateur, entre autres par la fonction appelante.

La ligne « std :: numeric_limits <result_type>::min() » assigne à la variable locale « d_M » la valeur minimale de l'espace de la fonction distance. Nous aurions également pu écrire simplement « result_type d_M = 0; ». Ensuite pour chaque point du voisinage, nous extrayons la valeur du point que nous plaçons dans la constante « pix » (ligne 10). Cette constante sera sollicité pour l'ensemble des points suivants dans le voisinage. Nous reprenons ensuite la position courante de l'itérateur en ligne 12, que nous incrémentons à la ligne suivante. Ceci correspond dans l'algorithme à la création des indexes j pour $j \in \mathbb{N}_n^* / j > i$. Le nombre de point n étant le même pour les indexes j, nous reprenons l'itérateur « end » pour l'arrêt de ces indexes (ligne 15). Le calcul effectif de la fonction distance est effectué à la ligne 16, dont nous gardons la valeur maximale dans « d_M ».

Enfin, cet opérateur est appelé par la même fonction que celle exposée au §2.2.5.2 page 31, utilisé alors pour illustrer l'opération d'érosion par un élément structurant plan. La fonction appelante reste ici parfaitement identique, ce qui montre bien la flexibilité et la puissance de l'approche. L'utilisateur peut alors créer une instance de l'opérateur de gradient morphologique généralisé, et modifier à sa guise la distance utilisée en modifiant la classe « op_distance » de la fonction de voisinage précédente.

L'opérateur de demi-gradient suit la même philosophie et est sensiblement plus simple. L'approche présentée n'est cependant pas réalisable pour le demi-gradient pour la simple raison que l'opérateur de voisinage ne peut connaître le centre du voisinage. En effet, le centre est connu de la fonction appelante, mais seule un voisinage (paire d'itérateurs) est transmis à l'opérateur. La fonction appelante ne peut donc être exactement la même que celle utilisée précédemment. La différence est cependant minime et nous ne la détaillerons pas: elle transmet pour chaque point la valeur du centre à l'opérateur de voisinage. Elle accepte donc des opérateurs de voisinage d'arité 4 : en entrées une paire d'itérateurs pour le voisinage et la valeur du centre, et en sortie la valeur du demi-gradient. Le code devient alors simplement:

```
1  template <class It_, class op_distance> struct s_demi_gradient_morphologique_generalise
2  {
3   typedef typename op_distance::value_type result_type;
4   op_distance func_dist;
5
6   result_type operator()(It_ it , It_ end, typename It_::value_type center){
7     result_type d_M = func_dist(*it , center);
8     ++it;
9
10    for(; it != end; ++it){
11      const double dist = func_dist(*it , center);
12      if(d_M < dist) d_M = dist;
13    }
14    return d_M;
15  }
16 };
```

Une opération en trop est cependant effectuée : lorsque l'itérateur « it » arrive sur le centre, la distance est trivialement nulle. Nous n'avons cependant pas trouvé de moyen simple d'éviter systématiquement ce cas trivial (ce genre de test à l'intérieur de l'opérateur est encore plus coûteux que d'effectuer un calcul supplémentaire), si ce n'est que de supprimer effectivement le centre dans le voisinage (ce qui incombe à la fonction appelante).

4.2.1.5. Résultats des gradients couleurs

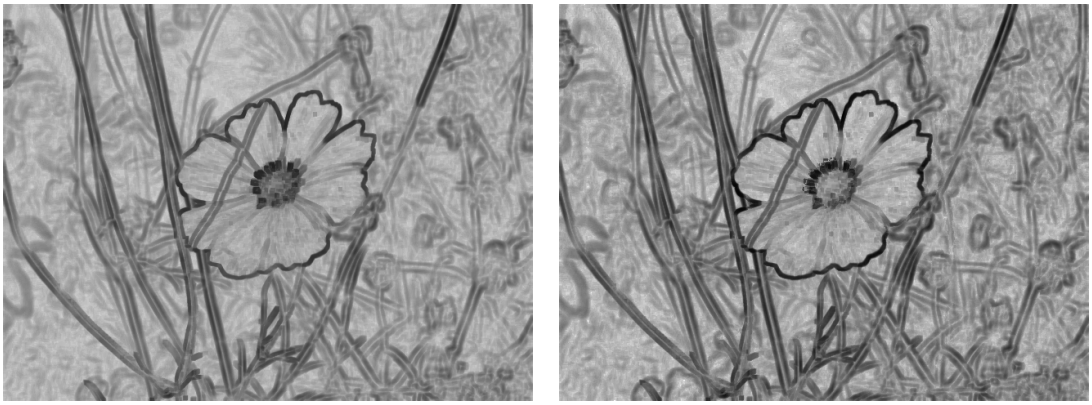
Il serait fastidieux d'exposer les résultats sur la totalité des combinaisons possibles. Nous allons suivre la démarche suivante: nous présenterons dans un premier temps le gradient lexicographique dans l'espace HLS, à l'aide des métriques appropriées. Nous présenterons ensuite le gradient morphologique généralisé dans Lab et HLS. Enfin nous comparerons gradient et demi-gradient morphologiques généralisés. Nous comparerons également au gradient morphologique classique en luminance lorsque cela est pertinent.

Gradients lexicographiques

Il existe deux inconnues majeures pour la définition de ce gradient: le treillis utilisé pour la lexicographie et la métrique utilisée pour le module. Pour cette raison, il est difficile d'extraire le réel apport de la lexicographie.

Dans l'espace HLS, les treillis lexicographiques plaçant la luminance comme espace prioritaire produisent en général des gradients très fidèles au gradient en luminance. Sur des images telles que celles de la figure 4.3, nous notons toutefois des contours plus profonds pour le gradient lexicographique, en partie dûs à la métrique utilisée. Par ailleurs, l'ajout de la teinte (en dernier espace de lexicographie) a un impact totalement négligeable sur le résultat.

Sur des images de qualité inférieure (figure 4.4), les métriques en supremum ont tendance à se rapprocher du résultat du gradient en luminance (figure 4.4(d)), dans la mesure où une grande partie de l'information se trouve dans ce canal. Par ailleurs, les treillis dans lesquels la saturation est prioritaire produisent des contours assez bruités et difficilement exploitables (figure 4.4(e)).



(a) Gradient en luminance

(b) Treillis $\prec_{L,S}$, métrique d^{hw}

Fig. 4.3.: Gradient lexicographique sur l'image « Fleur Roumanie 1 », élément structurant de taille 5×5. L'ajout de la saturation produit des contours plus précis dans les zones où la luminance est homogène (zone de fond notamment). La métrique d^{hw} produit également des contours plus importants.

Gradients morphologiques généralisés

Gradient dans Lab Les figures 4.5 et 4.6 sont deux exemples de résultat du gradient morphologique généralisé pour les distances dans le domaine Lab. Plusieurs définitions (primaires de RGB et température de l'illuminant) de la transformation vers Lab ont été utilisée.

Des définitions différentes conduisent naturellement à des gradients différents. Un intérêt prononcé est donné, à juste titre, à la détermination d'un gradient « meilleur » qu'un autre [(ix)]: la réponse est délicate mais donne souvent la préférence à un espace au détriment d'autres. Nous avorterons toute discussion de ce type et dirons simplement que ces gradients réagissent différemment selon les conditions : par exemple 4.5(c) semble faire apparaître certaines textures qui n'existent pas dans 4.5(b), et les contours sont également plus marqués. Le phénomène inverse apparaît entre ces deux gradients sur la figure 4.6. Les images sont bien-sûr acquises selon un matériel et des conditions incomparables.

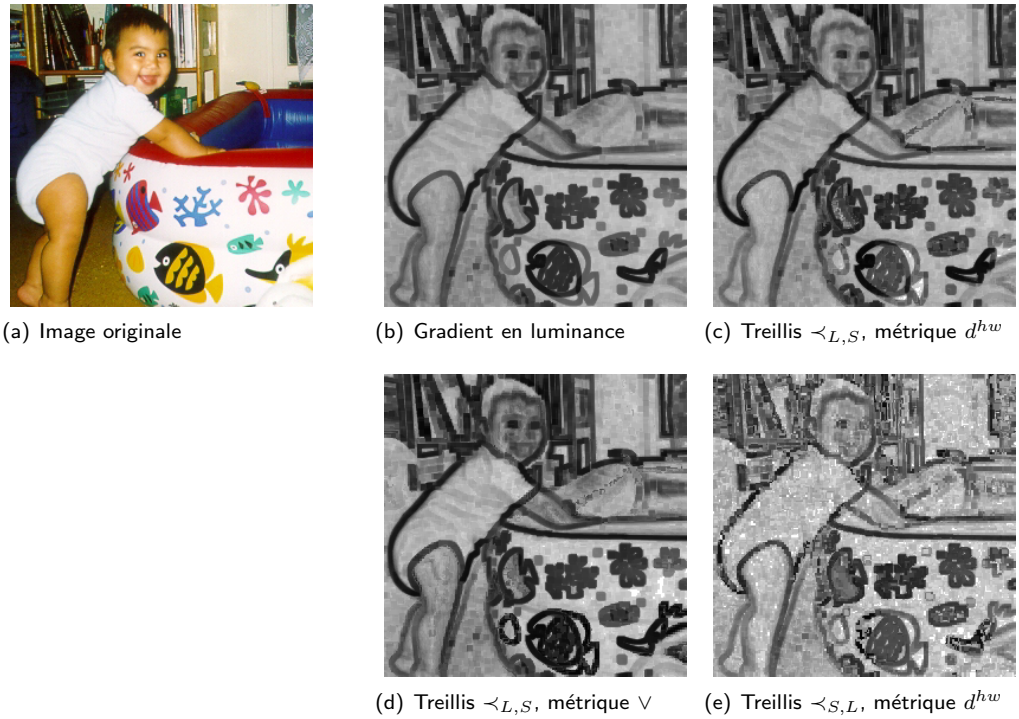
Nous allons à présent apprécier certains résultats liés à la métrique en teinte d^h_{Lab} , donnée par l'équation 4.3. La figure 4.7 présente les résultats sur l'image « Cavalier », ayant subit une compression moyenne de type *JPEG*. De nombreux artefacts apparaissent dans l'image, dûs à l'effet de bloc dans le schéma de compression, et la dégradation de l'information chromatique. Des contours inexistants dans l'image originale apparaissent alors entre les blocs.

L'utilisation des gradients morphologiques généralisés dans les espaces Lab, selon une métrique euclidienne, semble donner de bon gradients, cohérents avec la notion complexe de « contours ». Nous avons toutefois souligné le fait que ces gradients étaient assez sensibles aux bruits de compression, car celle-ci dégrade l'information chromatique. Par ailleurs, le gradient selon la métrique en teinte ne produit de bon contours uniquement sur des images dont l'information chromatique est de qualité.

Bien que l'appréciation de ce gradient n'est que subjective, une difficulté importante inhérente à Lab fut soulevé dans cet exemple: des hypothèses fortes sur l'espace de départ (primaire RGB et blanc de référence) doivent être émises. Nous avons longtemps cherché à déterminer si une représentation était plus stable qu'une autre sur des séquences d'images où la variation d'illuminant est un problème majeur. Nous avons pour cela envisagé des mesures sur la qualité du traitement effectué à partir de ces gradients. Nous ne sommes malheureusement arrivés à aucune conclusion [(x)].

[(ix)] et ce sans même définir la valeur sémantique du terme *meilleur*

[(x)] pour être plus exact, nous avons préféré ne rien conclure plutôt qu'émettre des conclusions fermes, restreintes à des cas très isolés et difficilement reproductibles



(a) Image originale (b) Gradient en luminance (c) Treillis $\prec_{L,S}$, métrique d^{hw}

(d) Treillis $\prec_{L,S}$, métrique $\vee$ (e) Treillis $\prec_{S,L}$, métrique d^{hw}

Fig. 4.4.: Gradient lexicographique sur l'image « Carmen », élément structurant de taille 5×5. (c) : l'ajout de la saturation a pour effet l'apparition de contours inexistants dans le gradient en luminance (b) (l'intérieur du poisson de la piscine, le contour entre le rouge et le bleu sur le coté de la piscine).

Gradient dans HLS L'absence de paramètres colorimétriques (référence de blanc et primaires RGB) dans la transformée a bien un avantage: il soulage l'utilisateur d'hypothèses sur le choix des inconnues. Nous présentons dans cette partie les résultats des gradients morphologiques généralisés selon des distances dans l'espace HLS.

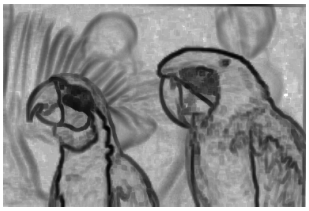
La figure 4.8 illustre le problème d'instabilité de la distance en teinte dans les régions de faible saturation. L'introduction d'un facteur pondérant, selon la saturation, rend par contre le gradient très satisfaisant, puisqu'il corrige les défauts d'un gradient en luminance uniquement, sans toutefois ajouter d'autres problèmes.

Le gradient en teinte simple étant assez bruité, l'utilisation d'une distance par supremum de distances marginales introduit systématiquement ce bruit dans le gradient final: la combinaison de distance par supremum n'est pas satisfaisante.

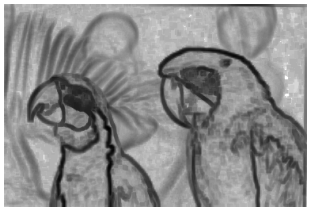
Gradients et demi-gradients morphologiques généralisés

Le demi-gradient morphologique généralisé (équation 4.8) a pour avantage un coût de calcul nettement inférieur au gradient morphologique généralisé (équation 4.7). La qualité des contours qu'il créé est inférieure à celle du gradient morphologique généralisé (complet). Pour se rendre compte de cela, nous devons considérer des tailles de voisinages assez importante. La figure 4.9 présente le demi-gradient en opposition au gradient complet pour un voisinage de type boule euclidienne de rayon 10, que nous notons $B_2(10)$. Nous remarquons pour le demi-gradient le dédoublement de certains contours, phénomène qui, par ailleurs, s'accentue avec la taille du voisinage considéré.

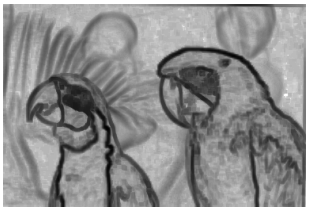
(a) Image originale



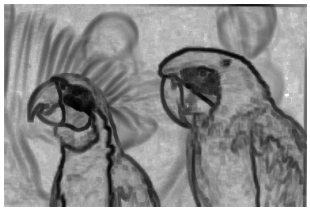
(b) Lab « Adobe 98 »- Illuminant D_{65}



(c) Lab « CIE »- Illuminant E



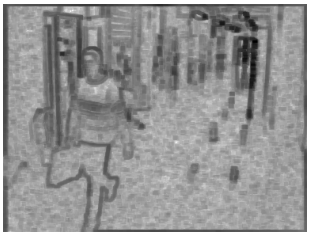
(d) Lab « NTSC »- Illuminant C



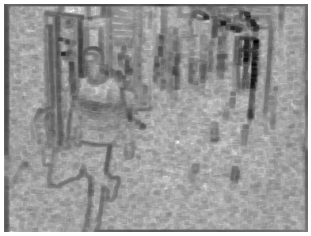
(e) Lab « sRGB »- Illuminant D_{65}

Fig. 4.5.: Gradient morphologique généralisé dans Lab selon la distance euclidienne sur l'image « Parrots ». Les quatre images sont produites selon une transformation différente vers le domaine Lab, selon des suppositions différentes sur les blancs de référence et les primaires RGB concernant l'image originale.

(a) Image originale



(b) Lab « Adobe 98 »- Illuminant D_{65}



(c) Lab « CIE »- Illuminant E

Fig. 4.6.: Gradient morphologique généralisé dans Lab selon la distance euclidienne sur l'image « CEA ». L'image originale est très pauvre en information chromatique.

(a) Image originale (b) Lab « CIE »- Illuminant E (c) Lab « sRGB »- Illuminant D_{65}

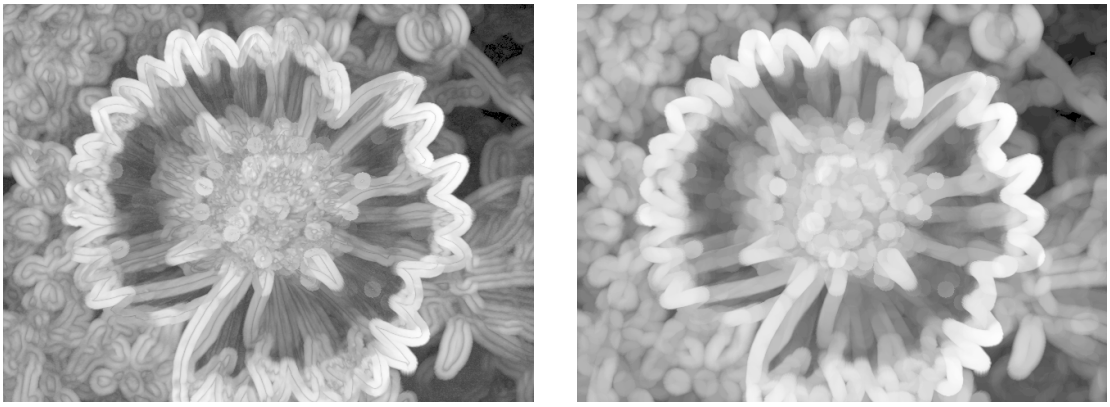
Fig. 4.7.: Gradient morphologique généralisé dans Lab selon la distance en teinte sur l'image « Cavalier ». Les quatre images sont produites selon une transformation différente vers le domaine Lab, selon des suppositions différentes sur les blancs de référence et les primaires RGB concernant l'image originale. L'image originale a subi une compression de type *JPEG*, très visible sur les gradients. L'élément structurant utilisé est de taille 5×5.

(a) Image originale (b) Gradient morphologique classique en luminance (c) Gradient morphologique généralisé, par distance en teinte (d) Gradient morphologique généralisé, par distance pondérée

Fig. 4.8.: Illustration du gradient morphologique généralisé sur l'image « Carmen ». (b) : le gradient en luminance semble correct, mais il manque certains contours qu'il est intéressant de faire apparaître: le contraste entre le bleu et le rouge sur le bord de la piscine ne présente qu'un contour extrêmement faible dans le gradient. (c) : le gradient morphologique généralisé selon une distance angulaire en teinte prononce fortement ce genre de contraste. Cependant certaines régions de saturation faible, telles que le maillot blanc de la petite fille, présentent de nombreuses irrégularité. (d) : le gradient en luminance/teinte pondéré par la saturation concilie les deux remarques précédentes: certains contours forts en teinte mais de luminance équivalente apparaissent, et les régions de faibles saturation restent « stables ».

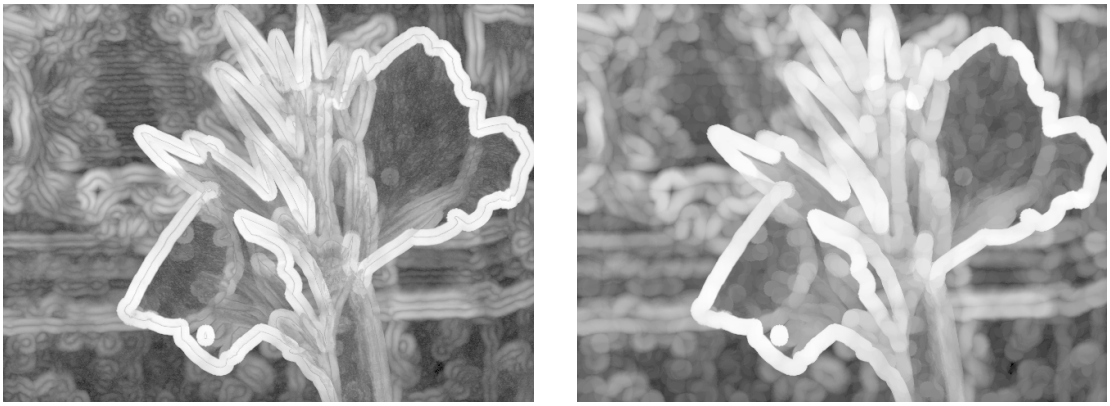
4.2.1.6. Remarque conclusive

Deux définitions de gradient ont été entrevues dans cette partie: des gradients lexicographiques utilisant une approche mixte, algébrique et distance, et des gradients totalement définis selon la notion de distance. Nous avons par ailleurs effectué le rapprochement entre ces derniers, que nous avons nommé *gradients morphologiques généralisés*, et les gradients morphologiques classiques. Les gradients



(a) Demi-gradient $B_2(10)$ - Lab sRGB (b) gradient $B_2(10)$ - Lab sRGB

Fig. 4.9.: Comparaison de gradients et demi-gradient morphologiques pour l'image « Fleur Roumanie ». L'élément structurant utilisé est un disque euclidien de rayon 10.



(a) Demi-gradient $B_2(10)$ - LabsRGB (b) gradient $B_2(10)$ - LabsRGB

Fig. 4.10.: Comparaison de gradients et demi-gradient morphologiques pour l'image « Fleur Budapest ». L'élément structurant utilisé est un disque euclidien de rayon 10.

morphologiques sont en effet basés sur la notion d'accroissement maximal d'une fonction dans un voisinage, notion pouvant être obtenue uniquement par le calcul de distance entre couple de points.

Cependant nous avons évoqué plusieurs problèmes autour du gradient morphologique généralisé. Le plus évident provient de la difficulté que nous avons à justifier d'un espace et d'une métrique. Nous pouvons toutefois guider le lecteur vers les distances teinte/luminance pondéré par la saturation, du fait de sa stabilité.

Le deuxième problème, que nous avons quelque peu étouffé mais qui réduit l'utilisation du gradient morphologique généralisé pour des applications courantes, réside dans le coût de calcul de complexité *quadratique* selon la taille du voisinage. Il est bien sûr possible d'améliorer cette complexité en utilisant une certaine redondance des calculs dans des voisinages proches, au prix d'images supplémentaires et de coût de gestion très probablement importants.

Cela introduit naturellement d'autres notions de gradients, cette fois basées sur des mesures statistiques locales et dont les coûts seraient nettement inférieurs. Nous nous sommes particulièrement intéressé à l'espace HLS, qui nous a semblé stable [(xi)] même exposé à des variations très importantes d'illumination globale de scène. Nous présentons les résultats qui y sont liés dans la partie suivante.

4.2.2. Approche statistique : utilisation sur des espaces cylindriques

L'une des finalités des espaces circulaires est de proposer une représentation plus intuitive de la couleur que les espaces colorimétriques. Cette représentation amène naturellement un problème à cause de la circularité de l'espace. Elle reste néanmoins intéressante et quelques travaux récents (notamment les thèse de Angulo [AL03] et Hanbury [Han02a]) motivent un approfondissement. Nous allons dans cette section étudier quelques mesures statistiques dans l'espace HLS. Ces mesures prennent en compte la circularité de l'information en teinte. Nous proposerons ensuite un gradient couleur, le gradient circulaire d'homogénéité en teinte ou gradient **CH**, basé sur une mesure de dispersion angulaire locale.

4.2.2.1. Statistiques angulaires

Selon toute mesure statistique, nous disposons d'une distribution statistique et d'un n-échantillon d'observation. Nous ne pouvons connaître entièrement les caractéristiques de cette distribution, et nous souhaitons la caractériser à travers certaines mesures. Dans le cadre plus particulier du traitement d'image, le n-échantillon peut par exemple être une région d'intérêt ou un voisinage.

La distribution qui nous intéresse dans cette section est la teinte des pixels. Chaque pixel est considéré comme la réalisation d'une distribution de probabilité en teinte. La teinte du pixel est donc notre variable aléatoire, que nous noterons $(\Theta_k)_{k\in\mathbb{N}_n^*}$. Chaque réalisation Θ_k est une information d'angle dans le domaine $[0, 2\pi[$.

Mesures angulaires : La moyenne et la variance sont des descripteurs statistiques certes basiques, mais souvent suffisants pour certains modèles de distribution. Le calcul de moyenne sur un espace circulaire a une approche différente du linéaire classique. Également par analogie au cadre linéaire, une mesure de concentration des données autour de la moyenne donnera sens à une variance angulaire.

Considérons un n-échantillon $\mathbf{\Theta} = (\Theta_k)_{k\in\mathbb{N}_n^*}$. Le calcul de l'angle moyen de l'échantillon prend en compte la circularité de l'espace de la variable. Un estimateur de l'angle moyen est donné par la formulation suivante [Mar72, page 20]:

$$\bar{\mu_0} = \arg\left[\frac{1}{n}\sum_{k=1}^{n} e^{\mathbf{i}\cdot\Theta_k}\right] \tag{4.9}$$

la barre indiquant que cette mesure est une estimation de la moyenne, et non nécessairement la « véritable » moyenne des (Θ_k) [(xii)]. La démarche est en fait la suivante: on somme des vecteurs de

[(xi)] ce choix est bien-sûr subjectif

[(xii)] la fonction $\arg$ étant l'argument complexe

module unité et d'orientation Θ_k. L'angle moyen $\bar{\mu_0}$ est l'angle que fait le vecteur somme avec l'axe des abcisses. Nous noterons :

$$\bar{R_0} = \frac{1}{n}\left|\sum_{k=1}^{n} e^{\mathbf{i}\cdot\Theta_k}\right| \tag{4.10}$$

la dispersion moyenne du vecteur somme des vecteurs unités de direction Θ_k. La direction moyenne calculée selon l'équation 4.9 est également l'angle minimisant la mesure de dispersion suivante [Mar72, page 22]:

$$S(\theta) = 1 - \frac{1}{n}\sum_{k=1}^{n}\cos\left(\Theta_k - \theta\right) \tag{4.11}$$

La quantité

$$S_0 = S(\mu_0) = 1 - \bar{R_0} \tag{4.12}$$

est appelée la *variance circulaire*.

De la même manière que dans le cas linéaire, des moments ont également été définis. Nous noterons [Mar72, page 36]:

$$\bar{\mu_p} = \arg\left[\frac{1}{n}\sum_{k=1}^{n} e^{\mathbf{i}\cdot p\cdot(\Theta_k - \bar{\mu_0})}\right] \tag{4.13}$$

et

$$\bar{R_p} = \frac{1}{n}\left|\sum_{k=1}^{n} e^{\mathbf{i}\cdot p\cdot(\Theta_k - \bar{\mu_0})}\right| \tag{4.14}$$

les moments centrés d'ordre p. Nous verrons par la suite leur utilité.

Mesures polaires: intervention de la saturation Les mesures précédentes concernent uniquement des distributions d'angle sur le cercle unité. Il est possible de ne considérer non plus le cercle unité, mais la totalité du plan complexe $\mathbb{C}$. La distribution est alors *polaire*. Dans l'espace HLS, cela revient concrètement à utiliser la saturation comme facteur radial, c'est à dire comme élément donnant le module de chaque point dans le plan complexe.

Deux approches sont toutefois envisageables:

- Nous savons que plus la saturation est faible, moins l'information de teinte est pertinente; inversement, plus la saturation est élevée, plus l'information contenu dans la teinte est riche. Si l'on se sert de la saturation comme facteur pondérant le nombre d'occurrences d'une teinte, nous atténuons les teintes qui ne contiennent pas ou peu d'information (ie. de saturation faible). Pour illustrer cette démarche, prenons pour exemple une distribution gaussienne: la densité de probabilité conditionnelle $p(\theta|r)$ peut être exprimée de la manière suivante
  $$-\log p(\theta|r) = \frac{\theta - \theta_0(r)}{\sigma^2(r)}$$
  avec $\theta_0(r) = \theta_0$ angle moyen constant, et $\sigma^2(r)$, fonction décroissante, exprimant l'incertitude de la mesure de θ selon le paramètre r (densité dite *hétéroscédastique*). Donc lorsque r diminue, la fiabilité de la mesure de la moyenne θ_0 diminue également.

- Le deuxième choix considère le couple (teinte, saturation) comme une variable aléatoire bidimensionnelle de type *polaire*: $z(r_k, \Theta_k) = r_k \cdot e^{\mathbf{i}\cdot\Theta_k}$. Le modèle devient rapidement complexe [(xiii)] sans toutefois ajouter des éléments d'intuition.

[(xiii)] cf. [Mar72, page 85] pour le lecteur curieux

Nous choisissons d'introduire la saturation de la première manière: elle remplace alors le paramètre r et joue le rôle de facteur pondérant la fiabilité d'une mesure de teinte. La pondération introduite de la manière suivante, modifie l'angle moyen et la longueur moyenne:

$$\bar{\mu_0}^w = \arg\left(\frac{\sum_{k=1}^{n} s_k \cdot e^{\mathbf{i}\cdot\Theta_k}}{\sum_{k=1}^{n} s_k}\right) \tag{4.15}$$

$$\bar{R_0}^w = \frac{\left|\sum_{k=1}^{n} s_k \cdot e^{\mathbf{i}\cdot\Theta_k}\right|}{\sum_{k=1}^{n} s_k} \tag{4.16}$$

En comparant les histogrammes angulaires et en rose sur des échantillons de peau dans le domaine HLS, nous remarquons que la pondération par la saturation ajoute un élément de précision dans la mesure. La figure 4.11 oppose l'approche classique à la pondération par saturation. Ces schémas donnent le nombre de pixels dans chaque secteur angulaire. Dans la version par pondération, chaque occurrence est quantifié par la valeur de saturation du pixel (et donc n'est plus de 1 comme sur un histogramme normal). Nous constatons sur cet exemple simple, issu d'une personne physique unique, que la saturation améliore la distinction de plusieurs modalités de la teinte.

4.2.2.2. Gradient basé sur la dispersion de teinte

La mesure locale de dispersion semble être une indication pertinente par rapport à l'homogénéité locale des données. Partant des équations 4.11 et 4.16, nous définissons un *gradient circulaire* de la manière suivante:

Définition 4.4 (Gradient circulaire d'homogénéité)
Le gradient circulaire d'homogénéité, par rapport à un voisinage $\mathcal{V}$ de n éléments, est défini comme étant la variance circulaire dans le voisinage $\mathcal{V}$. Il se décline en deux versions: la mesure de la variance circulaire uniquement sur la teinte

$$\nabla_{ch} : x \mapsto 1 - \frac{1}{n}\left|\sum_{p\in\mathcal{V}(x)} e^{\mathbf{i}\cdot\Theta_p}\right| \tag{4.17}$$

et la mesure de la variance circulaire pondérée par la saturation, avec pour ce dernier la convention $\left(\sum_{v\in se} s_v = 0 \Rightarrow S_0 = 0\right)$

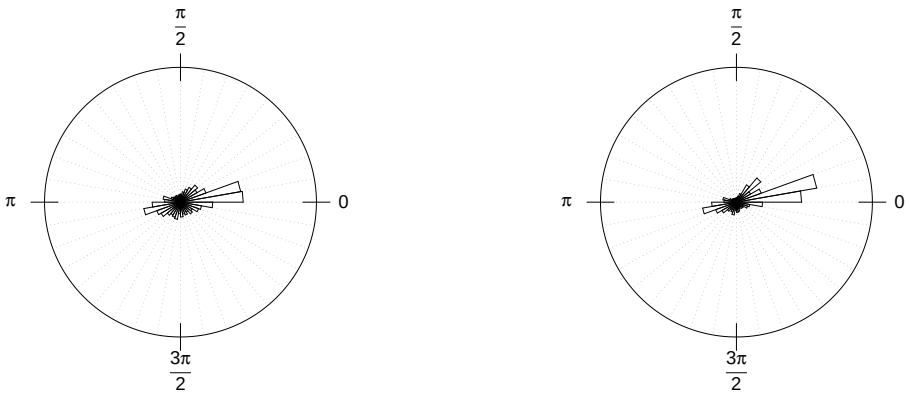
$$\nabla_{chp} : x \mapsto \begin{cases} 1 - \frac{\left|\sum_{p\in\mathcal{V}(x)} s_p \cdot e^{\mathbf{i}\cdot\Theta_p}\right|}{\sum_{p\in\mathcal{V}(x)} s_p} & si \sum_{p\in\mathcal{V}(x)} s_p \neq 0 \\ 0 & sinon \end{cases} \tag{4.18}$$

Résultats : Les résultats sur quelques images de tests sont assez intéressants. Pour chaque image, nous avons calculé les deux gradients à l'aide de deux éléments structurants: le premier hexagonal de taille unitaire, le second euclidien (boule) de taille 10. Les images des figures 4.12, 4.13 et 4.14 ont été prises à l'aide d'un appareil photographique numérique de qualité. L'image 4.15 est extraite d'une base de donnée de visage que nous nommons *Inria*[(xiv)] et enfin l'image 4.16 de la base *Logitech* [(xv)].

La figure 4.12 illustre la pertinence des différents gradients définis: ils sont représentatifs des transitions de teinte. La fleur centrale contraste très bien avec le fond vert, les quelques tiges de teinte jaune et fleur de fond contrastent également, mais à un niveau moindre car les transitions sont moins franches à cause du flou gaussien de la mise au point.

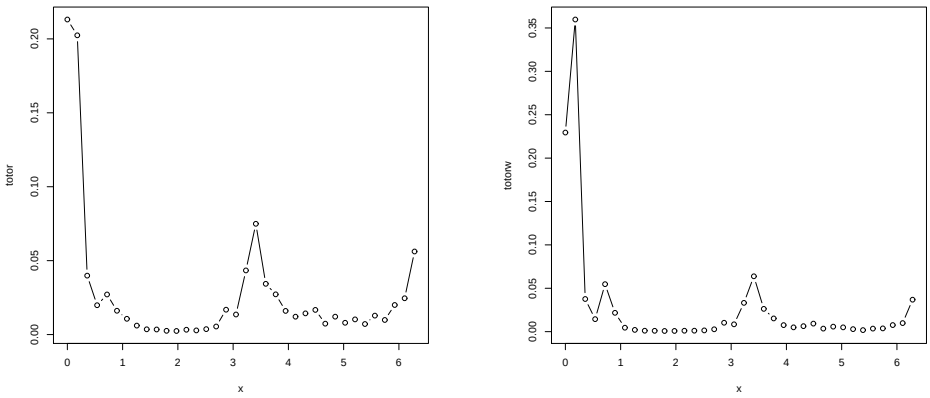
[(xiv)] nous décrirons cette base de manière plus complète dans la partie §4.4.1.2. Considérons pour le moment que cette base est de mauvaise qualité en raison de taux de compression élevé

[(xv)] même remarque que pour la base *Inria*. La qualité est ici meilleure, mais les conditions d'illumination ne sont pas maîtrisées.



(a) Rose sans saturation

(b) Rose avec saturation



(c) Histogramme sans saturation

(d) Histogramme avec saturation

Fig. 4.11.: Utilisation de la saturation comme facteur de pondération de la teinte. Les deux histogrammes de la partie basse représentent le développement des roses (partie haute) sur l'axe des teintes.

Sur la figure 4.13, deux zones majeures sont visibles dans l'image: la première est constituée de la fleur, fortement chromatique, et la seconde est constituée du fond, assez achromatique. Nous constatons encore une fois que les contours produits par le gradient pondéré par la saturation sont plus francs. Par ailleurs, certaines textures du fond provoquent des contours dans le gradient non-pondéré, alors que ces contours sont assez faibles car de chromatisme peu élevé dans le gradient pondéré.

Sur la figure 4.14, nous observons que les contours du gradient de dispersion pondérés sont plus nets que ceux du gradient sans pondération de saturation. L'information de saturation utilisée comme facteur pondérant, dans le sens de représentativité d'une teinte, donne les résultats attendus dans les zones de faible luminance. Rappelons que lorsque la luminance est faible, l'échelle de saturation est réduite compte tenu de la structure double-conique de l'espace HLS.

Sur la figure 4.15, la très faible qualité des canaux de chrominance produit des résultats visuellement déplaisant. La saturation est quasi-inexistante mais n'est pas totalement nulle: le cas $s = 0$ du gradient pondéré est évité et certains contours, manifestement du au bruit numérique, apparaissent sur les aplats du fond. Pour le gradient non pondéré, la faible saturation entraîne de grande fluctuation de teinte. Certains contours apparaissent également au niveau de la bouche et des yeux à l'intérieur du visage, mais ne sont pas suffisamment marqués. Pour cette image, la totalité de l'information se trouve dans la luminance.

Un phénomène très intéressant apparaissant sur la figure 4.16 est la faible influence de l'ombre portée sur le gradient obtenu. En effet, les ombres portées possèdent les mêmes propriétés chromatiques que leur support, et seule la luminance varie.

Remarque d'implémentation : Supposons que nous souhaitions calculer ce gradient sur un élément structurant de type *boule euclidienne* de rayon r. Le calcul des gradients **CH** et **CHP** est le calcul d'une moyenne, or certains éléments de l'élément structurant entre deux voisinages successifs (deux centres voisins) sont communs. Il est possible d'obtenir une complexité de calcul en $O(r)$, réduisant également le nombre de lecture-écriture en mémoire; alors qu'une implémentation naïve donnerait un nombre de lecture-écriture en $O(r^2)$ (ie. nombre d'éléments dans la boule). L'idée est la suivante: séparons l'élément structurant en deux parties selon son sens de déplacement dans l'image:

- la partie entrante est l'ensemble des pixels entrants dans l'élément structurant lorsque celui-ci se déplace d'une unité. Il faut ajouter la valuation de ces points à la somme totale contenu dans l'élément structurant.
- la partie sortante est l'ensemble des pixels sortants dans l'élément structurant lorsque celui-ci se déplace d'une unité. Il faut supprimer la valuation de ces points à la somme totale contenu dans l'élément structurant.

L'union de la partie entrante et sortante est égale à la périphérie de l'élément structurant (cf. figure 4.17). Nous voyons donc que le nombre de lecture-écritures nécessaires est de l'ordre de $\lceil 2\pi r \rceil$, et de l'ordre de $\lceil \pi r^2 \rceil$ pour l'approche naïve (donc intéressante pour $r > 2$). La complexité du calcul du gradient tombe donc de $O(N)$ à $O(\sqrt{N})$, N étant le nombre de points dans l'élément structurant.

Combinaison avec des gradients scalaires : Il est possible de combiner ce nouveau gradient avec un gradient scalaire défini classiquement, l'objectif étant d'injecter des informations issues des contours en chrominance dans un gradient dépourvu de ce type d'information (luminance et/ou saturation).

Sur la figure 4.18, nous avons affiché les gradients circulaires **CH** et **CHP**, que nous avons ensuite combinés avec un gradient morphologique scalaire ∇_s, pour lequel s représente soit la luminance, soit la saturation dans l'espace couleur HLS. La combinaison est effectuée de la manière suivante:

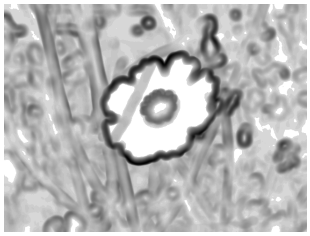
$$\nabla_{comb.} = \nabla_s \cdot \big(1 + \nabla_{ch(p)}\big) \tag{4.19}$$

où « $\cdot$ » est le produit pixel à pixel. De cette manière, le gradient final est rehaussé sur les lignes de contours de luminance ou saturation, lorsque ces contours marquent également un contour au sens de **CH** ou **CHP**. Le fait d'ajouter 1 au gradient circulaire avant recombinaison a pour objectif d'éviter de

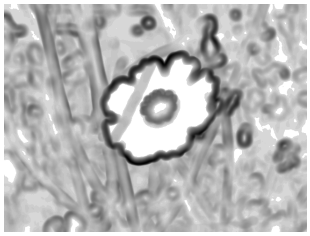
(a) Image originale

(b) ∇_{ch} - Élément structurant hexagonal

(c) ∇_{chp} - Élément structurant hexagonal



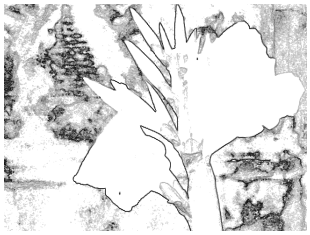
(d) ∇_{ch} - Élément structurant boule euclidienne de taille 10 pixels



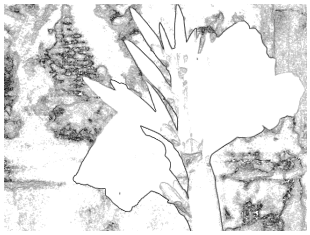
(e) ∇_{chp} - Élément structurant boule euclidienne de taille 10 pixels

Fig. 4.12.: Illustrations des gradients de dispersion - Fleur 1

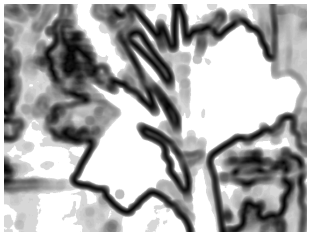
(a) Image originale



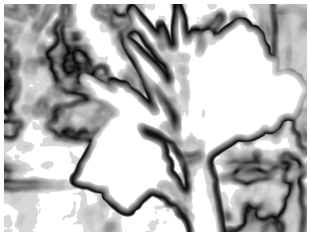
(b) ∇_{ch} - Élément structurant hexagonal



(c) ∇_{chp} - Élément structurant hexagonal



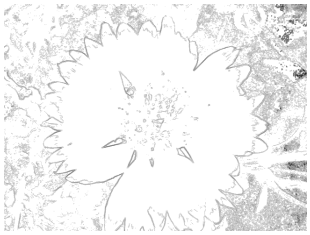
(d) ∇_{ch} - Élément structurant boule euclidienne de taille 10 pixels



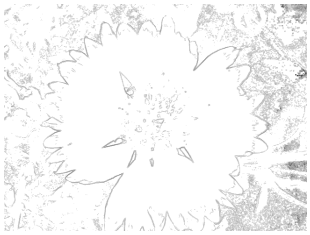
(e) ∇_{chp} - Élément structurant boule euclidienne de taille 10 pixels

Fig. 4.13.: Illustrations des gradients de dispersion - Fleur Budapest

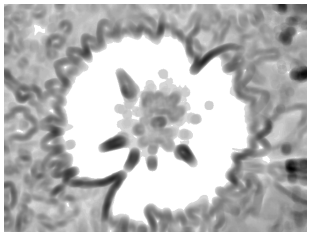
(a) Image originale



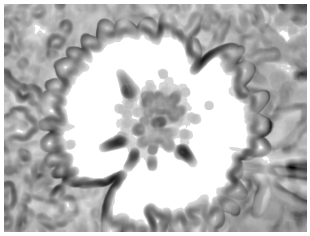
(b) ∇_{ch} - Élément structurant hexagonal



(c) ∇_{chp} - Élément structurant hexagonal



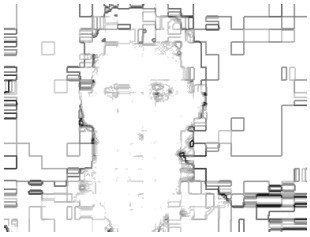
(d) ∇_{ch} - Élément structurant boule euclidienne de taille 10 pixels



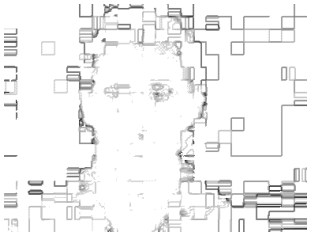
(e) ∇_{chp} - Élément structurant boule euclidienne de taille 10 pixels

Fig. 4.14.: Illustrations des gradients de dispersion - Fleur 2. Le gradient pondéré par la saturation, avec un élément structurant de taille 10 ((d) et (e)), fait davantage apparaître le contraste au niveau des extrémités des pétales - le jaune à la fois avec le rouge de l'intérieur de la fleur et le vert de fond. Le gradient pondéré réagit également mieux dans les zones où la luminance est faible (zones vertes, aux extrémités supérieures droites des images).

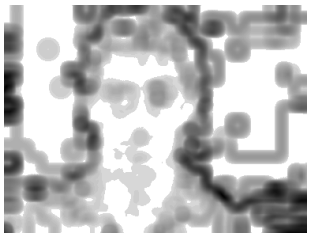
(a) Image originale



(b) ∇_{ch} - Élément structurant hexagonal



(c) ∇_{chp} - Élément structurant hexagonal



(d) ∇_{ch} - Élément structurant boule euclidienne de taille 10 pixels

(e) ∇_{chp} - Élément structurant boule euclidienne de taille 10 pixels

Fig. 4.15.: Illustrations des gradients de dispersion - Extrait base *Inria*. La faible importance du contenu chromatique sur la totalité de l'image produit un gradient chromatique fragile: certains contours manifestement liés au bruit apparaissent dans le fond, le contraste à l'intérieur du visage est peu visible.

(a) Image originale

(b) ∇_{ch} - Élément structurant hexagonal

(c) ∇_{chp} - Élément structurant hexagonal

(d) ∇_{ch} - Élément structurant boule euclidienne de taille 10 pixels

(e) ∇_{chp} - Élément structurant boule euclidienne de taille 10 pixels

Fig. 4.16.: Illustrations des gradients de dispersion - Extrait base *Logitech*. Il est intéressant de remarquer la faible influence de l'ombre portée (visage) sur le gradient, ce avec ou sans pondération de teinte, et sur les deux éléments structurant utilisés.

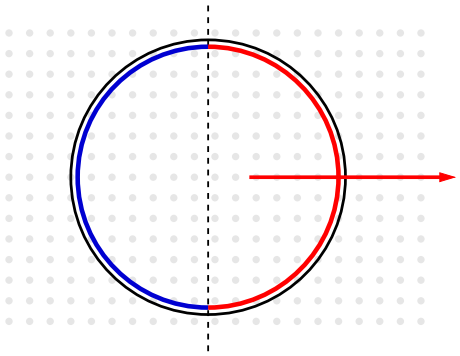
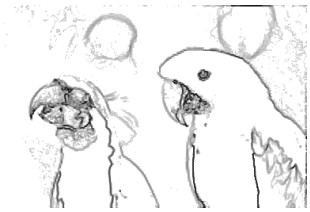


Fig. 4.17.: Implémentation de la moyenne locale pour le gradient de dispersion. Les parties rouge $\mathcal{P}_R$ et bleue $\mathcal{P}_B$ de la boule représentent les pixels devant être lu à chaque déplacement horizontal et vers la droite (respectivement après et avant déplacement). Les sommes pour le calcul du gradient sont alors simplement mise à jour par rapport à la somme précédente par l'ajout des pixels $\mathcal{P}_R$ et la soustraction des pixels $\mathcal{P}_B$.

« casser » certains contours du gradient scalaire. Le fait de pondérer le gradient scalaire par le gradient circulaire a pour effet de ne pas faire apparaître des lignes de contours inexistantes dans le gradient scalaire.

Bien que le résultat de cette figure soit très épais, nous constatons néanmoins que le gradient **CHP** produit une meilleure recombinaison au sens de l'équation 4.19: les contours sont plus nets et francs pour le gradient **CHP** que pour le gradient **CH**. La recombinaison avec le gradient scalaire accentue effectivement les contours: par exemple sur l'image 4.18(k), le contour entre le perroquet de droite et le fond (sur son flanc gauche) est plus important que pour le gradient en saturation uniquement (figure 4.18(i)).

(a) Image originale

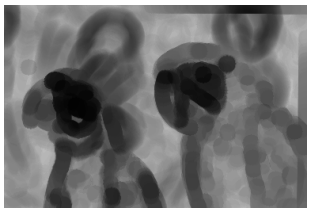


(b) ∇_{ch} - Élément structurant hexagonal

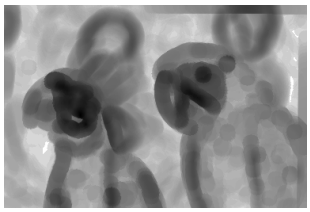
(c) ∇_{chp} - Élément structurant hexagonal

(d) ∇_{ch} - Élément structurant boule euclidienne de taille 10 pixels

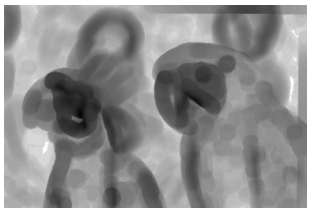
(e) ∇_{chp} - Élément structurant boule euclidienne de taille 10 pixels



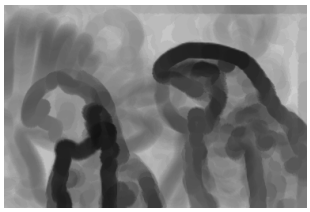
(f) ∇_L - taille 10 - Gradient morphologique en luminance



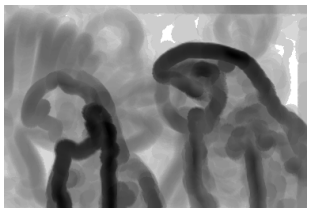
(g) ∇_{ch} - taille 10 - Combinaison avec le gradient en luminance (f)



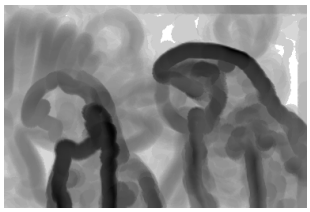
(h) ∇_{chp} - taille 10 - Combinaison avec le gradient en luminance (f)



(i) ∇_S - taille 10 - Gradient morphologique en saturation



(j) ∇_{ch} - taille 10 - Combinaison avec le gradient en saturation de (i)



(k) ∇_{chp} - taille 10 - Combinaison avec le gradient en saturation de (i)

Fig. 4.18.: Illustrations des gradients de dispersion - Parrots

4.2.2.3. Conclusion

Dans cette partie, nous avons vu certaines statistiques angulaires, desquelles nous avons dérivé une mesure d'homogénéité. Cette mesure a la bonne propriété de présenter des coûts de calcul croissant linéairement selon le nombre d'éléments. Il est même possible dans certains cas d'utiliser judicieusement la redondance des calculs de deux voisinages successifs pour amener ce coût à une croissance en racine carrée par rapport au cardinal du sous-ensemble considéré.

Cette mesure a ensuite été appliquée dans la définition de deux gradients dans l'espace couleur HLS, que nous avons nommé gradient circulaire d'homogénéité, pondéré ou non par l'information de saturation (respectivement gradient **CH** et **CHP**). Nous avons observé que ces gradients réagissaient correctement aux problèmes courants liés aux ombres portés, du fait même de la nature des ombres portées [(xvi)]. La faible complexité du calcul rend possible l'utilisation d'éléments structurants de taille importante. Le cas unitaire ne permet pas d'avoir, sauf pour les contours très prononcés, des mesures statistiques fiables.

4.2.3. Récapitulatifs sur les gradients couleurs

Dans cette partie, plusieurs points de l'approche des données multispectrales ont été entrevus. Après un récapitulatif succinct des différentes méthodes connues et existantes (marginale, vectoriel, algébrique), nous avons envisagé deux solutions majeures: l'approche de certains outils de morphologie mathématique exprimés selon des fonctions distances, et une approche selon une mesure statistique locale.

Le premier point nous a permis d'exprimer des opérateurs d'accroissement locaux, plus précisément le gradient morphologique, sous une forme métrique, et a donné naissance à la notion de gradient morphologique généralisé. Ce gradient reprend les récentes recherches annexes pour en extraire une définition conforme au cas scalaire classique.

L'expression de cet opérateur sous forme métrique contourne habilement le problème de la définition d'une origine de l'espace de travail: le gradient est exprimé uniquement par un supremum des distances entre les couples de points du sous-ensemble considéré. Nous avons cependant soulevé deux problèmes majeurs. Le premier est inhérent aux espaces colorimétriques, et provient de la multiplicité des définitions possibles pour les transformations vers Lab. Ces définitions différentes produisent à leur tour des gradients différents, dont l'impact sur les traitements futurs est difficile à prévoir. L'espace HLS ne présente pas cet inconvénient puisque les définitions des transformations ne sont pas fonctions de la scène. Le deuxième problème est d'ordre pratique: le coût de calcul d'un tel gradient présente une croissance quadratique selon le nombre d'éléments du voisinage considéré. Bien qu'il est possible de réduire ce coût, ce gradient n'est pas utilisable pour des voisinages de taille importante (au delà de une à deux fois le voisinage unité).

Nous nous sommes donc ensuite intéressés à des mesures statistiques pertinentes dans les voisinages. Nous avons présenté un cadre dédié aux informations de type circulaire de manière à utiliser l'information en teinte de l'espace HLS. L'utilisation d'une mesure de dispersion par rapport à l'angle moyen dans le voisinage nous a semblé être une mesure pertinente de variation locale. Nous en avons déduit deux définitions de gradients selon l'homogénéité circulaire locale : le premier en ne prenant compte que de l'information en teinte, le second en considérant la saturation comme mesure de pertinence de l'information de teinte. Cette mesure présente un coût croissant linéairement en fonction du cardinal de l'ensemble considéré, coût pouvant être réduit à la racine carré de ce cardinal selon certaines configurations. Ceci en fait un candidat adapté à des calculs de gradients épais dans le domaine couleur. Par ailleurs, ces gradients peuvent compléter les gradients scalaires classiques par des moyens simples de recombinaison.

Nous allons à présent aborder l'aspect algébrique des traitements morphologiques, par l'utilisation de treillis lexicographiques. L'approche est complètement différente de celles présentées jusqu'ici [(xvii)]:

[(xvi)] Une ombre ne présente pas de variation chromatique le long du gradient en luminance, cf. [Ris01]
[(xvii)] sauf peut-être en ce qui concerne le gradient lexicographique

la structure de treillis complet permet en effet d'une part d'étendre naturellement les opérateurs classiques, d'autre part d'utiliser le cadre algorithmique développé pour le cadre classique, sans ajout d'une quelconque complexité dans les calculs.

4.3 Approche algébrique : ordres lexicographiques

Dans cette partie, nous nous intéressons à l'approche algébrique pour le traitement des images couleur. L'approche que nous proposons d'étudier est celle des treillis lexicographiques. Ces treillis sont des structures sur des ensembles de type $\mathbf{F} = \mathbf{F}_1 \times \mathbf{F}_2 \ldots \times \mathbf{F}_n$ munis d'une relation d'ordre lexicographique $\preceq_{\mathbf{F}}$ sur $\mathbf{F}$. La relation précédente permet de définir les opérateurs $\veebar$ et $\barwedge$, comme nous l'avons vu dans la proposition 2.1 (chapitre 2 page 27). Rappelons que les treillis lexicographiques préservent la structure **complète** [(xviii)] des treillis sur lesquels ils sont définis, ce qui permet de rester dans le même cadre que le cas classique scalaire. Le cadre algébrique restant le même, l'extension des algorithmes classiques à ces nouveaux treillis est alors possible.

Les relations d'ordre lexicographiques mettent à disposition un moyen très simple pour définir une structure de treillis sur un espace vectoriel. Nous avons déjà vu quelques exemples sur les érosions et dilatations lexicographiques lors de la définition de la quasi-distance lexicographique (§3.3.3.2 page 93). Les érosions et dilatations géodésiques sont définis de manière identique à leur homologue « classique »-binaire ou scalaire. Nous allons nous intéresser plus particulièrement à la reconstruction, qui est un opérateur extrêmement puissant. Après une brève définition, nous donnerons l'algorithme classique de la reconstruction. Nous verrons ensuite comment tirer profit du cadre informatique introduit au chapitre 2 pour définir ces algorithmes dans le cadre lexicographique. Nous utiliserons ensuite ce résultat dans la définition des granulométries lexicographiques.

4.3.1. Reconstruction lexicographiques

Puisque la notion de *géodésie* est définie de manière algébrique, elle s'étend naturellement aux treillis lexicographiques. Transposer les opérateurs d'érosion et de dilatation géodésique sont des tâches immédiates grâce au cadre que nous avons développé. Nous allons plus particulièrement nous intéresser à l'algorithme de reconstruction introduit par L. Vincent [Vin93], basé sur des files d'attente hiérarchiques. Cet algorithme est également rapidement transposable dans le cas lexicographique.

L'intérêt de la reconstruction repose en outre sur la préservation des structures marquées. Elle sera intéressante par la suite pour la séparation d'objets d'intérêt. La différence par rapport à l'approche classique est que nous disposons d'une information couleur dont nous souhaitons profiter.

4.3.1.1. Définition

La reconstruction que nous nommerons « classique », au sens initial de Lantuéjoul & Maisonneuve [LM84] est donnée par la définition 4.5 suivante:

Définition 4.5 (Reconstruction - [LM84])
Soit $(\mathbf{M}, \mathbf{S}) \in \mathbf{F}^2$ *deux ensembles. Nous appellerons* $\mathbf{M}$ *le marqueur, et* $\mathbf{S}$ *le masque. Soit* $\delta_{\mathbf{S}}^{(i)}$ *la dilatation géodésique à l'intérieur de* $\mathbf{S}$ *par un élément structurant unitaire. L'ouverture par reconstruction* $\gamma_{\mathbf{S}}^{\infty}$ *de* $\mathbf{S}$ *à partir de* $\mathbf{M}$ *est définie comme étant la limite suivante:*

$$\gamma_{\mathbf{S}}^{\infty}(\mathbf{M}) = \lim_{i \to +\infty} \delta_{\mathbf{S}}^{(i)}(\mathbf{M}) \tag{4.20}$$

La fermeture par reconstruction $\varphi_{\mathbf{S}}^{\infty}(\mathbf{M})$ *de* $\mathbf{S}$ *à partir de* $\mathbf{M}$ *est définie de manière duale:*

$$\varphi_{\mathbf{S}}^{\infty}(\mathbf{M}) = \overline{\gamma_{\overline{\mathbf{S}}}^{\infty}(\overline{\mathbf{M}})} \tag{4.21}$$

En l'absence de précision, nous ferons référence à l'ouverture par reconstruction. Par extensivité de la dilatation, la suite $(\delta_{\mathbf{S}}^{(i)}(\mathbf{M}))_i$ est croissante. L'ensemble $\mathbf{S}$ est supposé être à support borné, ce qui par géodésie induit la convergence de la suite (croissante et majorée). Puisque la taille de l'élément

(xviii) cf. définition 2.13 page 27 : si chaque $\langle \mathbf{F}_i, \preceq_i \rangle, i \in \mathbb{N}_n^*$ est complet, alors le treillis défini par l'ensemble $\mathbf{F} = \mathbf{F}_1 \times \ldots \times \mathbf{F}_n$ muni d'une relation lexicographique d'ordre sur les $\preceq_i$ est également complet.

structurant de la dilatation est la même pour tous les indices, et par convergence de la suite, nous avons

$$\exists N \in \mathbb{N}^* \setminus \delta_{\mathbf{S}}^{(j)}(\mathbf{M}) = \delta_{\mathbf{S}}^{(N)}(\mathbf{M}),\ j > N$$

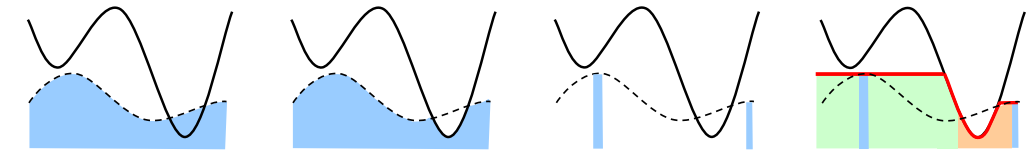
La reconstruction de l'équation 4.20 est une ouverture algébrique de $\mathbf{S}$ car elle est:
- anti-extensive : $\delta_{\mathbf{S}}^{(i)}(\mathbf{M}) \subset \mathbf{S}$
- croissante : $\mathbf{S}_1 \subset \mathbf{S}_2 \Rightarrow \delta_{\mathbf{S}_1}^{(i)}(\mathbf{M}) \subset \delta_{\mathbf{S}_2}^{(i)}(\mathbf{M})$
- idempotente (en utilisant la convergence précédente):

$$\begin{aligned}
\gamma_{\mathbf{S}}^{\infty} \circ \gamma_{\mathbf{S}}^{\infty}(\mathbf{M}) &= \lim_{i\to+\infty} \delta_{\mathbf{S}}^{(i)} \circ \gamma_{\mathbf{S}}^{\infty}(\mathbf{M}) \\
&= \lim_{i\to+\infty} \delta_{\mathbf{S}}^{(i)} \circ \delta_{\mathbf{S}}^{(N)}(\mathbf{M}) && \text{(selon les remarques précédentes)} \\
&= \lim_{i\to+\infty} \delta_{\mathbf{S}}^{(N+i)}(\mathbf{M}) \\
&= \lim_{i\to+\infty} \delta_{\mathbf{S}}^{(i)}(\mathbf{M}) && \text{(définition de la limite)} \\
&= \delta_{\mathbf{S}}^{(N)}(\mathbf{M}) && \text{(par définition de } N\text{)} \\
&= \gamma_{\mathbf{S}}^{\infty}(\mathbf{M})
\end{aligned}$$

La méthode naïve serait de faire N itérations de la dilatation géodésique. Ne connaissant pas N, il faut calculer la suite des dilatés géodésiques et tester l'idempotence à chaque itération. Cependant, dilater tous les points du marqueur est à la fois coûteux et inutile. Nous allons donc voir un algorithme de reconstruction efficace, basé sur des files d'attente hiérarchiques.

4.3.1.2. Algorithme à base de files d'attente hiérarchiques

L. Vincent [Vin93] a proposé un premier algorithme à base de files d'attente « simples ». Une implémentation à base de files d'attente hiérarchiques a ensuite été donnée dans [Mey95]. Il repose sur la remarque suivante: il est possible de reconstruire entièrement le marqueur sous (sur) le masque à partir des maxima (minima) locaux du marqueur. Les dilatations successives des extrema du marqueur permettent d'obtenir la reconstruction attendue, ce qui a pour avantage un volume de points traités moins important ainsi qu'un traitement minimal des points. En effet, ordonner le traitement à l'aide de la *fah* permet, lorsqu'un point est retiré de celle-ci, d'obtenir la valeur désirée du point sans traitement supplémentaire.



(a) Marqueur et masque (b) Intersection (c) Extremum (d) Propagation

Fig. 4.19.: Illustration de la reconstruction. (a) Marqueur initial (pointillé) et masque (trait plein). (b) : Intersection entre le marqueur et le masque. (c) : recherche d'extremum dans le résultat de (b). (d) : dilatation géodésique successive à l'aide d'un front de propagation. Les marqueurs bleus sont dilatés à l'intérieur du masque. Lorsque le front du marqueur de gauche arrive dans la zone orange, le marqueur de droite commence à se dilater également.

Les étapes de l'algorithme sont résumées sur la figure 4.19, l'algorithme lui-même est donné en 4.2 page 133. La fermeture par reconstruction est déduite par dualité des opérations élémentaires de la figure 4.19.

Description de l'algorithme 4.2 Dans cet algorithme, nous avons une image de marqueur « imMarqueur », une image de masque « imMasque », une image de sortie « imSortie » et un objet de files d'attente hiérarchiques « *fah* ».

• L'algorithme initialise l'image de sortie en prenant l'infimum du marqueur et du masque, de manière à imposer la géodésie (marqueur sous le masque). L'image marqueur ne sera plus utilisée par la suite, puisque le contenu utile se trouve à présent dans l'image de sortie. Ensuite, l'algorithme recherche les maxima de cette nouvelle image, et tous les points maxima sont insérés dans la *fah* à la priorité donnée par l'image de sortie. Les points de la *fah* sont ordonnés à priorité croissante selon le niveau de gris croissant. Ainsi, en dépilant un point de la *fah*, nous sommes sûrs d'effectuer une dilatation.

• Chaque point p retiré de la *fah* devient le siège d'une propagation (ou dilatation). Cette propagation est effectuée par une extension de la valeur de p sur son voisinage. Cette extension est contenue entre les lignes 19 et 22. Chaque voisin v qui n'est ni dans la *fah*, ni déjà traité (test ligne 20) est marqué comme étant dans la *fah*. La valeur de p est utilisée pour l'insertion des voisins dans la *fah*, et la ligne 22 force la géodésie selon le masque.

• Chaque point retiré de la *fah* est marqué définitivement (ligne 17). En effet, grâce à l'ordre total de la *fah*, les points extraits de la *fah* sont nécessairement à valeur décroissante (décroissance au sens large $\leq$). Si un point p était retiré plusieurs fois de la *fah*, il le serait également avec des valeurs décroissantes. Lors d'une seconde extraction, le point p aurait une valeur égale ou inférieure à celle qu'il possédait lors de la première extraction.

Dans le cas d'une égalité, un second traitement n'apporterait rien de nouveau. En effet, p est la base d'une dilatation. Le premier traitement place les voisins de p dans la file d'attente hiérarchique de manière à utiliser la valeur de p et le masque. Or le masque ne change pas de valeur, ce qui implique l'inutilité du second traitement.

Dans le cas d'une décroissance stricte, cela signifierait que le point p possède deux voisins, l'un étant au dessus de l'autre. Par dilatation, le second traitement ne serait par pris en considération.

Donc dans les deux cas, le second traitement est inutile, ce qui explique le marquage définitif de la ligne 17.

• La ligne 18 est utile dans le cas où un maxima M_1 serait dilaté sur un autre M_2, et telle que $M_1 > M_2$ et relié par un chemin de hauteur toujours supérieure strictement à M_2. Dans ce cas, M_2 n'aura jamais été dilaté à cause de l'ordre total de la *fah*. Bien que M_2 soit également dans la *fah*, sa valeur réelle d'insertion dans la *fah* n'était pas sa valeur définitive; ce qui explique le supremum de la ligne 18, ainsi que le test de la ligne 16.

• Enfin toujours grâce à l'ordre total donné par la *fah*, aucun voisin n'est inséré deux fois dans la *fah*. L'explication de ceci est assez similaire au marquage définitif des points dépilés. Puisque le centre p est le centre le plus « haut », et que le masque ne change pas, la valeur d'insertion de v dans la *fah* (ligne 22) est la valeur maximale possible. Ce qui justifie son traitement unique, et le marquage de la ligne 21.

Relation d'ordre & méta-programmation Nous constatons sur cet algorithme qu'une relation d'ordre ainsi que son dual interviennent à trois niveaux:

1. dans l'algorithme, par l'appel des opérateurs $\veebar$ et $\overline{\wedge}$.
2. par l'appel à la fonction d'extraction des extrema.
3. dans la notion de *hiérarchie* des files d'attente hiérarchiques.

Selon nos développements sur les ordres et grâce à la méta-programmation, nous pouvons déduire à partir d'un ordre, son ordre dual. Cela se fait simplement par la ligne suivante, en supposant « predicate » comme étant l'ordre normal:

```
1 // Accès à l'ordre dual de predicate
2 typedef typename predicate::dual predicate_dual;
```

Algorithme 4.2 : Reconstruction à base de files d'attente hiérarchiques

```
   Data : imMarqueur, imMasque
1    ▷ Le marqueur et le masque sont de type scalaire
2  return imSortie
3    ▷ L'image de sortie est du même type que marqueur

4  ▶ Initialisation
5  fah ← ∅
6  imageTravail ← non-traité
7  imSortie ← imMarqueur ⊼̄ imMasque
8  maximas ← Maximas(imSortie)
9  forall p ∈ maximas do
10 |  fah ← ajouter p à la priorité imSortie(p)
11 end

12 ▶ Propagation
13 while fah ≠ ∅ do
14 |  val, p ← priorité la plus élevé de fah et position correspondante
15 |  fah ← retirer (val, p)
16 |  if imageTravail(p) = traité then passer au point suivant de la fah
17 |  imageTravail(p) = traité
18 |  imSortie(p) = val ⊻̲ imSortie (p)
19 |  forall v ∈ N_p \ p do
20 |  |  if imageTravail(v) = non-traité then
21 |  |  |  imageTravail(v) = empilé
22 |  |  |  fah ← ajouter v à la priorité val ⊼̄ imMasque (v)
23 |  |  end
24 |  end
25 end
```

Il est possible d'utiliser les principes de la méta-programmation pour capitaliser les algorithmes duaux. Deux solutions sont possibles: soit nous utilisons un patron d'algorithme construit sur l'algorithme 4.2, soit nous plaçons la relation d'ordre en paramètre de la reconstruction.

La première méthode (patron d'algorithme) nécessite la démarche suivante: toutes les étapes impliquant l'ordre sont exportées dans un objet de contrôle, extérieur à l'algorithme. Le patron algorithmique est le même pour l'ouverture et la fermeture par reconstruction: chaque appel explicite à une fonction impliquant l'ordre est remplacé par l'équivalent de l'objet de contrôle. Ainsi, seuls les objets de contrôle sont réécrits entre l'ouverture et la fermeture. Cette méthode a l'avantage d'être à la fois simple et rapide à écrire, surtout si on dispose déjà d'un algorithme.

Cependant, puisque ces opérateurs peuvent être définis dualement l'un par rapport à l'autre (l'opérateur d'extraction d'extrema, ainsi que les différents appels à $\overline{\wedge}$ et $\underline{\vee}$), le fait de passer l'ordre en paramètre de l'algorithme semble être également une bonne solution. Cela permet entre autre de définir des ordres de manière dynamique, c'est à dire extérieurement à l'algorithme. Cet avantage est important dans le cadre des ordres lexicographiques: le nombre d'ordres possibles augmente rapidement avec la taille de l'espace $\mathbf{F}$, et il est pratique de pouvoir changer dynamiquement l'ordre pour se rendre compte des résultats rapidement. Dans ce cas précis, l'ordre lexicographique est implémenté à l'aide de deux vecteurs: le premier sert à coder la permutation, c'est à dire l'ordre des indices des espaces $\mathbf{F}_1, \mathbf{F}_2, \ldots \mathbf{F}_n$. Le second vecteur sert à coder l'ordre effectif sur chacun des espaces $\mathbf{F}_i, i \in \mathbb{N}_n^*$. Ces ordres sont au nombre de deux: $\leq$ et $\geq$. Ceci suppose que chacun des espaces $\mathbf{F}_i, i \in \mathbb{N}_n^*$ soient munis d'une relation d'ordre naturelle, $\leq$ ou dualement $\geq$. Dans nos développements, nous avons restreins chaque $\mathbf{F}_i$ au cas scalaire. Enfin le dual d'un ordre n'est autre que ce même ordre, en inversant le vecteur codant les ordres (remplacement de $\leq$ par $\geq$ et *vice versa*).

Nous allons à présent voir sur le simple exemple d'extraction d'extrema locaux la manière d'adapter l'algorithme pour une utilisation selon les ordres $\leq$ et $\geq$, plus précisément selon une relation unique d'ordre stricte et faible $\prec_R$.

4.3.1.3. Extraction des extrema locaux

Les extrema locaux sont un sous-ensemble des zones - strictement - plates de l'image. La condition pour qu'une zone plate soit minimum (maximum) local porte sur le voisinage immédiat de la zone : aucun point voisin de la zone n'est inférieur (supérieur) strictement à la hauteur de la zone. Ainsi, pour la construction de cet ensemble, une base algorithmique serait la labellisation en zone plate - algorithme 2.3 - décrite au chapitre 2 (page 43).

Description de l'algorithme 4.3 d'extraction des extrema locaux Dans cet algorithme « $\mathcal{I}$ » est l'image d'entrée sur laquelle nous cherchons les extrema locaux, et « imageTravail » est une image de travail de même dimension. $\prec_R$ est une relation d'ordre **stricte** et **faible**. Rappelons que pour un tel ordre: $\forall a, \neg (a \prec_R a)$ et

$$\forall (a,b), (\neg(a \prec_R b) \wedge \neg(b \prec_R a) \Rightarrow a \cong_{\prec_R} b)$$

$\cong_{\prec_R}$ étant une relation d'équivalence associée à $\prec_R$. Comme nous l'avons précisé au chapitre 2, les zones plates forment un ensemble quotient dans l'espace image (donné par la relation d'équivalence « *est voisin et de valeur égale* »). La transitivité rend possible le traitement des points qu'une unique fois. En ce sens, l'algorithme est sensiblement identique à son homologue de labellisation. La différence est qu'il faut déterminer la cause de la violation de l'équivalence entre un point et son voisin. Selon cette cause, la zone labélisée perd ou non sa qualité extrémale.

• Chaque point p de l'image originale à l'état « non−traité » est inséré dans la file « q_1 » (ligne 5). Cette insertion sert à initialiser la zone plate contenant p. Par ailleurs q_1 contient les points devant être parcourus à chaque étape de la propagation. La zone est supposée extrémale, supposition pouvant être par la suite invalidée. Puisque nous cherchons un sous-ensemble des classes d'équivalence de l'image [(xix)], l'ordre de traitement des points n'a pas d'importance.

• Chaque point p est ensuite dépilé de « q_1 », inséré dans « q_2 » et marqué définitivement dans « imageTravail », de manière à ne plus être traité. Les voisins de p sont ensuite explorés. Chaque voisin est soit équivalent, soit non-équivalent. La violation de l'équivalence est donnée par la validité de la relation d'ordre stricte, dans un sens ou dans l'autre (ligne 16). Le sens qui nous intéresse particulièrement ici est celui invalidant la propriété extrémale de la zone plate courante. Dans la recherche des minima locaux, si l'image de v est plus petite que celle de p ($\mathcal{I}(v) \prec_R \mathcal{I}(p)$), la zone plate contenant p n'est plus minimum local. La variable « est−extrema » change alors d'état pour devenir « *faux* ». L'ajout de v aux files « q_1 » et « q_2 » est évité.

• Si la relation d'ordre stricte $\prec_R$ n'est vérifiée ni dans un sens ni dans l'autre (ligne 20), cela conduit à l'équivalence entre v et p. Le voisin v appartient à la zone plate de p. v est alors inséré dans la file « q_1 » servant au parcours de la zone plate. Si la zone est toujours extrémale, v est également inséré dans « q_2 ». Cette insertion est inutile dans le cas contraire (ligne 26).

• Dans le cas particulier des ordres lexicographiques, la détermination de $\mathcal{I}(v) \prec_R \mathcal{I}(p)$ ou $\mathcal{I}(p) \prec_R \mathcal{I}(v)$ est toute aussi coûteuse que celle d'une équivalence. En effet, pour chaque espace $\mathbf{F}_i, i \in \mathbb{N}_n^*$, l'appel à une relation $=$ ou $<$ nécessite le même temps. Seule une opération supplémentaire (booléenne) intervient à la ligne 20 ($\neg$ test $\wedge$ ...), alors que cette conjonction est inutile si nous disposions de la

[(xix)] cf. ensemble quotient définition 2.9 page 26

relation d'équivalence. C'est également la seule différence lorsque l'on se restreint aux ordres classiques. Cet algorithme est donc légèrement plus coûteux que celui utilisant conjointement $\prec_R$ et $\cong_R$.

Algorithme 4.3 : Squelette algorithmique pour la détection des extrema locaux

Data : $\mathcal{I}$: image d'entrée
Data : $\prec_R$: relation d'ordre
Result : L: Liste de listes de points extrema

imageTravail $\leftarrow$ non-traité
$L \leftarrow \varnothing$
forall $p \in \mathcal{I}$ **do**
  if *imageTravail*$(p) =$ *traité* **then** continuer
  $q_1 \leftarrow$ initialiser avec p
  $q_2 \leftarrow \varnothing$
  est-extrema $=$ vrai
  ▶ *Parcours de la composante connexe commençant en* p
  while $q_1 \neq \varnothing$ **do**
    $p =$ premier élément de q_1
    $q_1 \leftarrow$ supprimer le premier élément
    $q_2 \leftarrow$ ajouter p
    imageTravail$(p) =$ traité
    Parcours du voisinage de p
    forall $v \in \mathcal{N}_p \setminus p$ **do**
      test $= \mathcal{I}(v) \prec_R \mathcal{I}(p)$
      if *est-extrema* $\wedge$ *test* **then**
        ▶ *p possède un voisin plus petit (plus grand) strictement. La zone n'est plus minimale (maximale)*
        est-extrema $=$ faux
      else if $\neg$ *test* $\wedge \neg (\mathcal{I}(p) \prec_R \mathcal{I}(v))$ **then**
        ▶ *Le voisin v est équivalent à p : v appartient à la même zone plate que p*
        if *imageTravail*$(v) \neq$ *traité* **then**
          imageTravail$(v) =$ traité
          $q_1 \leftarrow$ ajouter v
          ▶ *La zone est toujours extrémale, on ajoute v à q_2*
          if *est-extrema* **then** $q_2 \leftarrow$ ajouter v
  ▶ *Si aucun point de la composante connexe n'a invalidé la propriété d'extremum*
  if *est-extrema* **then**
    $L \leftarrow$ ajouter q_2

4.3.1.4. Résultats de la reconstruction lexicographique

Dans cette section, nous présentons quelques résultats liés aux reconstructions lexicographiques. Notons que ce type de traitement est assez difficile à appréhender.

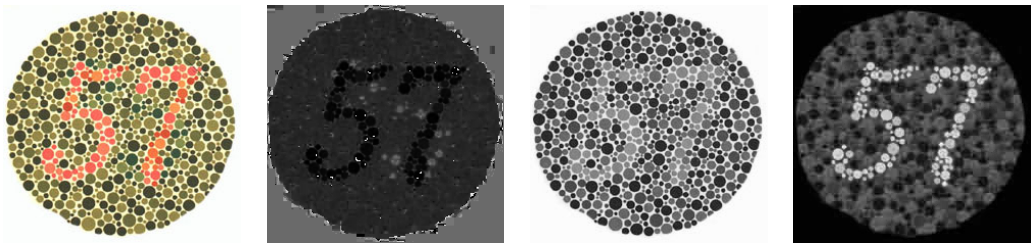
La figure 4.20 présente une application de séparation de lettre sur une image particulière, « Ishihara », destinée au test de daltonisme. L'image est construite de manière particulière: de petit disques de couleurs, non connexes, sont placés les uns près des autres. Pour les non-daltoniens, deux chiffres sont visibles. Nous souhaitons accentuer le chiffre « 7 » et filtrer le chiffre « 5 ». Sur l'image 4.20(c), nous constatons que la luminance est assez homogène sur l'ensemble du disque: une approche sur la luminance seule sera difficile. Par contre, les chiffres apparaissent légèrement plus lumineux que le fond vert. Nous constatons également sur l'image 4.20(d) que la saturation est discriminante.

Une approche marginale sur le canal de saturation pose, comme nous l'avons souligné en introduction de ce chapitre, un problème de recombinaison. Une fois le traitement effectué sur la saturation, la recombinaison avec les deux autres canaux introduira de nouvelles couleurs, ce que nous souhaitons éviter. Une approche algébrique semble adaptée.

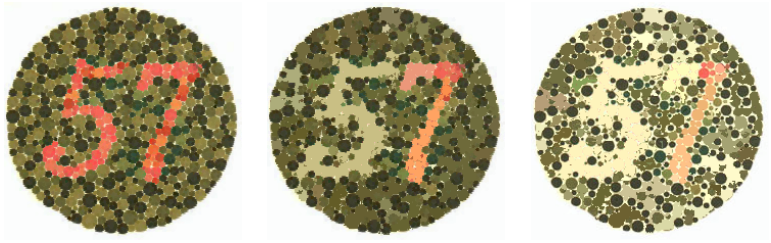
La première étape consiste à combler les trous (le fond blanc) entre les disques de couleur. Pour cela nous effectuons une « dilatation », mais l'ordre n'est pas le même sur tous les canaux. La saturation doit être dilatée au sens classique, alors que la luminance doit être érodée pour supprimer le fond blanc. Nous négligeons le canal de teinte et lui imposons un ordre arbitraire. Ainsi, la dilatation mentionnée est une dilatation sur la saturation, une érosion sur la luminance et une dilatation sur la teinte, pris dans cet ordre. C'est ce que nous notons le treillis $S\downarrow L\uparrow H\downarrow$. L'opérateur de voisinage utilisé est le supremum sur voisinage, comme pour le cas classique, et seule la relation d'ordre change. Le résultat de cette dilatation est donné en figure 4.20(e).

Nous effectuons ensuite la reconstruction lexicographique en prenant cette image pour masque. Nous utilisons cette fois-ci un treillis lexicographique $S\downarrow L\downarrow H\downarrow$ sur lequel chacun des ordres est dirigé vers l'origine (noir pour la luminance, achromatique pour la saturation et rouge pour la teinte). À partir de l'image 4.20(e), nous construisons une image marqueur pour la reconstruction. Cette image est nulle partout (correspond aux origines respectives des canaux) sauf en un point intérieur du chiffre « 7 » de l'image. En ce point, l'image marqueur prend la valeur de l'image originale.

Le résultat est présenté en figure 4.20(f). Premièrement, nous remarquons visuellement que le chiffre « 7 » est bien préservé par rapport au chiffre « 5 ». La teinte de « 7 » est assez proche de celle de l'image (masque de reconstruction) initiale. Le chiffre « 5 » a pris une teinte similaire aux pastilles vertes, et semble moins saturé que la lettre initiale.



(a) Image originale (b) Canal de teinte (c) Canal de luminance (d) Canal de saturation



(e) Dilatation de (a) sur le treillis $S\downarrow L\uparrow H\downarrow$ (f) Reconstruction de (e) sur le treillis $S\downarrow L\downarrow H\downarrow$, marquage du 7. (g) Reconstruction directe de (a) sur le treillis $S\downarrow L\downarrow H\downarrow$, marquage du 7.

Fig. 4.20.: Illustration de la reconstruction lexicographique sur « Ishihara 5 ». Consulter le texte pour les explications.

Un grand nombre de traitements dérive de la reconstruction. Nous allons voir de quelle manière les

(a) Image originale (b) Ouverture (c) Fermeture (ouverture dans le treillis dual)

Fig. 4.21.: Extraits du calcul de la granulométrie.

développements sur les ordres lexicographiques présentés peuvent nourrir d'autres opérateurs morphologiques. Pour cela, nous allons brièvement décrire la granulométrie selon les ordres lexicographiques.

4.3.2. Granulométries lexicographiques

Les granulométries sont des opérateurs utilisés pour caractériser la distribution en taille d'un ensemble d'objets. Bien que fondées sur des développements théoriques complexes (cf. Matheron [Mat69]), les *granulométries* [(xx)] sont explicables relativement simplement.

Les granulométries peuvent être effectuées de plusieurs manières. La plus simple est d'effectuer une série d'ouvertures de taille croissante. Plaçons nous dans le cadre binaire, et soit $i \in \mathbb{R}_+$ une taille. Pour chaque taille d'ouverture, lorsqu'un grain est de taille supérieure, une fonction sur l'ouvert mesure ensuite la réponse de l'ensemble des objets à une taille d'ouverture donnée. Les granulométries sont définissables de la manière suivante:

$$\Theta_i : \begin{cases} \mathbf{F}^{\mathbf{E}} & \rightarrow & \mathbb{R} \\ \mathcal{I} & \mapsto & \int_{\mathbf{E}} \mu\left(\gamma^{(i+1)}(\mathcal{I}), \gamma^{(i)}(\mathcal{I})\right) \end{cases} \tag{4.22}$$

où $\mathbf{F}^{\mathbf{E}}$ est l'ensemble des fonctions de $\mathbf{E}$ à valeur dans $\mathbf{F}$, μ une mesure réelle sur $\mathbf{F}$, et $\mathcal{I}$ une image. Pour une image donnée, chaque Θ_i « teste » le contenu de l'image par rapport l'élément structurant utilisé pour les ouvertures. La fonction

$$f_{g,\mathcal{I}} : \begin{cases} \mathbb{N} & \rightarrow & \mathbb{R} \\ i & \mapsto & \Theta_i(\mathcal{I}) \end{cases}$$

définit alors la fonction granulométrique, ou plus correctement sa fonction dérivée.

Extension aux treillis lexicographiques Nous avons jusqu'ici mentionné les termes *ouverture* et *mesure*. Dans le cadre des treillis lexicographiques, la définition de l'ouverture est immédiate. Pour des espaces images $\mathbf{F}$ vectoriels, nous pouvons reprendre l'approche décrite au §3.3.3 (page 91) concernant les quasi-distances. Nous utilisons alors comme mesure une fonction de distance couleur, dont la description se trouve en §4.2.1 (page 105).

μ produit alors une image à valeurs dans $\mathbb{R}_+$, qu'on intègre ensuite de manière classique.

Nous avons testé plusieurs treillis, en variant l'ordre des dimensions et leur sens (nous avons utilisé encore une fois l'espace ce couleur HLS). Celui qui semble nous donner les meilleurs résultats sur les images de billes colorées semble être le treillis $L \downarrow S \downarrow H \downarrow$. Un extrait du calcul de la granulométrie est donné en figure 4.21.

Le résultat de la granulométrie est donné en figure 4.22. La distance utilisée pour la mesure (sous le signe intégral) est la « distance » luminance-teinte pondérée (équation 4.6 page 108).

Nous remarquons sur cette image un « pic » des distributions vers la valeur 20 pour les ouvertures, et 25 pour les fermetures. Les ouvertures sont significatives de la taille des grains, alors que les fermetures de leur éloignement. La valeur 20 correspond bien à la taille des grains en largeur. Une fois le grain

[(xx)] c'est à dire des *mesures sur des grains*

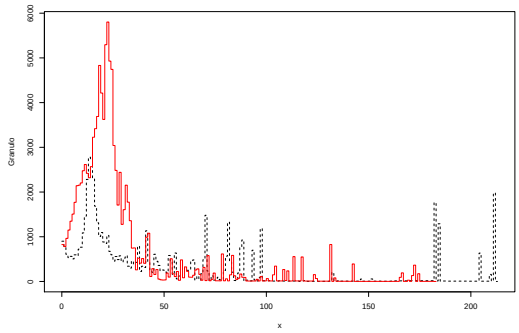


Fig. 4.22.: Fonction dérivée granulométrique de l'image 4.21. Respectivement en noir et rouge les granulométries par des ouvertures et des fermetures sur le treillis $L \downarrow S \downarrow H \downarrow$. L'axe des abscisses représente la taille de l'élément structurant, alors que l'axe des ordonnées représente la valuation de la mesure donnée par l'équation 4.22

(a) Ouverture par reconstruction (b) Fermeture (ouverture dans le treillis dual)

Fig. 4.23.: Extraits du calcul de la granulométrie avec des ouvertures par reconstruction. Il s'agit de la même étape que la figure 4.22 dans le calcul de la granulométrie. (a) Par rapport à l'image 4.21(b), les contours des grains sont mieux préservés. Également, quelques textures internes aux grains (variation de couleur) persistent, alors qu'elles sont complètement perdues par les ouvertures simples.

érodé en largeur, celui-ci perd sa structure géométrique et ne fournit plus de mesure significative à Θ_i, d'où la forme de la distribution de la figure 4.22.

Utilisation de la reconstruction L'ouverture utilisée dans l'équation 4.22 modifie les objets d'étude, et plus particulièrement leurs contours. Il est possible de remplacer dans cette équation l'ouverture simple par une *ouverture par reconstruction*. Nous avons alors:

$$\Theta_i^\infty : \left\{ \begin{array}{ccc} \mathbf{F}^\mathbf{E} & \rightarrow & \mathbb{R} \\ \mathcal{I} & \mapsto & \int_\mathbf{E} \mu \left(\gamma_\mathcal{I}^\infty \circ \epsilon^{(i+1)}(\mathcal{I}), \gamma_\mathcal{I}^\infty \circ \epsilon^{(i)}(\mathcal{I}) \right) \end{array} \right. \tag{4.23}$$

Les granulométries par reconstruction offrent l'avantage de reconstruire (presque) entièrement les objets non détruits par l'érosion, comme l'atteste la figure 4.23.

La fonction granulométrique qui en découle présente généralement des sauts plus francs sur les tailles significatives des objets de l'image. Nous constatons cela sur la figure 4.24, avec un pic important à la valeur 40, ce qui correspond bien à la taille des grains en longueur. Contrairement aux ouvertures, la reconstruction qui succède à l'érosion reconstitue presque en totalité le grain. La valeur significative de 20 (largeur du grain) n'apparaît plus.

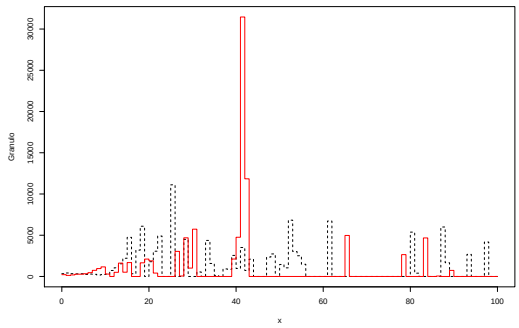


Fig. 4.24.: Fonction dérivée granulométrique de l'image 4.23. Respectivement en noir et rouge les granulométries par des ouvertures et des fermetures par reconstruction sur le treillis $L\downarrow S\downarrow H\downarrow$.

4.3.3. Conclusion sur la lexicographie

L'objectif de cette partie était double. D'une part, nous avons montré la faisabilité et l'adéquation d'une morphologie mathématique multivaluée, à l'aide des outils informatiques développés dans le chapitre 2. D'autre part, nous avons montré l'intérêt que peut susciter des opérateurs morphologiques sur des treillis lexicographiques. Rappelons en quelques points:

- Disposer d'un cadre algébrique, de manière à appliquer directement les opérateurs connus sans adaptation particulière.
- La formulation algorithmique, grâce au cadre algébrique, reste pratiquement inchangée. Notamment, des algorithmes performants développés initialement dans le cadre scalaire, peuvent être utilisés.
- Les développements suivent la même logique que ceux concernant le domaine scalaire. L'introduction de l'opérateur géodésique par excellence, la reconstruction, a permis la définition d'une fonction granulométrique classique.

Ainsi, nous pourrions voir prochainement d'autres opérateurs inhérents à la géodésie, comme le filtrage par dynamique, les nivellements, etc.. Enfin, la perspective peut-être la plus intéressante est de « dompter » le comportement des opérateurs lexicographiques avec la même intuition que dans le domaine scalaire.

4.4 Invariants chromatiques robustes de peau

Dans cette partie, nous allons appliquer les connaissances acquises pour extraire des invariants concernant la détection de peau. Nous nous plaçons dans le contexte du milieu automobile, contexte dans lequel l'illumination est soumises à de très fortes fluctuations. Nous justifions la présence de cette étude, qui n'est pas d'ordre morphologique, de la manière suivante: dans le cadre du traitement d'image et plus particulièrement de la segmentation, on est souvent amené à placer des marqueurs dans l'image. L'action de placer des marqueurs signifie qu'en certaines zones de l'image, l'ambiguïté est absente. L'objet de cette étude se place dans le cadre suivant : l'extraction de marqueurs robustes de peau, peu sensible à des grandes fluctuations de l'environnement et non dépendante d'un matériel d'acquisition.

À cause des contraintes fortes présentées en introduction, l'enjeu principal de l'étude sera de valider nos résultats dans des conditions proches de l'environnement de travail. Nous proposerons pour cela des résultats dans de nombreux espaces colorimétriques, simulant la variation des primaires RGB et des blancs de référence. Concernant les espaces circulaires HLS, à notre connaissance aucune étude sérieuse n'a pris en compte l'impact de la variation d'illuminant. Nous proposerons également un modèle de peau et une méthode de changement d'illuminant, afin de mesurer précisément l'influence du changement d'illuminant. Pour ces développements, nous nous servirons des développements précédents de statistiques circulaires.

La section s'articule de la manière suivante. Après une rapide introduction des méthodes existantes, de nos bases et de notre protocole de tests, nous effectuerons les projection des zones de peau dans les plans chromatiques dans de nombreux espaces. La variance de l'environnement sera simulée grâce aux équations de changement d'espace. Nous simulerons également la variance d'acquisition à l'aide de l'utilisation de nombreuses bases de données. Le choix se portera sur une représentation en coordonnées cylindriques pour la stabilité de la représentation chromatique qu'elle offre.

Ensuite, la discussion sera naturellement consacrée aux propriétés chromatiques de peau et à la recherche d'invariant dans l'espace cylindrique. Plusieurs points seront abordés. Nous proposerons un modèle statistique circulaire de peau, dont nous devrons estimer les paramètres. Les espaces circulaires ne permettent pas, par construction, de prendre en compte la variabilité de l'illuminant: cette dernière sera simulée grâce à une méthode colorimétrique. Nous analyserons de manière quantitative la variation des paramètres du modèle sous les contraintes de changement d'illuminant. Pour terminer l'étude, nous proposerons une méthode simple de marquage de peau.

4.4.1. Traitement de la peau

4.4.1.1. État de l'art

Un grand nombre de travaux utilise comme premier élément de segmentation les propriétés colorimétriques de la peau. Une image couleur porte plus d'information qu'une image noir et blanc (en teinte de gris ou en luminance). La représentation couleur est, de tous les points de vue, plus riche. Le problème de détection de peau est un sujet relativement récurrent dans la littérature, dans la mesure où un grand nombre d'applications se concentrent sur la détection des visages, l'identification et la reconnaissance [YKA02, MSL01, LKP96]. C'est la raison pour laquelle la littérature propose un contenu assez riche sur le sujet, et met un accent particulier sur le fait que certaines propriétés chromatiques de la peau sont des invariants intra et inter ethnique [(xxi)].

Les premiers éléments de détection de zones de peau basés sur la couleur sont assez simplistes, mais permettent de cerner toute la facilité avec laquelle cette tâche peut être achevée dans des conditions appropriées. Dans [SP96], un simple seuillage sur la dimension de teinte, dans l'espace couleur HLS,

[(xxi)] une référence à de telles études est donnée dans *Wysecki & Stiles* [WS82], mais nous n'en avons pas trouvé la source exacte

permet de séparer efficacement le visage du fond ou des zones de non intérêt. Les conditions sous lesquelles sont employées le traitement sont malheureusement *trop* simplistes pour nos conditions puisque la détection est grandement facilitée par un fond de teinte bleue. Nous prouverons cette proposition conjecturale par des tests quantitatifs. L'intérêt de cet article réside principalement dans la mise en évidence d'un espace couleur approprié à la résolution du problème, espace hybride que nous avons qualifié d'*interface* .

Les auteurs de [MJ02] tendent à proposer un modèle d'étude plus complet du locus de la peau, à partir de l'établissement de statistique sur une large base de test. La procédure est la suivante: deux histogrammes tri-dimensionnels représentant le cube RGB sont construits: un pour la peau et un autre pour la non-peau. Le système classifie ensuite un point comme étant de la peau si, à l'aide des histogrammes et en ce point, le quotient entre le nombre d'occurrences de peau et de non-peau est supérieur à un certain seuil. Les auteurs comparent cette méthode de classification aux performances d'une mixture de gaussiennes, et bien que très simple, la méthode par histogramme donne des résultats très comparables. Un point faible de cet article est l'absence d'investigation d'autres espaces couleurs, qui auraient certainement des propriétés de séparabilité entre les classes peau et non-peau bien meilleures. Par ailleurs il est dit dans cet article que beaucoup de points de la classe peau se retrouvent très proche de l'axe achromatique, ce qui lance un voile de doute sur la capacité de la méthode d'extraire de l'information chromatique. Enfin dernier point et non des moindres, l'utilisation d'un paramètre nécessite l'intervention humaine.

Plus récemment Störring [SAG99, Stö04] se base sur un modèle de réflexion dichromatique. Un modèle spectral de réflectance de la peau est combiné à des modèles spectraux de réception (acquisition du capteur de la caméra) et d'illuminant (lampes spéciales). Les lieux chromatiques de peau en sont déduits par intégration spectrale de ces trois modèles. Ceci permet à l'auteur d'établir les loci de peau pour des températures d'illuminant différentes. Le modèle est suffisamment précis pour permettre ensuite de déduire de l'observation la température de l'illuminant. Cependant un grand nombre de paramètres (réponse spectrale des caméras, des lampes, etc.) entrent dans la modélisation, ce qui rend l'exploitation en situation réelle impossible.

Direction dans le choix de l'espace de représentation Il existe un grand nombre d'espaces de représentation de la couleur [(xxii)]. Compte tenu de leur nature très différentes (colorimétrique, perceptuelle, subjective, pratique...), l'espace couleur de travail pose un problème de choix délicat. Il n'y a de solution plus adaptée qu'une autre que dans le sens où:

- un espace permet une modélisation plus facile ou intuitive des données. Plus précisément, les données projetées forment des ensembles compacts et dont les limites sont facilement modélisables.
- des outils adéquats et puissants sont disponibles dans cet espace, ce qui tend à exclure tous les espaces autres que métrique. Nous verrons que cette condition est bien trop dure et en profiterons pour adapter des outils existants aux espaces qui en justifient l'intérêt.

Nous allons dans la suite présenter le contenu de la base de donnée. Nous montrerons ensuite la projection de la classe de peau dans les espaces de représentation, et en déduirons le choix de l'espace.

4.4.1.2. Contenu des bases de données

Nous avons utilisé plusieurs bases de données pour disposer d'un large échantillon des conditions d'acquisition, du matériel et de la population étudiée. Malheureusement certaines de ces bases sont confidentielles et nous ne pouvons en divulguer les images. Nous nous en servirons pour donner un support supplémentaire à nos résultats. Il est néanmoins possible d'en décrire le contenu:

- La base de données 1 est une base très hétérogène présentant une « gamme » d'individus aussi large que possible. De nature très diverse, les visages sont ceux d'individus adultes, jeunes, ou enfants, et de pigmentation type européen, asiatique, caucasien, indien et africain. Les scènes sont

[(xxii)] cf. annexe §A.2 à ce sujet

également variées, les fonds très différents. Cette base sera considérée comme la base de données *Web*. Nous travaillerons sur deux sous-ensemble de la base, Web_1 et Web_2, respectivement de 52 et 13 éléments.

– La base 2 est également une base dont les individus sont très hétérogènes. Elle représente des individus adultes, et les conditions d'acquisition sont maîtrisées. Les conditions d'éclairage peuvent être considérées comme constantes, la proportion du visage par rapport à la taille de l'image également. Cette base sera référencée *Base T*. Nous travaillerons sur un extrait de 31 éléments.

– La base 3 est issue d'une étude de biométrie menée par l'Inria [GHC04]. Elle représente des individus variés, mais de « groupe ethnique » à peu près similaire, sous toutes les poses. Les images sont compressées aux format JPEG, et donc les informations couleurs sont très dégradées, comme le montre la figure 4.25. La qualité d'acquisition est d'une manière générale mauvaise, mais ressemblera probablement à certaine de nos conditions de travail. Cette base sera référencée *Inria*. Nous travaillerons sur un extrait de 24 éléments.

– La base 4 contient nos propres acquisitions dans une voiture à l'arrêt. Les individus sont assez homogènes dans le type, mais comme nous le verrons présentent des « couleurs » de peau assez variées. Les conditions d'éclairage sont du type *ciel couvert*. La caméra utilisée est une caméra CCD analogique du commerce, lié au PC par une carte d'acquisition. Cette base sera référencée *Hitachi*. Nous travaillerons sur un extrait de 20 éléments.

– Enfin la base 5 contient nos acquisitions dans une voiture en mouvement. Les individus sont à peu près les mêmes, mais les conditions d'éclairage sont variables et de type ensoleillé. La caméra utilisée est une caméra CCD de type *Webcam* lié au PC par une connexion USB, ce qui indique une éventuelle compression du signal. Cette base sera référencée *Webcam*. Nous travaillerons sur un extrait de 19 éléments.

(a) Image originale (b) Teinte (c) Luminance (d) Saturation

Fig. 4.25.: Exemple issu de la base Inria: bien que l'image de luminance (c) corresponde à notre perception de l'image originale (a), les informations chromatiques de teinte et de saturation présentent un contenu fortement dégradé, principalement dû à la compression JPEG utilisée. Les traitements utilisant la couleur sont loin d'y être insensibles.

4.4.1.3. Projection des lieux chromatiques de peau

Après avoir sélectionné un certain nombre d'image représentatives dans chacune des bases de test, nous avons créé, à la main, le masque des zones de peau correspondant. Les masques étant fait à la main, leur précision est « humaine », si bien qu'il n'est pas rare d'y inclure des éléments qui ne sont pas vraiment de la peau comme les yeux, la bouche, la barbe ou moustache, quelques mèches de cheveux... Toutefois, en termes de nombre de pixel, la peau est toujours dominante pour chacune des images.

Remarquons que la littérature traite parfois les pixels de classe *non-peau*, de manière à trouver un espace dans lequel les classes *peau* et *non peau* présentent la meilleure séparation. Si cette démarche

est justifiée dans une étude classique de classification, ce n'est pas le cas dans le problème que nous abordons ici. En effet, les pixels de classe *non-peau* sont potentiellement partout sur le gamut [(xxiii)] de l'espace considéré, et il est impossible d'énoncer des hypothèses particulières pour cette classe. Il est en effet évident que, dans notre base de test, les pixels *non-peau* sont un sous-ensemble difficilement représentatif de la totalité de l'ensemble *non-peau* possible. Par ailleurs, la classe *peau* est également inclue dans la classe *non-peau*. Ainsi, la démarche que nous proposons cherche à qualifier la représentation de la classe *peau* dans l'ensemble du gamut considéré.

C'est la raison pour laquelle nous avons apporté un soin particulier pour la détermination des limites des plans chromatiques des espaces couleur. Pour cela, nous avons placé des points sur une grille régulière dans le cube RGB de départ, nous avons appliqué ensuite à ce dernier les transformations identiques à celle des zones de peau. Nous avons ensuite extrait l'enveloppe convexe de la composante chromatique de la transformée du cube.

Il nous a paru également important de projeter le point blanc; en effet, la position de ce point indique clairement une orientation éventuelle des données selon une teinte dominante. De plus, un confinement des pixels de peau autour du point blanc est significatif de points dont la composante chromatique est pauvre.

Protocole : Pour les espaces colorimétriques, nous avons émis trois hypothèses sur l'espace RGB de départ: sRGB, CIERGB et enfin AdobeRGB. sRGB est défini de manière précise [(xxiv)], mais pour les autres primaires, il est tout à fait possible d'utiliser d'autres illuminants que ceux venant avec les définitions d'espace [(xxv)]. Ceci nous a permis de tester les projections avec les illuminants standards A, B, C, E, D_{50}, D_{55}, D_{65}, D_{75} pour les primaires CIERGB et AdobeRGB. De cette manière, il nous est possible de voir si les loci de peau présentent des formes comparables sous des hypothèses très différentes - de primaires RGB et d'illuminants.

Description des résultats: Pour chaque résultat et comme mentionné, le cube RGB se projette à l'intérieur de la ligne bleue, le point blanc est signalé par un cercle vert et enfin les pixels de peau sont représentés par des carrés dans les tons de rouge. Les zones les plus denses apparaissent blanches, et les moins denses rouge vif [(xxvi)].

Analyse: Nous allons analyser les résultats dans les espaces colorimétriques puis dans les espaces HLS. Les résultats sur les espaces colorimétriques sont nombreux et pour pouvoir les évaluer correctement, nous allons d'abord comparer les nuages de points en fonction des primaires RGB, puis par illuminants, pour une base fixée. Nous analyserons enfin les similitudes entre les bases.

• La figure 4.26 montre trois loci de la *base T* construit à partir de trois primaires RGB différentes: sRGB, Adobe et CIE. L'illuminant est le même pour les trois primaires. Le gamut, c'est à dire le lieu des points du cube RGB, est plus étroit pour sRGB que Adobe, qui lui même est plus étroit que CIE. Ceci modifie sensiblement la position du point blanc, assez différente malgré le fait que l'illuminant soit le même (et ce même entre sRGB et Adobe, qui tous deux sont définis initialement sur D_{65}). La forme des nuages est également assez différente. Il semble que les trois loci s'étendent symétriquement par rapport au point blanc, mais la densité semble fortement asymétrique orientée vers les rouges. sRGB présente encore un nuage qui semble prendre une orientation autour de la droite passant par le point blanc et faisant un angle de $30°$ avec l'axe x. Ce n'est plus le cas pour Adobe et CIE, où l'on pourrait presque distinguer deux modes: le premier à gauche du point blanc, étroit et dirigé selon $60°$ par rapport

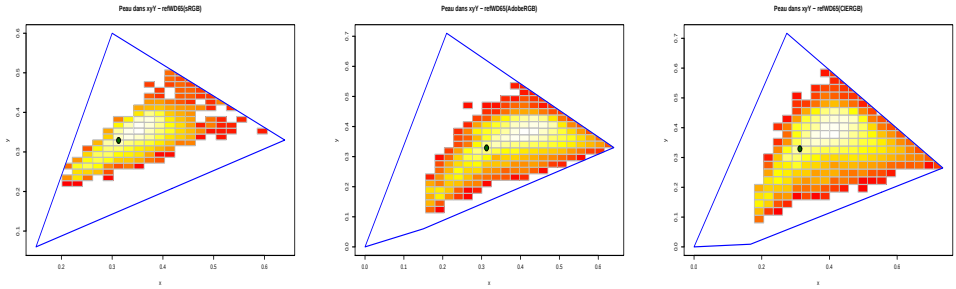
(xxiii) lieu des pixels dans l'espace couleur

(xxiv) la transformation de sRGB vers XYZ n'est pas la même transformation que celle des autres espaces RGB

(xxv) voir la partie §A.2 et la table A.3.2.1 pour ces définitions

(xxvi) Pour établir ces schémas, pour chacune des bases, nous avons créé un cube de 256pt. équi-distribués de coté, représentant le cube RGB. Nous avons ensuite compté les pixels de peau à l'intérieur de ce cube.

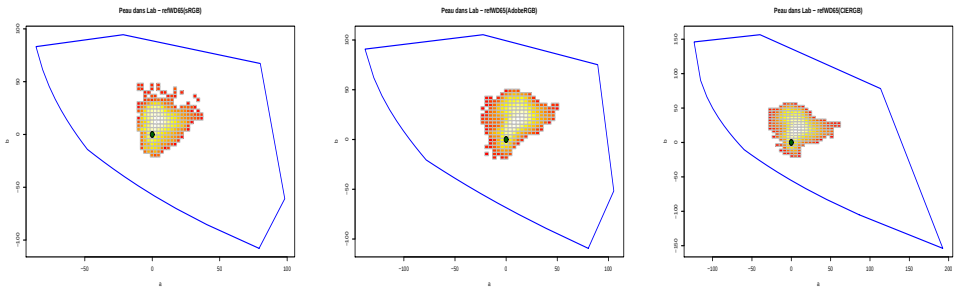
à l'axe x couvrant une partie des pourpres, le second à droite du point blanc, large et couvrant des jaunes aux rouges. L'analyse des autres bases infirme le mode à gauche - nous supposons que cela est dû à l'acquisition privilégiant les bleus - , mais confirme à la fois l'orientation plus étroite de sRGB par rapport aux deux autres, et l'orientation des trois nuages par rapport aux rouges. Un autre point commun semble être la position du point blanc, excentrée à gauche par rapport aux zones de densité importantes. Enfin, la dernière remarque concerne l'espace sRGB qui semble avoir une répartition plus « vaste » des données, ce qui semble être dû au facteur γ [(xxvii)] de valeur différente pour cet espace. Résumons donc ainsi: l'espace XYZ (ou xyY) montre des différences importantes au sein d'une même base, pour un même illuminant et pour des primaires RGB différentes.



(a) Locus de peau dans xyY, primaires sRGB - Base T (b) Locus de peau dans xyY, primaires Adobe - Base T (c) Locus de peau dans xyY, primaires CIE - Base T

Fig. 4.26.: Locus de peau dans xyY à partir de 3 primaires RGB différentes - *base T*. L'illuminant (D_{65}) est le même pour les trois primaires.

La figure 4.27 montre ces mêmes résultats mais dans l'espace Lab. Les nuages présentent ici des formes comparables, à la fois en orientation et en largeur de locus. Ce point se confirme sur les autres bases.



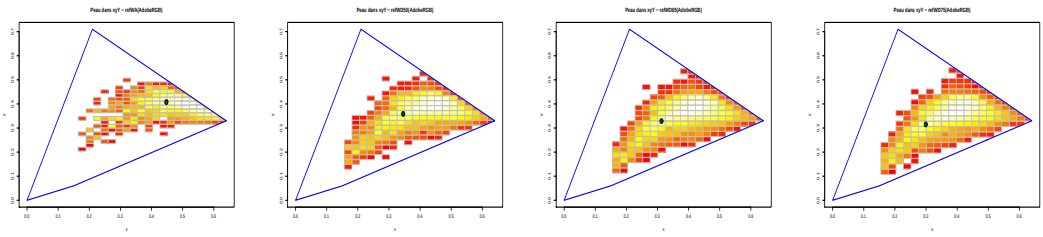
(a) Locus de peau dans Lab, primaires sRGB - Base T (b) Locus de peau dans Lab, primaires Adobe - Base T (c) Locus de peau dans Lab, primaires CIE - Base T

Fig. 4.27.: Locus de peau dans Lab à partir de 3 primaires RGB différentes - *base T*. L'illuminant (D_{65}) est le même pour les trois primaires.

• Comparons à présent les nuages par illuminant, et prenons au hasard les projections à partir des primaires Adobe. La figure 4.28 montre la variation du nuage en fonction de l'illuminant utilisé dans xyY. Il semblerait que le nuage se déplace de la même manière que le point blanc, ce qui conforte les observations précédentes de la position relative du point blanc par rapport au nuage. L'illuminant A confine les point naturellement vers le rouge, et plus l'on augmente en température [(xxviii)], plus l'on se

[(xxvii)] également appelé « *tone curve* »
[(xxviii)] rappelons que D_{50} signifie illuminant standard à 5000 K.

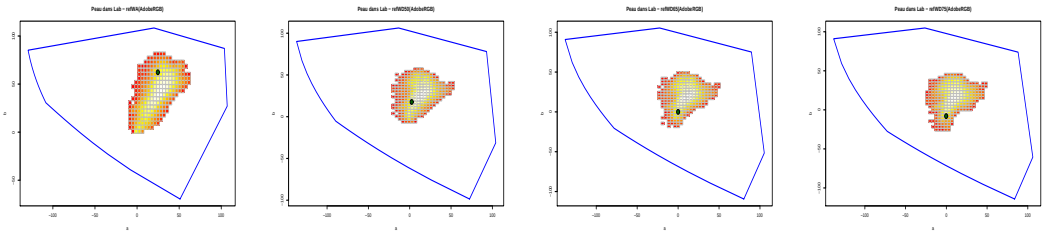
rapproche du centre du triangle. Nous avons observé ce même déplacement relatif sur les autres bases et également sur d'autres primaires.



(a) Locus de peau dans xyY, illuminant A - Base T (b) Locus de peau dans xyY, illuminant D_{50} - Base T (c) Locus de peau dans xyY, illuminant D_{65}, - Base T (d) Locus de peau dans xyY, illuminant D_{75} - Base T

Fig. 4.28.: Locus de peau dans xyY sur différents illuminants (primaires Adobe) - *base T*

Le comportement des nuages dans Lab en fonction de l'illuminants (figure 4.29) semble rejoindre les remarques émises précédemment, à savoir que les formes des nuages varient peu d'un illuminant à l'autre, sauf pour l'illuminant A qui produit une forte distorsion dans la forme du nuage. Cela se confirme sur les autres bases.



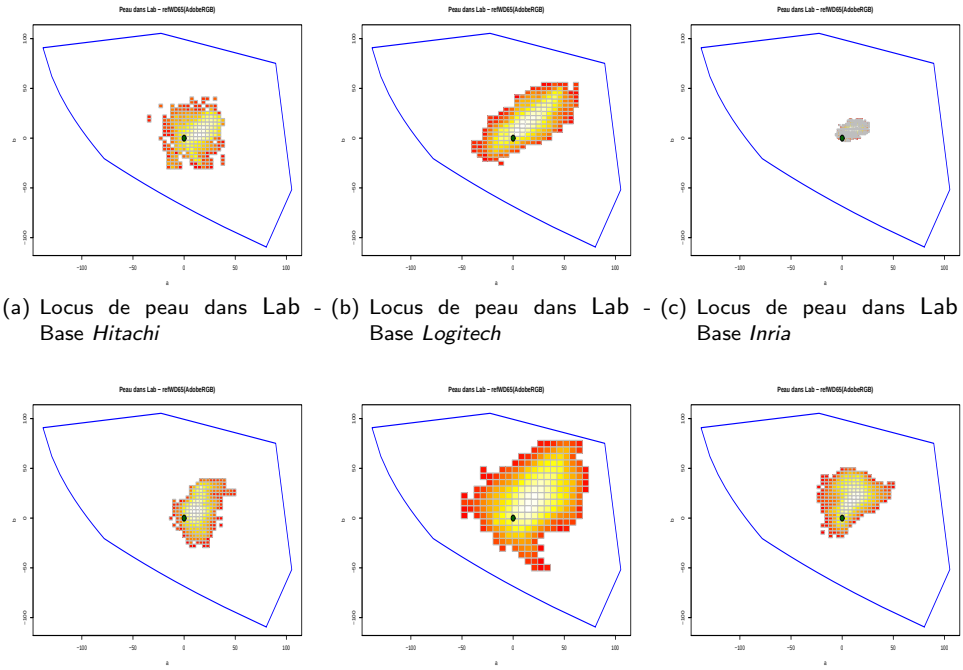
(a) Locus de peau dans Lab, illuminant A - Base T (b) Locus de peau dans Lab, illuminant D_{50} - Base T (c) Locus de peau dans Lab, illuminant D_{65} - Base T (d) Locus de peau dans Lab, illuminant D_{75} - Base T

Fig. 4.29.: Locus de peau dans Lab sur différents illuminants (primaires Adobe) - *base T*

• Enfin, il reste à comparer les nuages en fonction des bases, afin de voir s'il n'existerait pas une structure similaire. Compte tenu de ce qui précède, l'espace xyY / XYZ présentent une grande variabilité intra-base, alors que Lab semblerait être plus stable. Nous n'examinons donc que les projections dans Lab puisqu'une représentation dans XYZ risque fort d'être une cause perdue. La figure 4.30 présente les projections des bases sur l'espace Lab, avec l'illuminant D_{65} et les primaires RGBAdobe.

D'après la figure, il existe une importante variabilité de la forme du nuage (et des contours) entre les différentes bases. La plus étendue est la base *Web1* 4.30(e), étendue que l'on retrouve également sur tous les autres espaces de projection de cette base. Nous expliquons ce phénomène par la grande généralité des images de la base, ainsi que par leur qualité variable [(xxix)]. Cette étendue est comparable à celle de la base *Logitech* qui, rappelons-le, présente des images avec des conditions d'illumination très différentes; ce qui expliquerait en partie ces résultats. À l'opposé, le nuage est extrêmement confiné pour la base *Inria*, ce que nous expliquons par la faible variabilité des individus représentés par rapport aux autres bases. Enfin les trois autres bases présentent des nuages différents, mais de dispersion comparable autour du point blanc.

[(xxix)] certaines images sont mêmes marquée avec une mention textuelle de copyright indiquant la source, clairement visible.



(a) Locus de peau dans Lab - Base *Hitachi* (b) Locus de peau dans Lab - Base *Logitech* (c) Locus de peau dans Lab - Base *Inria*

(d) Locus de peau dans Lab - Base *Web2* (e) Locus de peau dans Lab - Base *Web1* (f) Locus de peau dans Lab - Base *T*

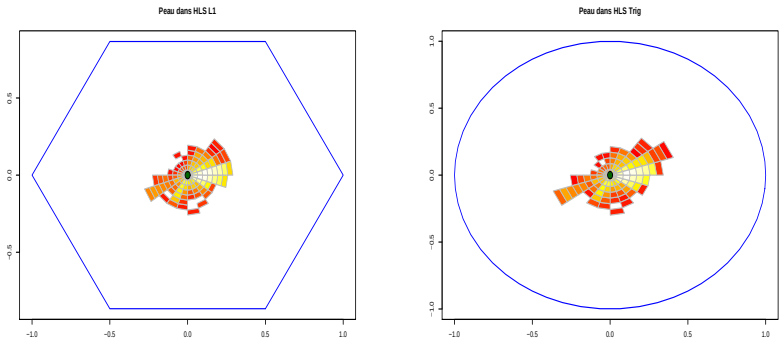
Fig. 4.30.: Locus de peau dans Lab (D_{65} et Adobe) de l'ensemble des bases.

Au vu de ces résultats, Lab semble être un choix intéressant pour la modélisation de la peau, bien que l'établissement d'un modèle y soit difficile et que l'étendue des nuages de points en fonction de la base d'étude soit variable. Ce que nous pouvons remarquer au vu de ces résultats est l'orientation assez similaire des nuages, ce qui laisse présager des résultats plus facilement exploitable dans des espaces couleurs type orientation. Nous pouvons ajouter que la proximité du point blanc par rapport au nuage indique l'achromatisme, ou la pauvreté chromatique, de ce dernier.

• La figure 4.31 montre le résultat de la projection de la base *Hitachi* dans les espaces HLSI1 et GHLS. L'espace GHLS semble étirer diamétralement les nuages de points, ce qui est explicable par une définition très différente de la saturation et par la forme de la projection du cube RGB, plus étroit dans HLSI1. Mise à part cet étirement, très peu de différences existent sur la compréhension des nuages entre ces deux espaces, effet que nous vérifions sur l'ensemble des bases de données. Par ailleurs, certains tests précédents (cf. §A.2) nous ont permis de constater très peu de différences numériques entre les deux définitions de teintes. Nous ne considérerons pour la suite de l'analyse que l'espace GHLS.

• Comparons les nuages en fonction des bases dans GHLS: les histogrammes de chacune des bases sont donnés en figures 4.32. La première remarque concerne la relative faiblesse de saturation des loci de peau, qui reste au mieux confiné à à la moitié inférieure de son échelle. Mis à part la base *Inria* dont la couleur semble extrêmement dégradée, cela indique que la saturation n'est pas très élevée, voire inexistante dans certains cas (les bases montrent de bonnes concentrations autour de l'axe achromatique). L'orientation principale des bases concerne les teintes de rouge (normal), mais il semblerait qu'il existe également des modes vers les bleus (toutes les bases sauf *Inria*); nous avons déjà observé ce phénomène sur les autres espaces couleurs, et au vu de la pauvreté de la saturation, il semble correct que la teinte présente ce genre d'imprécisions.

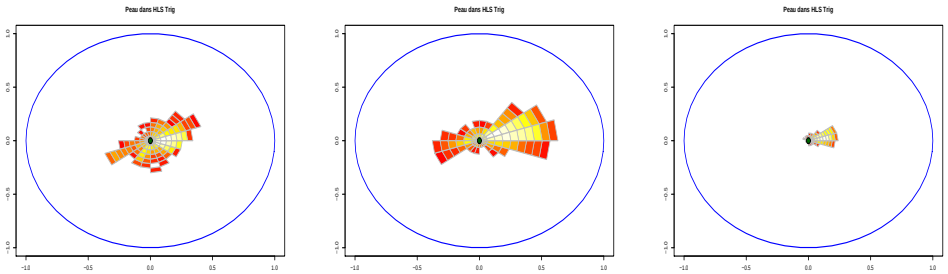
La figure 4.33 présente les histogrammes marginaux de la teinte selon un diagramme en rose. Ces



(a) Locus de peau dans HLS l1 - Base Hitachi

(b) Locus de peau dans GHLS - Base Hitachi

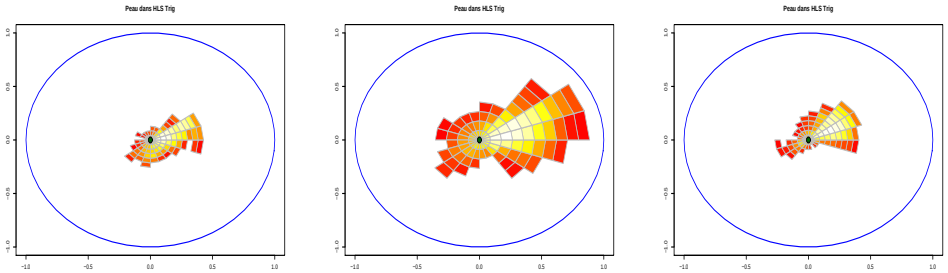
Fig. 4.31.: Locus de peau sur les espace HLSl1 et GHLS - base Hitachi. Les formes des nuages de points indiquent les comportement très similaires des deux espaces.



(a) Locus de peau dans GHLS - Base *Hitachi*

(b) Locus de peau dans GHLS - Base *Logitech*

(c) Locus de peau dans GHLS - Base *Inria*



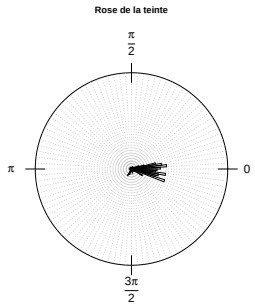
(d) Locus de peau dans GHLS - Base *Web2*

(e) Locus de peau dans GHLS - Base *Web1*

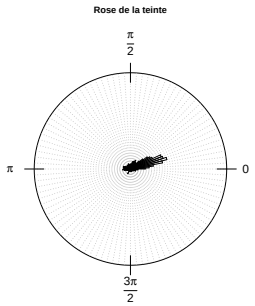
(f) Locus de peau dans GHLS - Base *T*

Fig. 4.32.: Locus de peau dans GHLS de l'ensemble des bases.

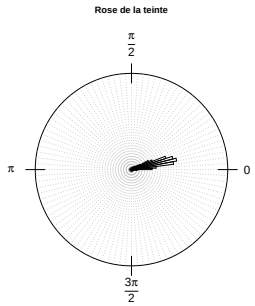
diagrammes montrent une direction nette de la teinte vers les angles compris dans $[-5°, 35°]$. Une certaine dispersion uniforme est également observable, en partie due à la pauvreté de la saturation.



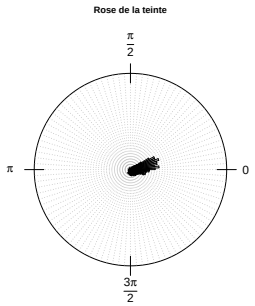
(a) Locus de peau dans GHLS - Base *Hitachi*



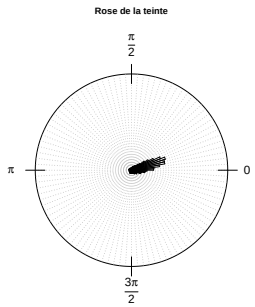
(b) Locus de peau dans GHLS - Base *Logitech*



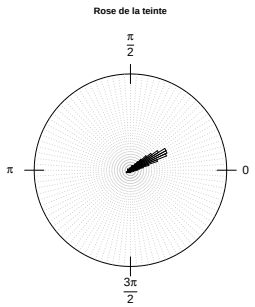
(c) Locus de peau dans GHLS - Base *Inria*



(d) Locus de peau dans GHLS - Base *Web2*



(e) Locus de peau dans GHLS - Base *Web1*



(f) Locus de peau dans GHLS - Base *T*

Fig. 4.33.: Rose des loci de peau dans GHLS sur l'ensemble des bases.

Conclusion

En conclusion, la composante de teinte des espaces HLS considérés semble être un excellent candidat pour la modélisation de la peau, et ce en vertu des points suivants:

- Les données semblent être très compactes, ce qui induit une facilité indéniable pour l'établissement d'un modèle.
- Bien qu'en termes d'orientation principale les bases de données présentent des teintes dominantes

différentes, leur compacité nous amène à penser qu'il y a là un point commun fort, ceci au delà de la mise à défaut des hypothèses sur l'illuminant.
– il n'y a que très peu de différences (concernant la teinte uniquement) entre les résultats des espaces HLSI1 et GHLS. Le calcul de la composante de teinte peut donc être faite sans avoir recours à des fonctions trigonométrique, sans pour autant sacrifier la qualité de la représentation.

Il faut toutefois nuancer ces propos. L'espace HLS n'est assujetti ni à la notion de blanc de référence ni à celle d'illuminant. Or, pour que soit valide la représentation de la peau dans le contexte que nous avons présenté en introduction, il serait pertinent de valider le comportement de la base en simulant le changement d'illuminant. C'est précisément l'objectif de ce qui va suivre: la section §4.4.2 introduit un modèle de distribution circulaire dépendant de deux paramètres. Ce modèle nous permettra ensuite de quantifier le changement d'illuminant à travers l'impact sur les paramètres du modèle. Conjointement, une méthode de changement d'illuminant nous permettra de simuler différentes conditions initiales d'illumination. Enfin, nous analyserons et commenterons les résultats.

4.4.2. Influence de l'illuminant sur la représentation cylindrique

Dans cette section, nous nous intéressons à l'influence de l'illuminant sur la représentation chromatique en coordonnées cylindriques. Nous avons vu dans la partie précédente que l'impact de l'illuminant n'était pas négligeable dans le cadre de notre problème, c'est à dire sur les propriétés chromatiques de la peau.

La première difficulté à laquelle nous devons faire face est l'établissement d'un protocole nous permettant de quantifier correctement cette influence. En effet, pour les espaces colorimétriques, cette difficulté n'existe pas dans la pratique puisque la température de l'illuminant est incluse dans les équations de transformation d'espace (cf. annexe A.3.2 page 256).

Pour quantifier cette influence, nous allons proposer un modèle de peau selon une distribution statistique circulaire très répandue: le modèle de *von Mises*. Nous verrons que ce modèle dépend de deux paramètres - estimateurs - directement calculables à partir du jeu de données. La variabilité des estimateurs sera ensuite utilisée pour la quantification de l'influence du changement d'illuminant.

4.4.2.1. Distribution de von Mises

La distribution de probabilité de *von Mises* pour les espaces circulaires est - *presque* - l'analogue de la distribution normale sur les domaines linéaires. Une variable aléatoire θ présente une distribution de *von Mises* si sa densité de probabilité est donnée par :

$$f(\theta, \mu_0, \kappa) = \frac{1}{2\pi I_0(\kappa)} \cdot e^{\kappa \cdot \cos(\theta - \mu_0)} \tag{4.24}$$

avec I_k la fonction de Bessel modifiée d'ordre k donnée par

$$I_0(\kappa) = \sum_{r=0}^{+\infty} \frac{1}{r!^2} \left(\frac{\kappa}{2}\right)^{2r} \ , \ I_p(\kappa) = \frac{1}{\pi} \int_0^{\pi} \cos p \cdot \theta e^{\kappa \cdot \cos\theta} d\theta$$

Le paramètre μ_0 est considéré comme étant la direction moyenne de cette distribution, et le paramètre κ comme le facteur de concentration autour de μ_0. Si l'on reprend l'analogie avec la distribution normale, μ_0 et κ jouent respectivement les rôles de moyenne et d'inverse de la variance. Pour $\kappa \to 0$, cette distribution tend vers un distribution uniforme sur le cercle unité; à l'inverse, pour $\kappa \to \infty$, la distribution se concentre en un point sur le cercle unité d'angle μ_0.

Il y a donc deux paramètres à estimer: $\hat{\mu_0}$ et $\hat{\kappa}$, le $\hat{}$ indiquant qu'il s'agit d'un estimateur. D'après les développements de [Mar72, page 122] et pour une telle distribution, le meilleur estimateur $\hat{\mu_0}$ de l'angle moyen au sens du maximum de vraisemblance est donné par:

$$\hat{\mu_0} = \bar{\mu_0} \tag{4.25}$$

avec $\bar{\mu_0}$ selon l'équation 4.9 (page 119). Soit $A(x) = \frac{I_1(x)}{I_0(x)}$; le meilleur estimateur $\hat{\kappa}$ de κ au sens du maximum de vraisemblance, est solution de:

$$A(\hat{\kappa}) = \bar{R} \tag{4.26}$$

Bien qu'il n'y ait pas de solution analytique pour l'inverse de l'équation 4.26 précédente, il est possible d'approcher $\hat{\kappa}$ par des fonctions de $\bar{R}$ uniquement. Nous voyons à partir de ce modèle que les seules estimations de $\bar{R}$ et $\bar{\mu_0}$ sont suffisantes pour pouvoir appliquer ce modèle. L'analyse ces deux paramètres sur l'ensemble des bases de test s'inscrit de manière logique dans la suite. Il serait surtout intéressant de tester la réaction de ces deux estimateurs dans des conditions dégradées. Nous l'avons vu pour les espaces colorimétrique en §4.4.1.3, le changement d'illuminant est une dégradation majeure.

Dans les espaces colorimétriques, le changement d'illuminant se fait de manière presque naturelle, en changeant la référence de blanc lors de la création de la matrice de passage de $\mathcal{M}_{RGB \to XYZ}$ [(xxx)]. Ce n'est plus le cas dans l'espace HLS; il est cependant possible de simuler ce changement d'illuminant par une méthode colorimétrique et c'est ce que nous allons voir à présent. Cette approche ressemble à une méthode de balance de blancs, mais nous souhaitons souligner l'originalité de l'approche: l'utilisation de telles méthodes dans la détermination des invariants de peau (ou plus généralement couleur) n'est pas, à notre connaissance, utilisée dans la littérature.

4.4.2.2. Changement d'illuminant synthétique

Nous l'avons déjà mentionné un certain nombre de fois, il n'existe pas, dans l'espace de représentation circulaire, de notion de blanc de référence. Or l'un des problèmes majeur de notre environnement est la variabilité extrême de l'illumination. L'objectif de cette partie est précisément de simuler un changement d'illuminant.
Ce changement ne peut se faire au niveau de la transformation RGB vers HLS. Par contre, il existe une méthode fondée sur une approche colorimétrique transformant le contenu de l'image d'un illuminant vers un autre. Nous pouvons ainsi intégrer de manière indirecte la variabilité de l'illuminant dans nos mesures, et observer ces mesures lors d'un coucher de soleil ou d'une illumination par une lampe à tungstène, et ce même si *a priori* nous ne disposons pas des images sous de telles prises de vue.

À notre connaissance, il n'existe pas de travaux modélisant ces impacts pour l'espace couleur choisi.

Adaptation chromatique: Soient deux illuminants $W_1 = (X_{w_1}, Y_{w_1}, Z_{w_1})$ et $W_2 = (X_{w_2}, Y_{w_2}, Z_{w_2})$ ayant pour coordonnées chromatiques dans l'espace XYZ. Nous notons XYZ_{W_1} et XYZ_{W_2} les espaces correspondants. L'adaptation chromatique est une transformation linéaire permettant de changer les coordonnées de l'illuminant d'un vecteur couleur. Ainsi, un objet illuminé par W_1 sera, après transformation, tel qu'il serait vu si on avait changé de « lampe », et qu'on avait remplacé W_1 par W_2. La transformation étant linéaire, elle peut être exprimée sous forme matricielle. Soit $\mathcal{M}_{XYZ_{W_1} \to XYZ_{W_2}}$ cette matrice de passage. Le calcul de $\mathcal{M}_{XYZ_{W_1} \to XYZ_{W_2}}$ est effectué selon les étapes suivantes:

1. Transformer l'espace XYZ_{W_1} en un domaine de réponse conique que l'on noteras (ρ, γ, β) à l'aide d'une matrice $\mathcal{M}_A$.
2. Modifier les composantes des vecteurs couleur selon des facteurs dépendants des blancs de référence source et destination.
3. Calculer la transformation inverse de l'étape 1. La référence de blanc de la transformation inverse n'est pas modifiée par l'adaptation chromatique: si à l'étape 1 la référence de blanc W_1 était utilisée, cette même référence de blanc sera utilisée dans cette étape.

Nous voyons que la méthode est extrêmement simple puisqu'elle se résume à une transformation matricielle dans le domaine XYZ par la matrice $\mathcal{M}_{XYZ_{W_1} \to XYZ_{W_2}}$ suivante :

$$\mathcal{M}_{XYZ_{W_1} \to XYZ_{W_2}} = \mathcal{M}_A \cdot \begin{bmatrix} \rho_2/\rho_1 & 0 & 0 \\ 0 & \gamma_2/\gamma_1 & 0 \\ 0 & 0 & \beta_2/\beta_1 \end{bmatrix} \cdot \mathcal{M}_A^{-1} \tag{4.27}$$

[(xxx)] cf. annexe §A.3.2.1

avec:

$$(\rho_i, \gamma_i, \beta_i) = (X_{w_i}, Y_{w_i}, Z_{w_i}) \cdot \mathcal{M}_A \ , \ i \ \in \ \{1, 2\}$$

Elle dépend par contre de trois données qui sont: la matrice de transformation $\mathcal{M}_A$, l'illuminant de départ et l'illuminant de destination. Concernant la matrice de transformation $\mathcal{M}_A$, nous n'avons vu que trois définitions différentes : la matrice de *Von Kries*, la matrice identité et la matrice de *Bradford*.

$$\mathcal{M}_{A_{\textit{Von Kries}}} = \begin{bmatrix} 0.40024 & -0.22630 & 0.00000 \\ 0.70760 & 1.16532 & 0.00000 \\ -0.08081 & 0.04570 & 0.91822 \end{bmatrix} \tag{4.28}$$

$$\mathcal{M}_{A_{\textit{Bradford}}} = \begin{bmatrix} 0.8951 & -0.7502 & 0.0389 \\ 0.2664 & 1.7135 & -0.0685 \\ -0.1614 & 0.0367 & 1.0296 \end{bmatrix} \tag{4.29}$$

La mise à l'échelle à partir d'une matrice identité correspondrait à la transformation vers l'espace Lab utilisant le blanc de référence W_1 suivie de la transformation inverse utilisant le blanc de référence W_2. Cette méthode donne les résultats les moins bons selon [Lin]. La matrice de *Bradford* est considérée comme donnant les meilleurs résultats [(xxxi)]. Elle est utilisée dans certains logiciels professionnels. Nous ne travaillerons dans la suite qu'avec la matrice de *Bradford*.

Concernant les illuminants source et destination, le nombre de combinaisons devient rapidement important, et ce même si l'on se restreint aux illuminants standards. Nous allons nous limiter en considérant l'illuminant source fixé au standard D_{65}. Le fait de considérer l'illuminant source comme constant n'est pas tellement restrictif dans la mesure où cela nous facilite l'analyse des résultats. La seule variable est donc l'illuminant de destination. Observons l'effet de cette transformation sur quelques images.

Résultats visuels du changement d'illuminant: Deux extraits de la transformation d'illuminant proposée sont donnés en figure 4.34 et 4.35. Nous avons pour cela utilisé un illuminant de source D_{65} et les illuminants de destination suivant: A, B, C, E, D_{50}, D_{55} et D_{75}. Bien que cela ne soit certainement pas flagrant à l'impression, nous avons effectivement un changement global de teinte dans l'image.

Le changement vers le blanc A (lampe à tungstène) est assez surprenant: toute l'image prend une teinte prononcée vers le rouge. Il s'agit néanmoins d'une transformation assez réaliste, même si nous pensons que les réponses des caméras réelles sont différentes (peut être légèrement plus claire et « moins » rouge, mais il est difficile d'expliciter précisément cette sensation). La discrimination entre les pixels de peau et de non-peau risque par contre d'être plus délicate sous cette transformation, ce qui sera confirmé par l'analyse quantitative.

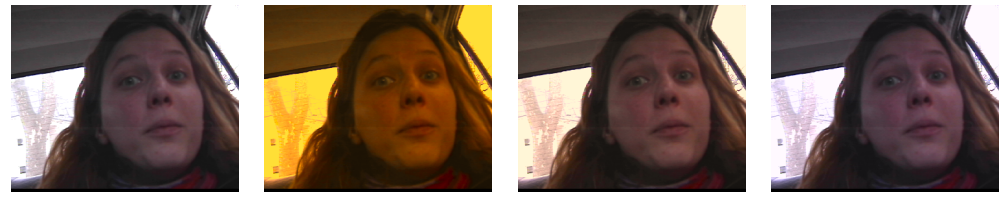
Les changements se voient particulièrement dans les blancs. La transformation vers le B semble assez proche de celle du D_{50}, et celle du C à celle du D_{75}. La transformation vers le E semble être entre le D_{55} et le D_{75}.

Les résultats visuels semblent très satisfaisants. À présent, nous allons faire une étude quantitative du changement d'illuminant sur les estimateurs du modèle de *von Mises* proposé.

4.4.2.3. Analyse quantitative du changement d'illuminant dans le domaine HLS

L'objectif de cette partie est d'extraire des invariants de peau, et plus particulièrement de quantifier leur degré d'invariance sous des conditions variées. Nous allons tout d'abord décrire le protocole de test, nous présenterons ensuite les résultats accompagnés de notre analyse.

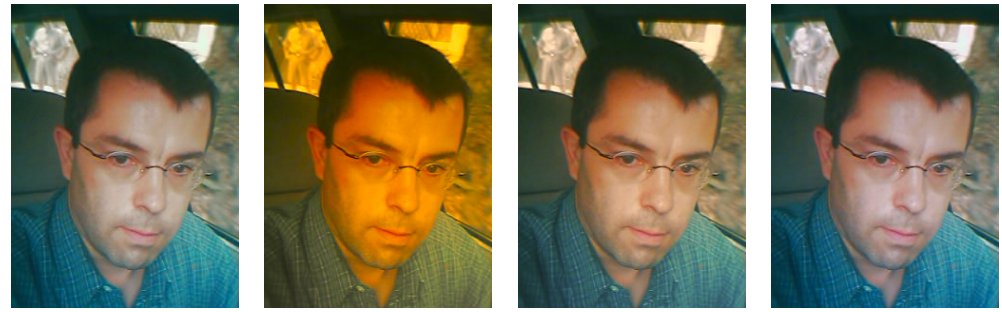
[(xxxi)] Nous n'avons pu déterminer correctement la qualité objective de l'adjectif « meilleur ».



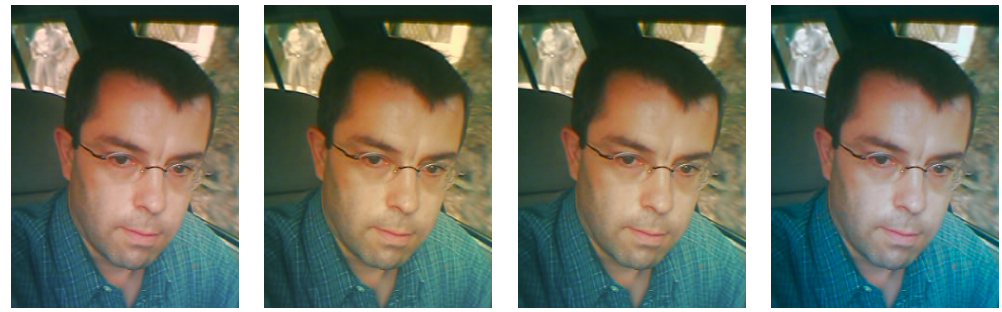
(a) Image originale (b) RefW A (c) RefW B (d) RefW C

(e) RefW E (f) RefW D_{50} (g) RefW D_{55} (h) RefW D_{75}

Fig. 4.34.: Exemple de changement d'illuminant - Base *Hitachi*.



(a) Image originale (b) RefW A (c) RefW B (d) RefW C



(e) RefW E (f) RefW D_{50} (g) RefW D_{55} (h) RefW D_{75}

Fig. 4.35.: Exemple de changement d'illuminant - Base *Logitech*.

Description des tests: Les tests menés pour le changement de référence de blanc supposent le blanc de départ à D_{65} et appliquent successivement la transformation de *Bradford* vers les blancs A, B, C, E, D_{50}, D_{55}, D_{75}, ce qui nous a semblé suffisamment réaliste et exhaustif. Les mesures effectuées sur chaque image (ie. individu) sont les suivantes:

1. la teinte moyenne $\bar{\mu_0}$.
2. la dispersion $\bar{R_0}$.
3. les mêmes paramètres pour les moments centrés d'ordre 1, 2 et 3 : la teinte $\bar{\mu_1}$, $\bar{\mu_2}$, $\bar{\mu_3}$ ainsi que les dispersions associées $\bar{R_1}$, $\bar{R_2}$, $\bar{R_3}$.

Pour chacun de ces paramètres, nous avons besoin d'un sous-ensemble d'intégration. Nous disposons déjà des masques de zone de peau grâce à l'étude précédente. Ces masques supposent néanmoins une connaissance *a priori* des lieux de peau, aussi il est intéressant d'analyser les estimations des paramètres en supprimant cette connaissance *a priori*. L'estimation des paramètres $\bar{\mu_0}$ et $\bar{R_0}$ pour un individu se résume donc ainsi:

1. l'ensemble *peau* connu *a priori* sert de sous-ensemble d'intégration pour le calcul.
2. le calcul est effectué sur des voisinages sans intervention *a priori* des lieux de peau. Les paramètres estimés localement sont ensuite intégrés (moyenne et variance des estimateurs $\bar{\mu_0}$ et $\bar{R_0}$ [(xxxii)]) sur les lieux de peau connu *a priori*.

Ce schéma est repris sur la figure 4.36; nous indicerons les paramètres globaux et locaux respectivement par g et l. Pour le calcul des moments supérieurs $\bar{\mu_i}$ et $\bar{R_i}$, $i \in \{1, 2, 3\}$, il est également possible d'effectuer des combinaisons: $\bar{\mu_{0,g}}$ peut donner lieu à l'estimation de $\bar{\mu_{1,l}}$, etc. Nous n'avons pas entrepris de telles combinaisons: les estimateurs de premier ordre locaux et globaux servent à construire les estimateurs d'ordre supérieur locaux et globaux, sans croisement.

Concernant les voisinages mentionnés, deux rayons seront utilisés: respectivement 1 (voisinage hexagonal 6-connexe) et 10 (boule euclidienne).

Ces estimateurs sont associés à une distribution circulaire, dont l'angle est donné par la teinte. Or d'après le §4.2.2.1, il existe deux versions: celle sans pondération par la saturation (cf. equation 4.9) et celle utilisant la saturation comme facteur de pondération (cf. equation 4.15).

Angle moyen: L'angle moyen est un paramètre délicat à traiter à cause de la circularité de l'espace. Des résultats préliminaires d'histogrammes de teinte, calculés sur la totalité du visage, sont donnés sur les figures 4.37, 4.38. Ces résultats intra-individu, montrent une importante variabilité de direction pour la teinte moyenne, et par rapport à d'illuminant. Nous observons ce phénomène sur la totalité des bases (cf. figure 4.39); ceci nous pousse à rejeter la teinte moyenne comme descripteur invariant à l'illuminant. Ainsi, les méthodes qui cherchent à séparer la peau de la *non-peau* uniquement avec des approches de seuillages sur les angles de teinte ne sont ni précises, ni robustes. Ceci nous amène dans la même mesure à rejeter les approches présentées en [SP96] et [**?**]: ces approches appliquent un seuil angulaire assez large, et utilisent ensuite un filtrage morphologique basique de manière à introduire la connexité des points de peau. Les conditions d'utilisation de ces méthodes restent toutefois valables lorsque les conditions s'y prêtent fortement: lorsque le fond est par exemple composé d'une couleur saturée très différente de la teinte de peau (vert ou bleu par exemple), ces méthodes sont suffisantes.

Dans notre problème, nous pouvons considérer la teinte comme un **invariant faible**. Son utilisation sera toutefois réduite à la suivante: un seuil large sur la direction de teinte moyenne, calculée localement, sera appliqué à l'image pour écarter les pixels de *non-peau* et fera office de premier filtrage. La probabilité de supprimer des faux négatifs sera de cette manière faible. Déterminer avec grande précision les limites du seuil n'avait par contre pas trop d'intérêt, et nous sommes restés sur nos observations: tous ce qui n'est pas entre -80 et 80 degrés est considéré comme étant de la *non-peau* [(xxxiii)].

[(xxxii)] nous avons également analysé les bornes supérieures et inférieures de $\bar{\mu_0}$ et $\bar{R_0}$ mais nous n'avons pas réussi à exploiter ces résultats.

[(xxxiii)] La peau peut en effet prendre une teinte emprunte de l'illuminant, tendant parfois vers le vert ou le bleu.

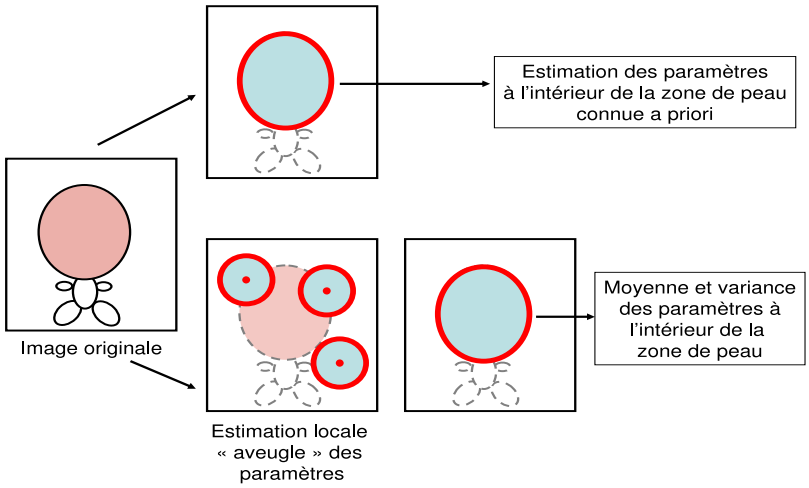


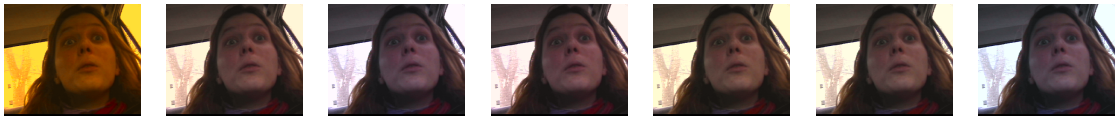
Fig. 4.36.: Calcul local et global des paramètres $\bar{\mu_0}$ et $\bar{R}_0$. Les zones bleues servent de support pour l'estimation des paramètres. La première méthode utilise la connaissance *a priori* des zones de peau. La deuxième méthode effectue d'abord un calcul local « aveugle » des paramètres, et réduit ensuite les valeurs locales estimées à leur moyenne et variance en faisant intervenir la connaissance *a priori* des zones de peau; ceci purement à des fins d'exploitation des résultats.

D'après ces résultats, le paramètre de dispersion semble extrêmement intéressant, car la teinte moyenne est très confinée par rapport à la direction moyenne. Nous allons à présent étudier ce paramètre.

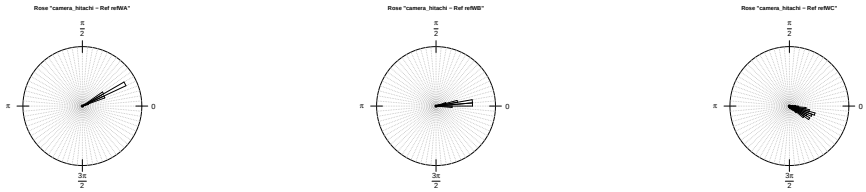
Dispersion: Le nombre de résultats concernant la dispersion est assez important. Aussi, dans un premier temps, nous allons analyser le comportement des dispersions calculées avec et sans la pondération de saturation. La figure 4.40 montre les résultats pour la base « Logitech », et met en concurrence les dispersions $\bar{R}_{0,1,2,3}$ et l'illuminant.

Nous pouvons déjà remarquer que les dispersions concernant la peau sont assez faibles (ie. $\bar{R}_{0,1,2,3}$ relativement proche de 1), que cela soit sur les calculs avec et sans pondération par la saturation. Nous remarquons également sur cet exemple que l'ajout de la pondération par saturation conduit à une variabilité de la dispersion moins élevée pour l'ensemble des transformations d'illuminant, toutefois assez proches des homologues sans pondération. Ceci s'explique de la manière suivante: les zones faiblement saturées sont plus sensibles au changement d'illuminant et naturellement engendrer une variabilité au niveau de la teinte. La pondération par le module (id est saturation) limite l'influence des points variables. Cela rejoint ce que nous avons énoncé dans le paragraphe §4.2.2.1 page 119 concernant la « fiabilité » des mesures lorsque celles-ci sont pondérées par un facteur de variance. Nous choisirons par la suite de ne traiter que les mesures obtenues par pondération.

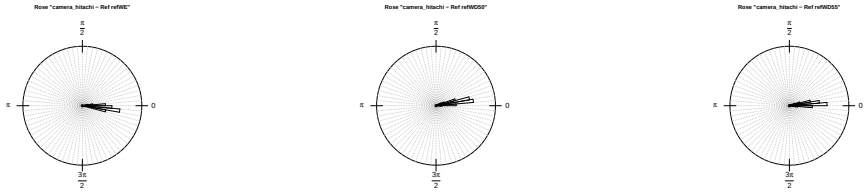
La figure 4.41 montre les résultats pour d'autres bases. En première observation, le paramètre de dispersion est très stable par changement d'illuminant. En effet, pour la majeure partie des bases, les dispersions prennent des valeurs supérieures à 0.9. Cela est également vrai pour les bases très hétérogènes comme la base *Web1* en figure 4.41(c), qui donne des résultats difficilement exploitables dans d'autres espaces couleurs. Nous remarquons également une dégradation nette des résultats pour l'illuminant D_{75}, quelle que soit la base. L'illuminant C donne également des résultats de dispersion moins bons que les autres illuminants. Ceci s'explique par le fait que les chrominances bleues de ces deux illuminants sont particulièrement éloignées de celle de la peau. L'application de la transformation dégraderait alors l'information de chrominance, et expliquerait un confinement moins bon des teintes autour d'une teinte moyenne.



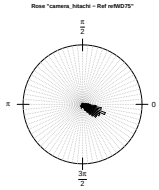
(a) RefW A (b) RefW B (c) RefW C (d) RefW E (e) RefW D_{50} (f) RefW D_{55} (g) RefW D_{75}



(h) RefW A (i) RefW B (j) RefW C



(k) RefW E (l) RefW D_{50} (m) RefW D_{55}



(n) RefW D_{75}

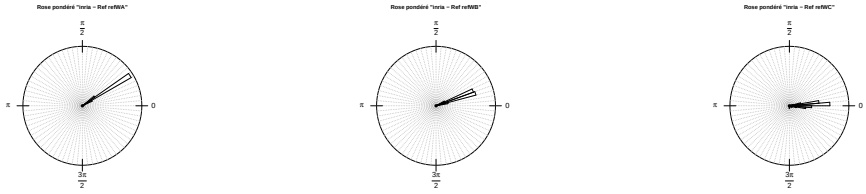
Fig. 4.37.: Influence sur la teinte du changement d'illuminant sur une image de la base *Hitachi*. Nous voyons que la composante de peau est assez concentrée autour d'une teinte moyenne, moyenne qui par contre n'est pas robuste aux changements d'illuminants.

Les résultats concernant l'illuminant A sont également à prendre avec une certaine réserve. Nous avons vu que l'application de cette transformation teinte l'image vers le rouge, ce qui signifierait également que les propriétés discriminantes seront moins bonnes dans cet espace.

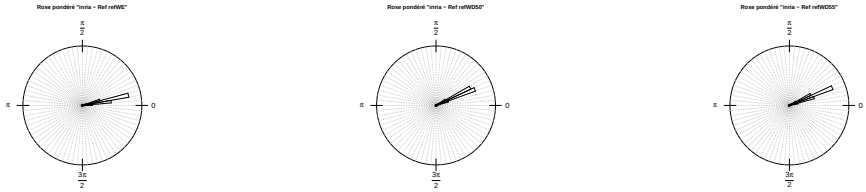
Nous remarquons également que la dispersion reste faible pour les moments supérieurs, c'est à dire pour $\bar{R}_0$, mais également pour $\bar{R}_1$, $\bar{R}_2$ et $\bar{R}_3$. Il est également intéressant de se pencher sur la différence entre la mesure locale et globale (visage) de la dispersion. La figure 4.42 montre que cette valeur reste très faible, ce qui signifie que la dispersion locale est une mesure assez significative de la dispersion calculée sur tout le visage. De la même manière, la variance de $\bar{R}_{0,l}$ reste très faible pour l'ensemble des bases et des illuminants (cf. figure 4.43), ce qui fait de la moyenne de la dispersion locale $E_l(\bar{R}_{i,l})$ un bon estimateur pour la dispersion de l'ensemble du visage $\bar{R}_{i,g}$.

Conclusion sur la dispersion: Nous avons vu que la moyenne n'était pas un invariant suffisamment robuste pour la détection des pixels de peau. Il pouvait par contre être utilisé comme *invariant faible*, laissant échapper un grand nombre de faux-positif, mais un faible nombre de faux négatif. Nous avons ensuite étudié les dispersions associées aux moments angulaires et nous en avons déduit que la dispersion agit effectivement comme un invariant extrêmement pertinent pour nos applications. Nous avons comparé les dispersions calculées localement et globalement (sur la totalité du visage) et nous avons

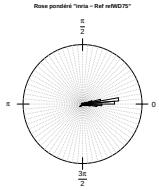
(a) RefW A (b) RefW B (c) RefW C (d) RefW E (e) RefW D_{50} (f) RefW D_{55} (g) RefW D_{75}



(h) RefW A (i) RefW B (j) RefW C



(k) RefW E (l) RefW D_{50} (m) RefW D_{55}



(n) RefW D_{75}

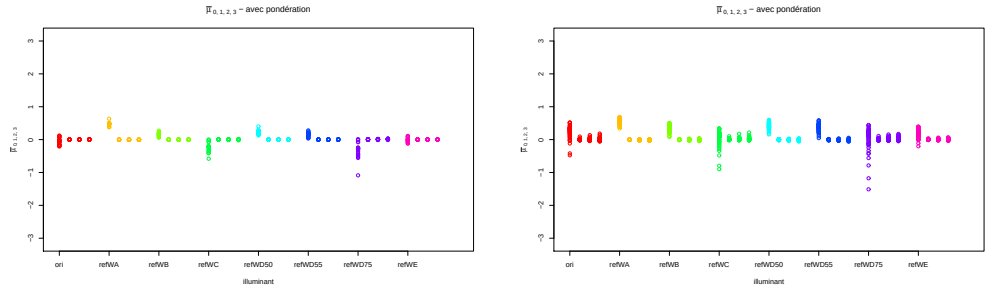
Fig. 4.38.: Influence sur la teinte du changement d'illuminant sur une image de la base *Inria*

observé que la différence moyenne de ces deux estimateurs était très faible, et que la moyenne de l'estimateur local est également robuste (variance faible). Ce qui nous permet par la suite d'appliquer sans réelle retenue les estimateurs locaux pour améliorer la détection des pixels de peau.

4.4.2.4. Conclusion intermédiaire

Nous avons également introduit un modèle circulaire pour la classe de peau, le modèle de *von Mises*, dépendant de deux paramètres facilement estimables. Afin de mener des mesures correctes de ces deux paramètres sur l'ensemble des bases, nous avons introduit une méthode de changement d'illuminant colorimétrique. Cette approche est également une contribution originale de nos travaux.

Nous avons ensuite mené l'étude des paramètres sur l'ensemble des bases de test à notre disposition, nous en avons déduit que la teinte n'est pas un descripteur fort de la classe de peau, mais qu'il peut toutefois être utilisé comme un descripteur faible. Couplée à la très faible dispersion de la classe peau, dispersion très faible également pour les moments circulaires centrés de premier et second ordre, nous avons obtenu une méthode beaucoup plus robuste que celles existantes pour décrire la peau uniquement à partir de l'information de chrominance.



(a) Résultats sur la bas « Hitachi »

(b) Résultats sur la bas « Web1 »

Fig. 4.39.: Teinte moyenne locale, intégrée sur la totalité du visage. La légende est la suivante: il y a 8 groupes (couleurs) de 4 verticales chacune. La couleur correspond à l'illuminant d'arrivée et est (respectivement de gauche à droite): l'illuminant original (image originale sans transformation), A, B, C, D_{50}, D_{55}, D_{75} et E. La teinte moyenne $\bar{\mu}$ calculée localement, est ensuite intégrée sur le visage, et sa moyenne est extraite. Ceci est fait pour les teintes $\bar{\mu_0}$, $\bar{\mu_1}$, $\bar{\mu_2}$ et $\bar{\mu_3}$, représentée respectivement de gauche à droite, et verticalement dans chaque groupe (soit 4×8 points par individu). L'échelle en ordonnée est dans $[-\pi, \pi]$, la taille boule de calcul est de 10 pixels.

4.4.3. Marquage des zones de peau

Maintenant que nous savons approximativement à quoi doit correspondre la peau, nous allons essayer de créer un marquage correct des zones de peau en condition « réelle ». Nous restreignons la discussion au cadre d'une détection issue directement des informations couleur, sans autre type de traitement. Ainsi, nous ne discuterons pas de filtrage par redondance temporelle pour les séquences vidéos, ni d'utilisation de modèles pour la forme du visage.

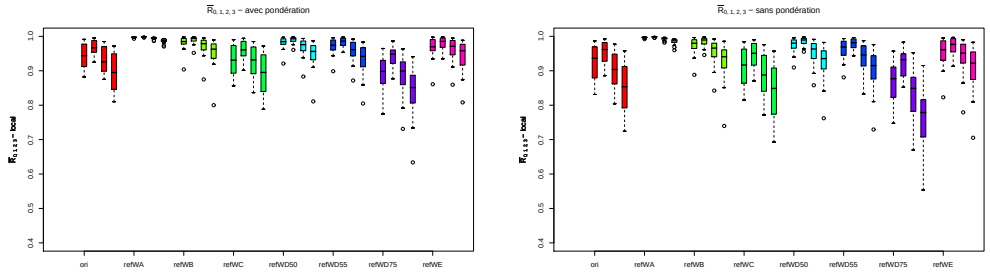
D'après ce que nous avons vu tout au long de cette partie, et plus particulièrement sur les caractéristiques de la peau dans l'espace couleur HLS, les moments circulaires semblent être des mesures pertinentes et robustes pour ce genre de détection. Voyons à présent l'effet d'un simple seuillage sur les deux paramètres que sont la teinte moyenne et la dispersion, prises toutes deux localement.

Nous avons qualifié la moyenne en teinte comme étant un paramètre *faible*, c'est à dire soumis à des fluctuations importantes. Cela n'exclut en rien cette mesure pour la détection. Nous avons utilisé un seuillage circulaire très large, en prenant par exemple le domaine $[-80^o, 80^o]$. Nous avons parallèlement calculé la dispersion en teinte, et nous avons gardé les points dont la dispersion est supérieure à la valeur 0.99 (soit une variance circulaire inférieure à 0.01). Nous avons également effectué ce seuillage sur des moments centrés d'ordre 1 et 2. Voici les résultats sur les bases vidéos « Hitachi » et « Logitech ».

Base « Logitech » Deux extraits de cette base sont donnés sur les figures 4.44 et 4.45.

Le protocole expérimental, similaire à l'ensemble des images, est le suivant. Nous avons tout d'abord effectué un seuillage en teinte (figure 4.44(a)), en ne considérant que les pixels sans leur voisinage. C'est précisément le type de traitement qui est fait classiquement, et nous présentons cela de manière à considérer l'apport de notre approche. Nous avons ensuite calculé localement la moyenne en teinte. Pour ce calcul, nous avons utilisé un disque euclidien d'un rayon de 10 pixels. Cette moyenne nous produit une image de teinte et dispersion moyenne locale. La teinte moyenne locale, seuillée de la même manière que l'image précédente, est donnée en figure 4.44(b).

Nous avons ensuite seuillé la dispersion (figure 4.44(c)). La figure 4.44(e) est une intersection entre le seuillage en teinte et celui en dispersion. L'intersection est suivie d'un filtrage très basique, composé d'une ouverture de taille 4 et une fermeture de taille 7, de manière à supprimer les petites particules



(a) Avec pondération par saturation

(b) Sans pondération par saturation

Fig. 4.40.: Comparaison de la dispersion moyenne avec et sans pondération par la saturation. La dispersion $\bar{R}$ calculée localement est intégrée sur le visage, et sa moyenne est extraite. Chaque barre représente une valeur de dispersion pour un moment et un illuminant donnés. La légende des couleurs est similaire à celle de la figure 4.39. Dans chaque groupe de couleur, les dispersions $\bar{R}_0$, $\bar{R}_1$, $\bar{R}_2$ et $\bar{R}_3$ sont représentées de gauche à droite (soit 4×8 points par individu). Chaque barre est le résumé pour tous les individus de la base de donnée « Logitech ». L'échelle en ordonnée est dans $[0.4, 1]$, la taille boule de calcul est de 10 pixels.

et à joindre les grandes.

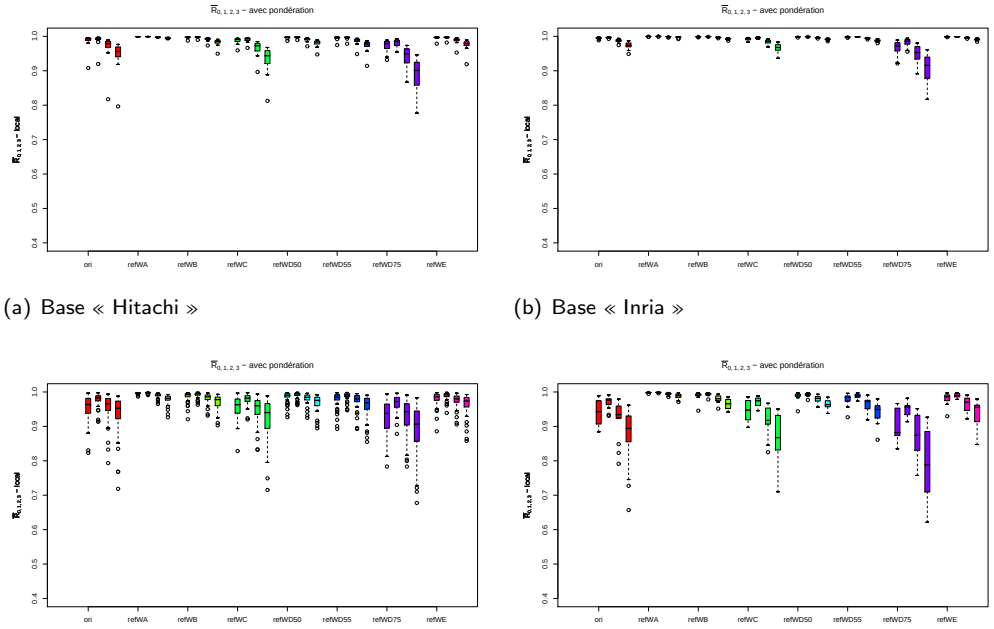
Les images 4.44(d) et 4.44(f) représentent les opérations similaires, mais sur des moments circulaires supérieurs. La moyenne en teinte permet dans un premier temps de centrer la teinte. Un facteur multiplicatif de 2 (deuxième ordre) est ensuite appliqué à cette image de teinte centrée. La composante de saturation reste par contre inchangée. Sur cette nouvelle image, la moyenne circulaire locale est calculée de nouveau, ce qui donne deux images, une de teinte et une de dispersion. Cette nouvelle image de teinte ne nous intéresse pas, contrairement à l'image de dispersion (figure 4.44(d)). La teinte moyenne seuillée (figure 4.44(b)) est utilisée de nouveau pour générer une image d'intersection (figure 4.44(f)), de la même manière que précédemment.

Pour ce matériel, nous pouvons remarquer la qualité du seuillage initial en teinte uniquement (figures 4.44(a) et 4.45(a)): le seuillage donne des résultats tout à fait corrects. L'utilisation d'une moyenne spatiale en teinte (figures 4.44(b) et 4.45(b)) lisse naturellement les contours des ensembles. La majeure partie du visage est incluse dans ce seuillage initial, mis à part une zone surexposée. Les vêtements de la personne font que les contours de la partie basse du visage sont très correctement placés. Cependant les contours de la partie haute de la personne ne le sont pas. En effet, les cheveux d'une part et le plafond de l'habitacle d'autre part, ont une teinte très proche du rouge.

Le seuillage en dispersion permet d'affiner les contours de la partie haute. Comme nous pouvons le voir sur les figures 4.44(c) et 4.44(d), la dispersion crée des contours au niveau des cheveux du conducteur. Elle ajoute en revanche une imprécision sur certaines zones, notamment avec la partie surexposée du visage. Dans cette partie, l'œil et la bouche, associés à l'élément structurant de taille relativement importante, créent une zone de dispersion faible.

Enfin, la combinaison avec le seuillage en teinte permet ensuite de ne garder que les composantes connexes pertinentes dans l'image. Ce dernier est correctement marqué, bien que la totalité du visage ne soit pas incluse dans ce marqueur.

Pour l'image « Thomas » (figure 4.45), une forte illumination éclaire le visage du conducteur. L'ensemble des détections y est sensible et divise le visage en deux parties. La dispersion seule, calculée à partir de la moyenne (image 4.45(c)) fournit un contour de visage pertinent. Par rapport à la teinte, la partie droite du visage (à gauche sur l'image) est correctement séparée du dossier. De plus, les cheveux présentent une séparation plus franche avec la zone de peau que celle obtenue par la teinte uniquement. Une perte de résolution est toutefois observée sur la partie gauche du visage (à droite sur l'image), précisément à cause de la forte luminosité. La zone fortement exposée à la lumière devient presque



(a) Base « Hitachi »

(b) Base « Inria »

(c) Base « Web1 »

(d) Base « Web2 »

Fig. 4.41.: Dispersion moyenne par site et par base (cf. 4.40 pour légende).

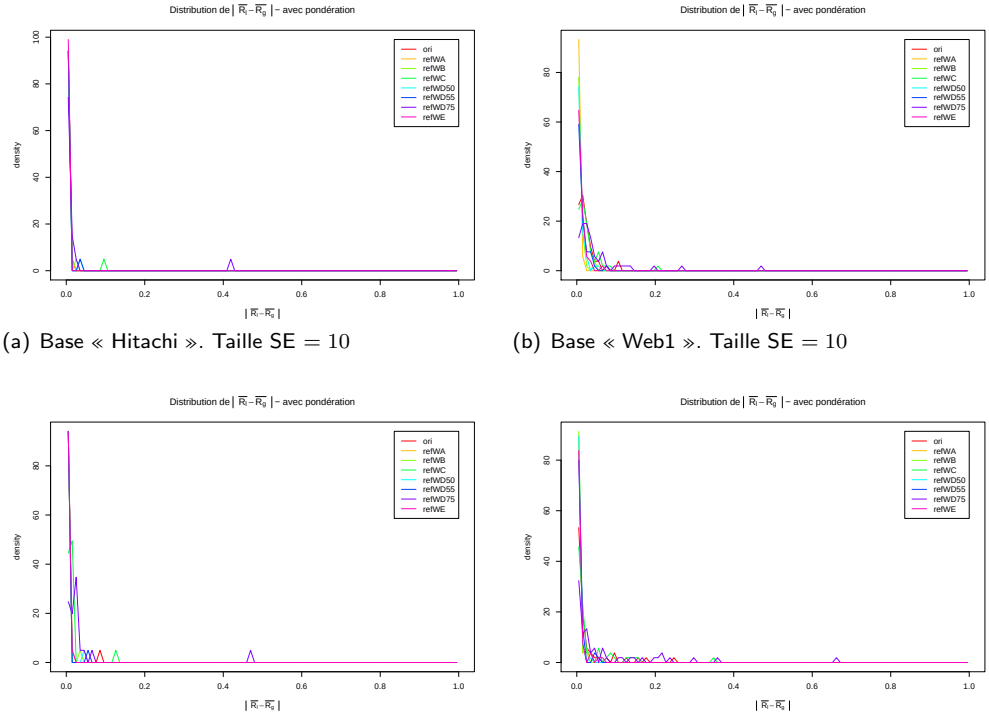
achromatique, ce qui explique la création d'un contour dans la dispersion. Le voisinage utilisé, de taille relativement importante, déplace d'autant plus cette ligne de contours.

Base « Hitachi » Deux extraits de cette base sont donnés sur les figures 4.46 et 4.47. Contrairement à la caméra « Logitech », le seuillage en teinte ne donne ici pas d'information pertinente. En effet, sur ces images, l'ensemble de la scène tire vers le rouge, et le seuillage en teinte ne fournit pas d'information utile. Bien que s'agissant du même véhicule que pour la caméra « Logitech », le traitement de la teinte ne produit pas les mêmes résultats. En effet, si simple seuillage large en teinte est suffisant pour la base « Logitech », cela s'avère totalement insuffisant pour la base « Hitachi » (voir par exemple les images 4.46(a) et 4.46(b)).

En revanche, le seuillage en dispersion répond à nos exigences. Nous pouvons par ailleurs remarquer sur les images 4.46(c) et 4.46(e) que la personne à l'arrière est également correctement détectée. Cette détection disparaît dans des moments supérieurs. Ceci s'explique par le fait que le calcul du moment d'ordre 2 effectue deux moyennes locales successives, le premier pour la teinte moyenne qui sert à centrer la variable de teinte, et le second pour le calcul effectif du moment; ce double calcul entraîne donc un voisinage plus grand et fait naturellement disparaître les objets « petits », entourés de zones de dispersion faible. Ce phénomène rejoint la perte de résolution évoquée plus haut.

Enfin, la figure 4.47 montre des résultats également tout à fait corrects dans l'application de marquage. Le seuillage en teinte échoue pour les mêmes raisons que celles évoquées pour la figure 4.46. La dispersion apporte cependant des contours pertinents au visage, ainsi qu'un très bon marquage. À noter cependant la détection de zone fortement lumineuses (à travers la vitre, de part et d'autre du visage). Dans ces zones, nous avons une faible saturation (forte luminance) mais qui n'est toutefois pas nulle. L'ensemble de l'image tirant encore une fois vers le rouge, la teinte existe, et possède une direction privilégiée. Ceci fait que la dispersion est grande dans ces zones.

À noter également une application envisageable pour la dispersion, que nous avons également vérifiée sur nos bases de tests. Lorsque le visage, vu de face, occupe une grande partie de l'image, les zones correspondant aux yeux et à la bouche présentent des dispersions plus faibles que le reste du visage.



(a) Base « Hitachi ». Taille SE $= 10$

(b) Base « Web1 ». Taille SE $= 10$

(c) Base « Hitachi ». Taille SE $= 1$

(d) Base « Web1 ». Taille SE $= 1$

Fig. 4.42.: Distribution de $|\bar{R}_{0,l} - \bar{R}_{0,g}|$. Pour l'ensemble, $|\bar{R}_{0,l} - \bar{R}_{0,g}|$ reste très faible à part quelques cas isolés. Ceci est vrai pour les tailles d'intégration locales de 1 et 10.

Ainsi, on peut observer des « trous » dans les zones de peau (figures 4.47(c) et 4.47(e)) au niveau des yeux. Pour les lèvres, cela dépend plus de la personne.

Conclusion Malgré la simplicité du traitement, nous constatons sur ces quelques exemples que l'ajout de la mesure de dispersion dans le seuillage en teinte donne une information tout à fait pertinente dans la tâche de marquage des zones de peau. La dispersion crée également des contours intéressant entre les cheveux d'une personne et sa peau, qu'il est difficile d'avoir uniquement par la teinte, tant ces deux parties sont proches chromatiquement [(xxxiv)]. Rappelons que nous avons utilisé les mêmes paramètres pour les deux bases vidéos (« Logitech » et « Hitachi »), acquises dans des conditions absolument non comparables et avec du matériel différent.

[(xxxiv)] les cheveux contiennent de la mélanine, tout comme la peau.

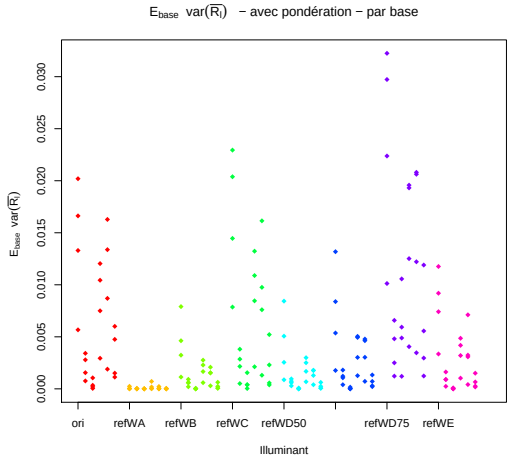
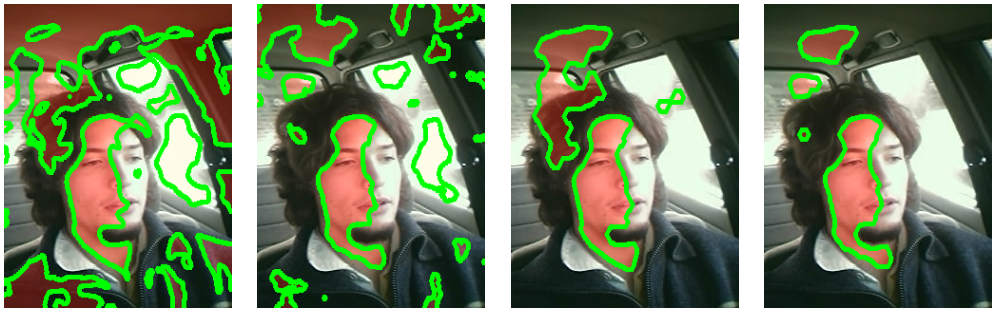


Fig. 4.43.: Variance de $\bar{R}_{i,l}, i \in \{0,1,2,3\}$ - toutes bases confondues. Nous observons une variance très faible, pour toutes les mesures locales de la dispersion, toutes les bases et tous les illuminants. Le maximum de toutes ces valeurs est à 0.03 (échelle en ordonnée sur $[0, 0.03]$).

(a) Seuillage en teinte

(b) Seuillage en teinte moyenne & locale



(c) Seuillage en dispersion - moment 0

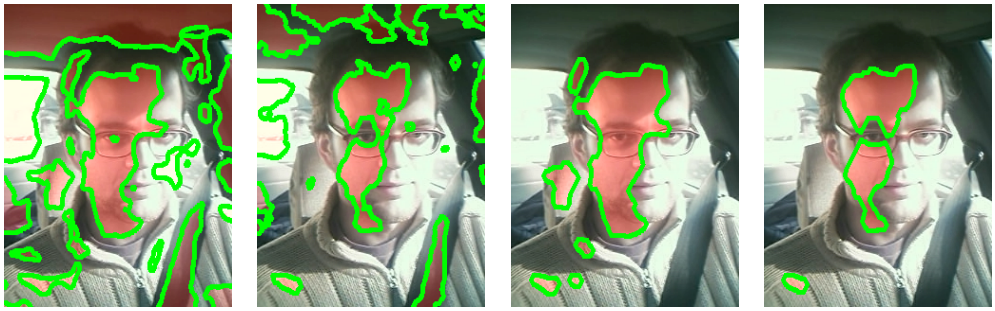
(d) Seuillage en dispersion - moment 2

(e) Combinaison des seuillages (b) & (c) - moment 0

(f) Combinaison des seuillages (b) & (d) - moment 2

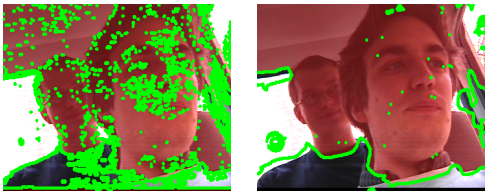
Fig. 4.44.: Résultats de l'utilisation de la dispersion et de la teinte. Séquence « Gabriel » - base « Logitech ». Les zones marquées en rouge (intérieures aux lignes vertes) sont celles détectées comme étant la peau, selon les critères énoncés. Excepté pour (a), l'élément structurant utilisé est un disque euclidien de rayon 10.

(a) Seuillage en teinte (b) Seuillage en teinte moyenne & locale

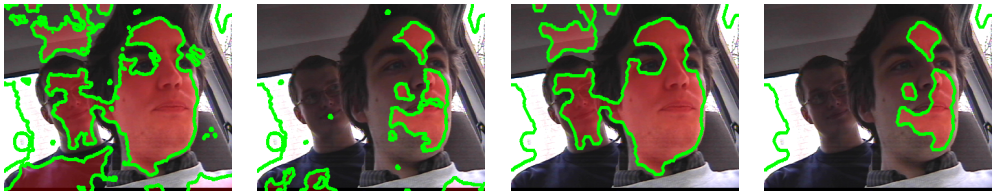


(c) Seuillage en dispersion - moment 0 (d) Seuillage en dispersion - moment 2 (e) Combinaison des seuillages - moment 0 (f) Combinaison des seuillages - moment 2

Fig. 4.45.: Résultats de l'utilisation de la dispersion et de la teinte. Séquence « Thomas » - base « Logitech ».

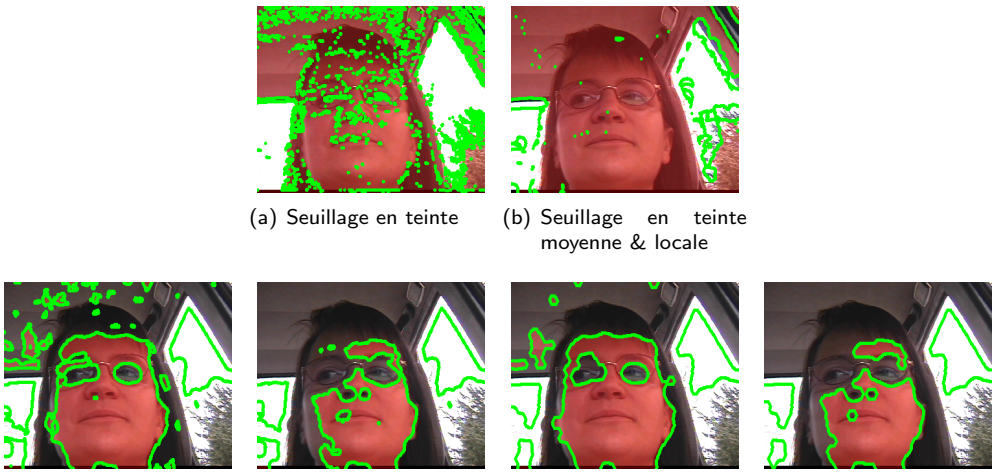


(a) Seuillage en teinte (b) Seuillage en teinte moyenne & locale



(c) Seuillage en dispersion - moment 0 (d) Seuillage en dispersion - moment 2 (e) Combinaison des seuillages - moment 0 (f) Combinaison des seuillages - moment 2

Fig. 4.46.: Résultats de l'utilisation de la dispersion et de la teinte. Séquence « Romain » - base « Hitachi ».



(a) Seuillage en teinte (b) Seuillage en teinte moyenne & locale

(c) Seuillage en dispersion - moment 0 (d) Seuillage en dispersion - moment 2 (e) Combinaison des seuillages - moment 0 (f) Combinaison des seuillages - moment 2

Fig. 4.47.: Résultats de l'utilisation de la dispersion et de la teinte. Séquence « Beatriz » - base « Hitachi ».

4.5 Détection d'objets en mouvement

Les développements de cette partie concernent l'apport de la couleur dans le cadre de la détection d'objets en mouvement pour la vidéosurveillance.

Nous allons dans un premier temps détailler les principales techniques existantes de détection d'objet en mouvement. Nous nous servirons ensuite des métriques couleurs (§4.2.1.1 page 106) et des statistiques angulaires (§4.2.2.1 page 119) pour améliorer certains résultats.

4.5.1. Principales techniques de détection de mouvement

Chaque image est une vue statique de la scène et de la position des objets. Pour découvrir le mouvement de ces objets, il faut naturellement analyser une séquence d'images indexées par le temps.

De façon à appuyer les développements sur la couleur, nous présentons ici les principales méthodes de détection de mouvement. Ceci nous permettra ensuite de faire les adaptations pour la couleur.

Gradients temporels

L'idée est de comparer deux ou trois images supposées successives dans le temps, et de déduire par *différence mutuelle* les zones en mouvement: ces dernières sont caractérisées par une différence élevée alors que les zones d'observation fixes ont une différence faible. Cette technique est très analogue au calcul du gradient spatial, la seule différence porte sur l'élément structurant utilisé contenant ici une dimension temporelle; c'est pourquoi on parle de *gradient temporel*. L'avantage de cette méthode est que son calcul mobilise peu de ressources.

Deux inconvénients majeurs limitent toutefois son utilisation. Le premier est l'absence de mémoire : l'information des zones de différence n'est pas exploitée pour les détections futures. Ce dernier point rend la méthode particulièrement fragile par rapport aux différents bruits. Le second inconvénient vient du fait que les contours ne sont pas fermés. En effet, le gradient est nul pour des déplacement parallèles aux contours, comme l'explique la figure 4.48.

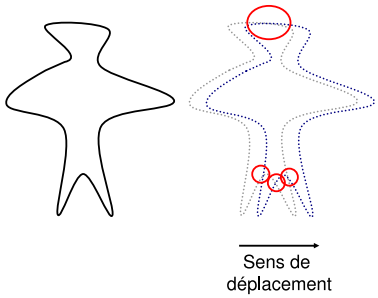


Fig. 4.48.: Ouverture des contours du gradient temporel. La zone indiquée en rouge en haut de la figurine présente un contour parallèle au sens de déplacement. Les quelques zones dans la partie inférieure possèdent des contours d'intersection non nulle pour les deux temps d'observation. Ces deux phénomènes ont pour conséquence une ouverture du gradient temporel

Ce défaut doit être résolu avant l'application d'outils de la ligne de partage des eaux. Plusieurs moyens sont à notre disposition pour fermer ces contours. L'utilisation d'opérateurs morphologiques tels que composition d'ouverture et de fermeture affecte énormément les contours d'intérêt. Une méthode pourtant simple, de lissage par convolution de gaussienne amène néanmoins à une fermeture suffisante des contours pour une application de la ligne de partage des eaux, et réduit considérablement le bruit de fond.

Somme pondérée

La technique précédente ne permet pas d'utiliser de l'information temporelle de plus long terme. La méthode de « somme pondérée » répond à ce problème par une intégration de l'information dans la

construction d'une image de *fond*. Chaque nouvelle trame $\mathcal{I}^{t+1}$ est combinée avec l'image de fond précédente $\mathcal{F}^t$, afin de fournir une nouvelle image de fond $\mathcal{F}^{t+1}$. La combinaison se fait de manière plus ou moins astucieuse. L'idée la plus simple est une combinaison linéaire de la forme

$$\mathcal{F}^{t+1} = \alpha \cdot \mathcal{F}^t + (1-\alpha) \cdot \mathcal{I}^{t+1}$$

où α est un facteur de pondération. Les zones de mouvement sont ensuite détectées par soustraction avec l'image de fond. Les variantes de cette méthode se basent sur la détermination de α - constant ou pixel-dépendant sur l'image, fonction de l'image de fond, ou enfin l' intégration d'information tierce de type *mouvement*. L'avantage principal de cette méthode est qu'elle reste très simple et ne nécessite que deux, voire trois images, l'éventuelle troisième image étant l'image α dans le cas où ce paramètre est pixel-dépendant. Les inconvénients découlent naturellement de cette simplicité relative: les modèles associant fonctionnellement α aux images incidentes ou de fond deviennent rapidement très complexes et le mouvement est détecté sur une petite échelle temporelle. Ainsi, dans les modèles les plus simples, un objet immobile pour un certain nombre de trames sera intégré rapidement dans l'image de fond.

Apprentissage de fond

Le fil directeur entre les méthodes de détection de mouvement exposées est la construction d'une image de référence, dite de fond, de manière à détecter les différences avec cette référence. Les différences sont associées alors aux objets en mouvement, ce qui est une conjecture pouvant être invalidée relativement facilement. D'un point de vue statistique et classification, la soustraction du fond permet l'utilisation des informations acquises au cours du temps pour la classification des pixels d'une nouvelle image en deux classes: les pixels de forme - ou *« foreground »* - d'une part, et les pixels de fond - *« background »* . L'*apprentissage* de l'image de fond suit alors un modèle statistique plus ou moins simple et permettant de faire une classification facile et non coûteuse des pixels.

L'un des modèles des plus populaire est le modèle gaussien. Il est de mise en œuvre simple et approprié pour les scènes très statiques, où seules les variations liées à l'acquisition sont visibles. Ces variations, le plus souvent dues aux chocs thermiques à l'intérieur du capteur, sont modélisés de manière très réaliste par le bruit blanc. Les pixels sont alors des variables aléatoires gaussiennes dont il faut estimer la moyenne et la variance, ce qui est trivial.

Explicitons d'avantage cette méthode : nous nous inspirons ici de [Bev02] et [WS02]. Considérons un point X de l'image observée: nous supposons que X est une variable aléatoire dont le type de distribution est connu (modèle de bruit). X est l'observation d'une classe particulière, par exemple la classe des pixels de fond ou de forme; nommons k ces classes, et K le nombre de ces classes: soit $k \in \mathbb{N}_K^*$. Comme k dépend de l'observation X, k peut être assimilé également à une variable aléatoire. Enfin, les classes peuvent avoir des modèles de bruit différents.

La probabilité *a posteriori* de la classe du pixel (fond ou forme) sachant l'observation [(xxxv)] s'écrit selon la règle de *Bayes*:

$$P(k|X) = \frac{P(X|k) \cdot P(k)}{P(X)}$$

Concrètement cela signifie que, connaissant sa classe, si on assigne à X un modèle de distribution (modèle *a priori*), nous sommes capable de passer à une distribution *a posteriori*. En général le modèle *a priori* dépend d'un jeu de paramètres, que nous noterons Θ.

Revenons à présent au modèle gaussien. Une distribution gaussienne est donnée par sa moyenne μ et son écart type σ: $\Theta = (\mu, \sigma)$. Si toutes les classes ont une distribution gaussienne, chaque classe possède son jeu de paramètres de type $\Theta_k = (\mu_k, \sigma_k)$. La densité de probabilité conditionnelle à une classe k est donc:

$$f(x|k, \Theta_k) = \frac{1}{\sqrt{2\pi \cdot \sigma_k^2}} \cdot e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$$

[(xxxv)] probabilité *a posteriori* : probabilité déterminée à partir de l'observation

La règle de Bayes permet de passer de la distribution *a priori* de la classe k à sa distribution *a posteriori*. Choisir la classe k parmi l'ensemble des classes est la démarche *d'inférence statistique* par rapport à nos connaissances et à nos observations. Il existe plusieurs méthodes d'inférence, la plus connue étant le maximum de vraisemblance qui choisit la classe qui présente une distribution de probabilité similaire à la réalisation *a posteriori*.

Pour pouvoir appliquer la règle de Bayes, il faut connaître $P(X)$ et les $P(k)$. $P(X)$ est le même pour toutes les classes, il n'intervient donc pas en première approximation pour la décision. En ce qui concerne les $P(k)$, c'est à dire la probabilité qu'une classe se réalise - par exemple rapport entre le nombre de pixels de la classe k et le nombre total de pixels, sur l'ensemble de l'observation - ils sont soit connus à l'avance, soit déterminés par l'observation.

L'*apprentissage* s'effectue dans la détermination des paramètres Θ du modèle, la moyenne et la variance dans le cas gaussien. Ce type d'apprentissage est également qualifié de *génératif*, puisque nous connaissons le modèle et en estimons ses paramètres.

Techniques d'appariement de bloc et de flot optique

Les deux méthodes suivantes sont données à titre indicatif mais ne seront pas développées par la suite.

Appariement de bloc L'appariement de bloc [(xxxvi)] consiste à découper l'image en blocs de tailles égales et d'analyser une séquence d'image. Les blocs sont appariés entre deux trames successives en générale selon une mesure de corrélation telle que le *PSNR*[(xxxvii)]. La recherche pour l'appariement est souvent limitée à une fenêtre de recherche maximale autour du bloc. Les vecteurs de déplacement entre trames successive et blocs appareillés sont ensuite estimés.

Flot optique Les méthodes de flot optique ne travaillent pas sur le bloc mais sur le pixel. La conjecture principale est donnée par l'équation de flot optique, indiquant que seul le mouvement explique la variation de la luminance d'un point dans l'image. La résolution de cette équation au niveau pixel (et uniquement en luminance) ne donne pas de résultat satisfaisant car le champ de mouvement est bruité. Elle est soit complétée d'une étape de relaxation utilisant l'information dans un voisinage, soit effectuée selon une approche multi-échelle pyramidale (du mouvement le plus grossier sur une petite image, vers le mouvement le plus fin à l'échelle du pixel).

Ces deux méthodes sont assez coûteuses en termes de calcul. Les techniques d'appariement de bloc bénéficient toutefois d'implémentations matérielles efficaces, mais toutes génèrent un champ de mouvement difficile à exploiter avec les outils de MM. On perd en effet beaucoup d'information sur les contours des objets, précisément à cause du découpage de l'image en blocs. Les techniques de flot optique sont par contre très intéressantes, car elles fournissent un champ de mouvement dense et exploitables. Elles ont l'inconvénient d'être lentes et coûteuses en ressources.

4.5.2. Utilisation de la couleur

Les métriques couleur introduites en §4.2.1.1 (page 106) sont utilisables pour la détection d'objets en mouvement et, comme nous allons le voir, apportent certaines améliorations notables par rapport à l'approche classique en luminance. Tous les développements présentés ici sont effectués dans le cadre de caméras fixes.

4.5.2.1. Gradients temporels

Nous avons étudié plusieurs techniques dérivées des gradients temporels dans le domaine de la couleur. Dans la suite nous noterons $\mathcal{I}_g$ l'image de « gradient » temporel, $\mathcal{I}^t$ l'image - couleur - au

[(xxxvi)] de l'anglais « *block matching* »
[(xxxvii)] de l'anglais « *Peak Signal to Noise Ratio* »: rapport signal sur bruit maximal

temps t, d une métrique couleur et Δt_0 le pas de temps. Nous notons également ∇_d le gradient morphologique généralisé selon la métrique d.

Différence entre 2 trames Le gradient le plus simple est formulé par : $\mathcal{I}_g = d(\mathcal{I}^t, \mathcal{I}^{t-\Delta t_0})$. Malheureusement, la métrique couleur n'apporte pas de résultat notable par rapport à une approche en luminance seulement, car le canal de saturation reste souvent proche de 0 (faible pertinence de la couleur). Qui plus est, ces métriques couleur dégradent souvent l'information à cause du bruit présent dans les canaux de chrominance.

Gradient spatio-temporel Nous avons ensuite généré des images $3D$ à l'aide de 3 images $2D$, à des temps consécutifs $t, t-\Delta t_0, t-2\cdot\Delta t_0$. Nous avons calculé le gradient morphologique généralisé sur ces images, à l'aide d'un élément structurant de type *cube*, et nous avons pris pour résultat l'image intermédiaire, correspondante à $t-\Delta t_0$, de manière à avoir un gradient symétrique. Par définition, nous trouvons dans ces images à la fois les contours spatiaux et les contours temporels. Il est donc difficile d'extraire les zones de mouvement à partir de cette opération, il faut en effet supprimer la composante spatiale des gradients, car seule la composante temporelle nous intéresse dans ce problème.

Gradient spatial puis temporel Nous pouvons également nous intéresser aux contours ayant variés dans le temps; pour cela, nous calculons les gradients dans chaque trame dans un premier temps, que nous combinons ensuite pour détecter les contours ayant variés. Pour ce type de gradient, un choix important de gradient spatiaux ainsi que de combinaisons temporelles s'offrent à nous. Nous considérons dans la suite que ∇ est un gradient purement spatial.

Combinaisons temporelles L'idée sous-jacente est de détecter dans un premier temps les contours immobiles par intersection temporelle, puis d'utiliser le résidu avec un gradient spatial. La formulation générale devient:

$$\mathcal{I}_g^t = \nabla\mathcal{I}^{t_i} \setminus \left(\nabla\mathcal{I}^t \cap \nabla\mathcal{I}^{t-\Delta t_0}\right)$$

En prenant $t_i = t$ ou $t_i = t-\Delta t_0$, nous obtenons les contours qui respectivement *apparaissent* et *disparaissent* dans le temps. En pratique, nous utilisons une version symétrisée en prenant l'union des deux combinaisons temporelles (par distributivité de $\cap$ sur $\cup$):

$$\mathcal{I}_g^t = \left(\nabla\mathcal{I}^t \cup \nabla\mathcal{I}^{t-\Delta t_0}\right) \setminus \left(\nabla\mathcal{I}^t \cap \nabla\mathcal{I}^{t-\Delta t_0}\right)$$

Se pose à présent le choix de l'opérateur de gradient spatial ∇.

Gradient spatiaux Nous nous sommes aperçus que, concernant les contours spatiaux, les gradients morphologiques généralisés n'apportent pas de grande différence par rapport à leur homologue en luminance. Cette observation est également vérifiée lorsque l'on épaissit le gradient. Ceci s'explique par le fait que la plupart des métriques utilisées font intervenir la luminance (gradient teinte-luminance pondérée dans HLS- équation 4.6 - ou euclidien dans Lab- équation 4.2). Les métriques homologues en teinte uniquement sont par contre corrompues par du bruit généralement important, ce qui les rend en pratique inutilisables. Les deux observations précédentes tendent à nous faire dire que le canal de luminance possède un poids, dans les gradients l'exploitant, plus important que la chrominance; ainsi la différence entre gradients couleur et luminance n'est pas suffisamment pertinente.

Nous sommes ensuite intéressés aux gradients circulaires d'homogénéité qui, comme nous l'avons précisé en §4.2.2.2 (page 121), possèdent l'avantage d'être calculables sur des voisinages de grande taille sans induire un coût de calcul important. La taille du voisinage a pour but de circonvenir au problème du bruit des canaux de chrominance, trop important pour une exploitation avec les gradients morphologiques généralisés.

4.5.2.2. Apprentissage du fond

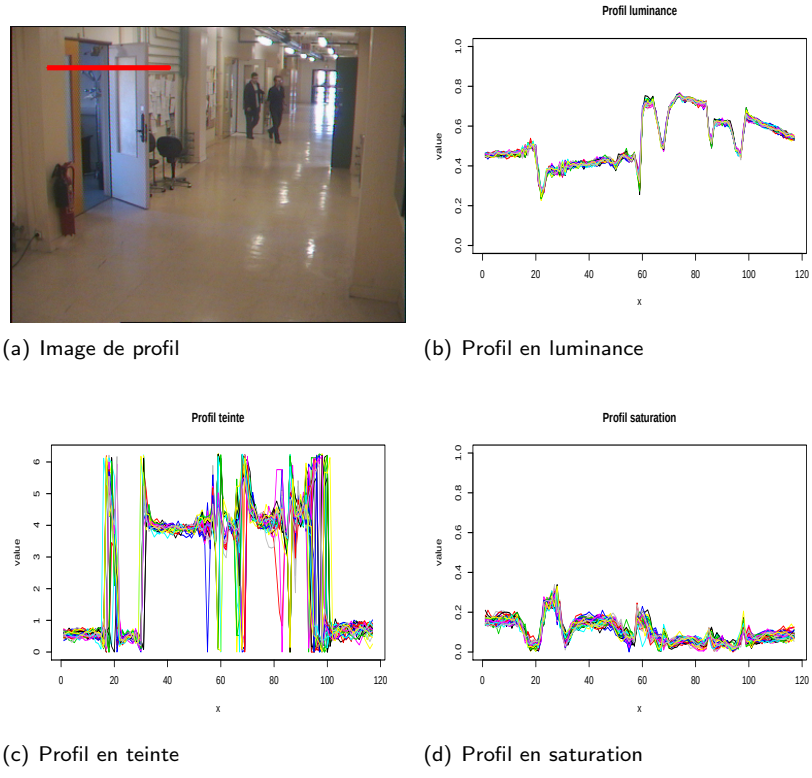
Nous nous plaçons dans le domaine HLS, et nous allons établir une méthode de construction d'image de fond à partir des mesures statistiques sur cet espace (§4.2.2.1 page 119). Nous considérons toujours

un pixel X comme étant la réalisation d'une variable aléatoire dont il faut estimer les paramètres. L'objectif est d'obtenir une image de chrominance moyenne stable. Pour atteindre ce but, nous allons tout d'abord observer dans le temps la teinte considérée au niveau du pixel. Nous essaierons ensuite de stabiliser les résultats en traitant préalablement spatialement la chrominance.

Moyenne temporelle Soit donc X un pixel prenant des valeurs dans le domaine de chrominance de l'espace HLS. Nous avons N observations de X correspondant à N trames successives, et soient $X_k, k \in \mathbb{N}_N^*$ ses réalisations aux trames k. Puisque nous sommes dans le domaine HLS et que nous nous intéressons à la chrominance, nous supposons chaque X comme étant une variable dans le plan complexe $\mathbb{C}$, de module $|X|$ et d'argument $\arg X$:

$$\forall k \in \mathbb{N}_N^*,\ X_k = |X_k|\, e^{\mathbf{i}\cdot \arg X_k}$$

Voyons comment se comporte la moyenne et la dispersion circulaire d'échantillon. La figure 4.49 montre le même profil sur l'image originale au cours du temps sur les trois canaux. Le profil a été choisi de manière à ce qu'il n'y ait pas d'objet créant d'occlusion (c'est donc une zone sans mouvement apparent).



(a) Image de profil

(b) Profil en luminance

(c) Profil en teinte

(d) Profil en saturation

Fig. 4.49.: Profil originaux en teinte, luminance et saturation sur un échantillon vidéo. (b), (c) et (d) : en abscisse la position sur la ligne rouge de (a), en ordonnée les valeur de (a) respectivement en luminance, teinte et saturation. Chaque courbe sur chacune des figure représente une acquisition (image) au temps t. Il y a au total 40 acquisitions.

Selon cette figure, le canal de luminance est très stable ce qui n'est pas une surprise. Le canal de saturation présente un bruit relativement important, mais serait toutefois facilement exploitable. Le fait est que la saturation reste assez faible. Enfin le canal de teinte présente un bruit important (sans tenir compte de la circularité de l'espace). Quand bien même nous serions capables d'avoir une moyenne temporelle satisfaisante, la différence en teinte de l'image moyenne et de l'image courante sera toujours corrompue par un bruit important.

La moyenne temporelle est donnée en chaque point par, pour la version non pondérée par la saturation

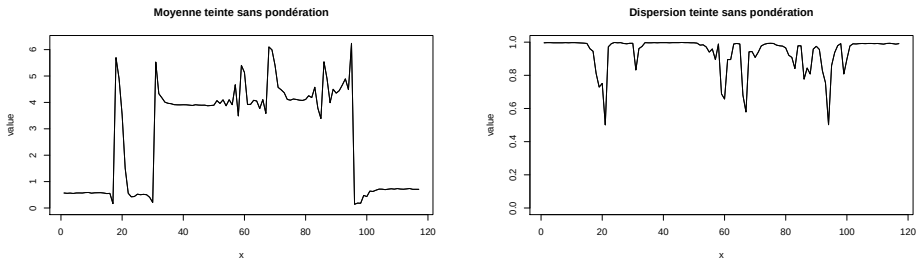
$$\overline{X} = \frac{1}{N}\sum_{k=1}^{N} e^{\mathbf{i}\cdot\arg X_k}$$

et pour la version pondérée par

$$\overline{X}^{w} = \frac{\sum_{k=1}^{N} X_k}{\sum_{k=1}^{N} |X_k|}$$

La teinte moyenne par intégration temporelle est donnée en figure 4.50. Elle est relativement cohérente à l'observation de la figure 4.49(c), mais l'observation de la dispersion (figure 4.50(b)) indique une variance élevée de cette moyenne.

En pratique, nous n'avons pas été satisfaits par l'utilisation seule de l'intégration temporelle. En effet, ces quelques observations étant sur des zones sans mouvement apparent, l'intégration temporelle doit donner une moyenne stable (ce qui se traduit par dispersion aussi proche de 1 que possible). Aussi nous nous sommes intéressés à un traitement préalable de la teinte par moyenne spatiale. Nous allons voir à présent ces résultats.



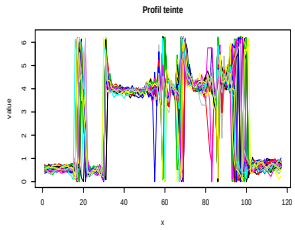
(a) Profil en teinte après moyenne temporelle - $\arg \overline{X}$ (b) Profil en dispersion après moyenne temporelle - $|\overline{X}|$

Fig. 4.50.: Teinte moyenne de la figure 4.49(c) au cours du temps. (a) teinte moyenne après intégration temporelle. (b) dispersion associée. Nous constatons une dispersion assez faible en de nombreux points (variance élevée). La version pondérée par la saturation est similaire.

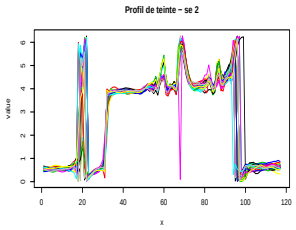
Prétraitrement par moyenne Nous nous intéressons ici à un simple prétraitement par moyenne spatiale avant toute moyenne temporelle. Nous en déduirons une qualité de moyenne temporelle plus importante que lorsque calculée sur chaque pixel de manière indépendante. Ce traitement modifie naturellement la variable aléatoire d'intérêt: soit donc X un pixel et $\mathcal{V}(X)$ ses voisins selon un élément structurant fixé (et n'ayant pas de dimension temporelle). Soit $\mu_{\mathcal{V}}(X)$ la nouvelle variable aléatoire définie comme étant la moyenne de X sur le voisinage $\mathcal{V}$.

De manière similaire à la figure 4.49, la figure 4.51 illustre la moyenne spatiale du même profil au cours du temps. Nous remarquons que par intégration spatiale, la teinte moyenne $\Theta_{\mathcal{V}}(X)$ se comporte relativement de manière stable au cours du temps. Même un voisinage petit améliore les résultats. Sur cette même figure, nous avons également tracé la dispersion à titre indicatif: elle est significative des contours spatiaux dans le domaine de chrominance.

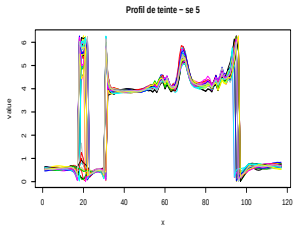
Observons à présent l'intégration temporelle de $\mu_{\mathcal{V}}(X)$. La figure 4.52 montre l'intégration temporelle de la variable $\mu_{\mathcal{V}}(X)$ pour différentes tailles du voisinage $\mathcal{V}$. Les résultats sont assez éloquents puisque la dispersion temporelle de cette nouvelle variable est assez faible. En effet, nous constatons un point de fluctuations relativement important (sur la partie gauche des courbes de dispersion) pour l'élément structurant unitaire. Cette zone correspond en fait à la « vitre » de la porte, laissant transparaître le fond légèrement bleu entre deux zones de teinte assez similaires (le mur et le cadre de la porte). D'après cette observation, l'intégration spatiale réagit mal aux hautes fréquences spatiales (transitions rapides)



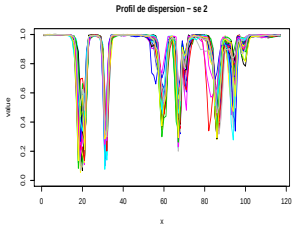
(a) Teinte originale



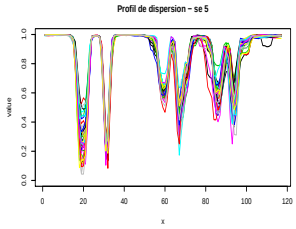
(b) Teinte localement moyennée par un ES de taille unitaire - $\mu_{\mathcal{V}_1}(X)$



(c) Teinte localement moyennée par un ES de taille 5 - $\mu_{\mathcal{V}_5}(X)$



(d) Dispersion locale pour un moyenne locale donnée par un ES de taille unitaire



(e) Dispersion locale pour un moyenne locale donnée par un ES de taille 5

Fig. 4.51.: Moyenne spatiale en teinte (sans pondération) au cours du temps. (a) teinte originale. (b) et (d) moyenne et variance spatiaux pour un lissage avec un élément structurant de taille unitaire. (c) et (e) moyenne et variance spatiaux pour un lissage avec un élément structurant de taille 5. La boule unité apporte une stabilité non négligeable, améliorée davantage avec un élément structurant de taille 5.

lorsque l'élément structurant utilisé est d'une taille comparable à la longueur d'onde (ici, l'élément structurant est de largeur 3 et la vitre de la porte est visible sur 5 pixels), ce qui est assez naturel. En agrandissant l'élément structurant, nous améliorons cette dispersion, mais nous perdons de la résolution spatiale (les contours deviennent plus « flous » et une incertitude s'installe sur leur position).

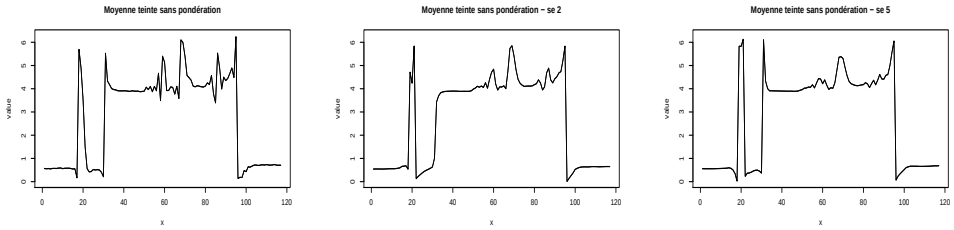
Concernant la comparaison entre la moyenne spatiale avec et sans pondération par la saturation, nous avons observé des contours plus saillants pour la version pondérée (dans les espaces images). Les résultats des profils sont sensiblement équivalents. Par ailleurs, nous n'avons pas réussi à rendre la pondération, d'un point de vu temporel, pertinente.

Voyons à présent quel type d'utilisation nous pouvons avoir de ces développements.

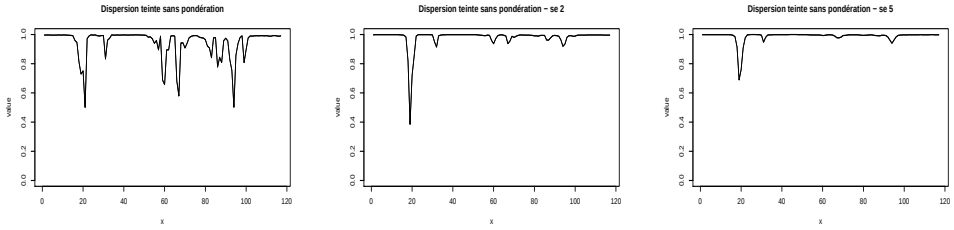
Utilisation Nous avons déjà très succinctement mentionné l'apport de la couleur dans la détection de zone de mouvement (§4.2.1.1 page 109). Rappelons que nous avons changé de variable aléatoire, et que nous ne nous intéressons plus uniquement au pixel, mais également également à ses voisins (variable $\mu_{\mathcal{V}}(X)$). Nous devons donc comparer cette variable avec sa moyenne, ce qui implique un traitement de toutes les trames avant comparaison. Par ailleurs, comme nous l'avons mentionné pendant les développements des statistiques circulaires (§4.12 page 120), la variance circulaire ς et la dispersion sont liés simplement par $\varsigma = 1 - R_0$, où R_0 est la dispersion calculée ici par intégration temporelle.

Dans le cas classique en luminance, si l'hypothèse gaussienne pour la variable aléatoire « pixel » est énoncée, la variance de la variable permet généralement de « binariser » l'image en deux classes de *fond* et de *forme*. Pour cela, une constante α est choisie, et les pixels de fond sont ceux vérifiant la condition suivante, avec σ_X l'écart type de X et $\overline{X}$ sa moyenne:

$$\mathcal{F} = \left\{ X \in \mathbf{E} \,/\, \left| X - \overline{X} \right| < \alpha \cdot \sigma_X \right\}$$



(a) Teinte par intégration temporelle - $\arg \overline{X}$ (b) Teinte prétraitée par intégration temporelle - $\arg \overline{\mu_{\mathcal{V}_2}}(X)$ (c) Teinte prétraitée par intégration temporelle - $\arg \overline{\mu_{\mathcal{V}_5}}(X)$



(d) Dispersion par intégration temporelle - $\left|\overline{X}\right|$ (e) Dispersion par intégration temporelle - $\left|\overline{\mu_{\mathcal{V}_2}}(X)\right|$ (f) Dispersion par intégration temporelle - $\left|\overline{\mu_{\mathcal{V}_5}}(X)\right|$

Fig. 4.52.: Intégration temporelle de la teinte prétraitée par moyenne locale. La première ligne présente les résultats en teinte, la deuxième présente la dispersion de l'intégration temporelle. Cette figure oppose l'intégration temporelle sans prétraitement (résultat du paragraphe précédent) à celle avec prétraitement par moyenne locale. Nous remarquons d'une part que les résultats sont cohérents avec ceux de la figure 4.51, d'autre part que la dispersion reste faible.

Nous appliquons donc le même schéma pour la détection en teinte, en utilisant l'opérateur de distance angulaire $\angle : (a, b) \mapsto |a \angle b|$ au lieu de la valeur absolue de la différence. Seul l'argument de X est considéré, ce qui nous donne (avec ς_X^2 la variance circulaire de X):

$$\mathcal{F} = \{X \in \mathbf{E} / \left|\arg(X) \angle \arg\left(\overline{\mu_{\mathcal{V}}}(X)\right)\right| < \alpha \cdot \varsigma_X\}$$

Comparons les résultats pour la luminance et la chrominance selon ce procédé. Quelques résultats pour une séquence possédant une faible chrominance sont donnés sur la figure 4.53. Nous remarquons de bonnes détections lorsque le contraste avec le fond est important, cependant la différence montre des « trous » dans le pull et des ombres portées importantes. Enfin lorsque le second personnage se trouve sur les escaliers, le bruit des ombres est assez important. Il disparaît totalement dans cette même zone pour $\alpha = 2$; en effet, la différence en luminance est très faible.

Les détections en chrominance sont assez bonnes malgré la faible quantité de saturation au long de la séquence. Nous avons pris $\alpha = 1$. Les problèmes d'ombre portée disparaissent. Le pull et le pantalon du premier personnage disparaissent partiellement lorsque celui-ci passe devant la rambarde: le pantalon et le pull sont tous deux achromatiques, et la comparaison angulaire n'est pas efficace. Ceci est partiellement résolu en considérant un voisinage plus grand. Le pantalon du deuxième personnage disparaît dans la zone des escaliers à cause de la similitude en teinte.

L'augmentation de la taille de l'élément structurant, dans le calcul de $\overline{\mu_{\mathcal{V}}}(X)$, conduit généralement à de meilleures détection mais ajoute un peu de bruit. Les détections pour les tailles unitaire et 5 sont toutefois très similaires, sauf éventuellement dans quelques zones achromatiques.

Conclusion Dans cette section, nous avons vu comment adapter les outils développés concernant l'espace circulaire en teinte pour la création d'une image de fond. Dans un premier temps, nous avons observé la faible stabilité du canal (que nous avons également vérifié sur de nombreuses séquences), ce qui nous a empêché de formuler une moyenne temporelle. Nous avons ensuite contourné ce problème

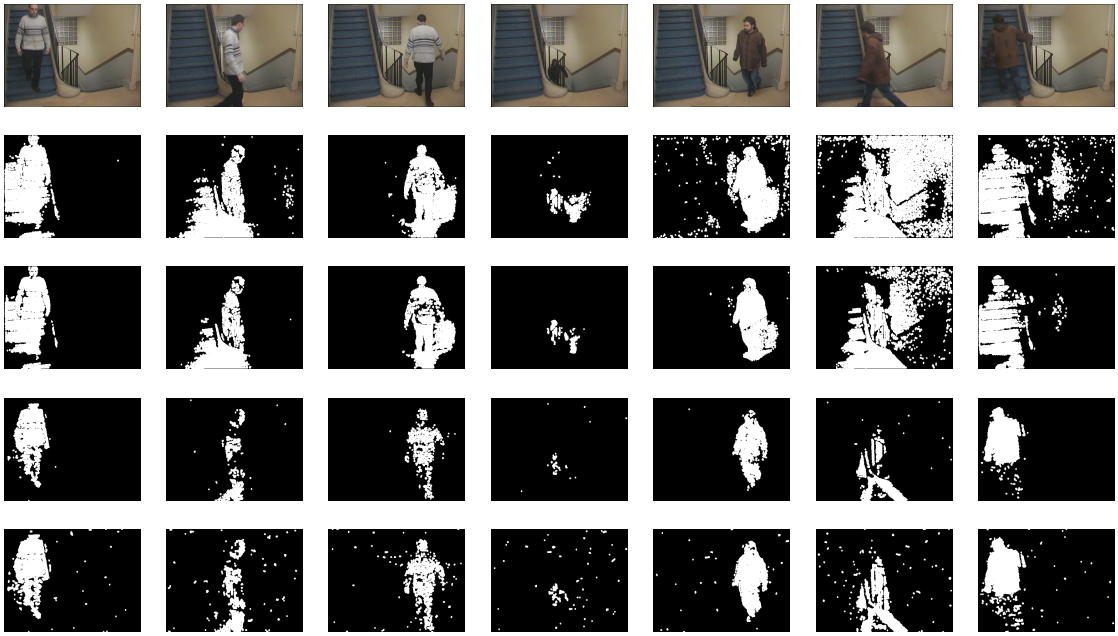


Fig. 4.53.: Comparaison des approches en luminance et chrominance. La première ligne montre quelques extraits de la séquence. Les deuxième et troisième lignes présentent la détection en luminance en prenant respectivement $\alpha = 1$ et $\alpha = 2$, puis en appliquant une petite ouverture pour supprimer les fausses détections. La quatrième ligne est la détection en chrominance en prenant $\alpha = 1$ et une taille de voisinage unitaire. Enfin la dernière ligne présente les mêmes résultats sur un voisinage de taille 5.

en effectuant préalablement une moyenne locale dans le domaine de teinte. Ces images ont montré une meilleure stabilité de la teinte, ce qui se répercute sur la dispersion (après intégration temporelle). Nous avons ensuite vu comment déduire une détection de type fond-forme en utilisant conjointement la teinte moyenne et sa dispersion, de manière analogue au cas gaussien pour la luminance. Nous avons montré la pertinence de ces résultats sur une séquence « type » sur laquelle la chrominance est assez peu saturée. Enfin, par comparaison avec l'approche en luminance seule, nous avons démontré l'intérêt d'utiliser la moyenne en chrominance pour la résolution d'un certain nombre de problème, comme ceux des ombres portées et du faible contraste en luminance.

4.6 Conclusion

Résumé et synthèse

L'objectif de cette partie était de transposer, selon différentes méthodes, des schémas de calcul connus du domaine « scalaire » au traitement vectoriel. Nous avons nommé ce traitement « multispectral ». Pour cela, nous avons mis en parallèle trois méthodes dans la définition d'opérateurs simples (ie. des gradients dits « morphologiques »). Les trois méthodes mentionnées sont: l'approche par des *distances* couleur, par des *statistiques* de voisinage et enfin *algébrique* à travers l'utilisation de relations d'ordre lexicographique.

Nous avons développé davantage l'approche algébrique en montrant comment les relations d'ordre lexicographique pouvaient être utilisées, de manière informatique, dans la définition d'opérateurs morphologiques de plus haut niveau. Enfin nous avons utilisé certains de nos développements dans deux applications concrètes en recherche d'invariant de peau et en segmentation pour la vidéosurveillance. Nous reprenons dans la suite les points qui nous semblent importants dans ce chapitre.

• **Approches couleur** La conclusion majeure est qu'une fois de plus, le traitement de la couleur pose des problèmes difficiles à résoudre. Si la transposition des concepts intervenant dans les opérateurs simples (gradients) est une tâche facile (cf. §4.2.1), cette même transposition est souvent impossible pour des opérateurs plus complexes ne trouvant pas de formulation selon un point de vue d'*accroissement* [(xxxviii)].

L'approche par distance nous semblait au premier abord avantageuse puisque bénéficiant d'une justification colorimétrique. Cependant, les combinaisons entre les points d'un voisinage sont plus nombreuses par cette approche 2 à 2, ce qui conduit à une complexité algorithmique importante. Enfin, et nous ne l'avons pas évoqué, l'augmentation des dimensions de l'espace de travail ou l'utilisation de capteurs multispectraux rend le choix de la fonction distance largement non-trivial.

Selon ces dernières remarques, l'approche statistique semble plus simple à manipuler puisque la notion « d'accroissement » est remplacée par celle de « similitude statistique ». L'absence de contour est définie comme étant un échantillon de points de même distribution statistique. Inversement, un échantillon ne satisfaisant pas une hypothèse de distribution définit un contour. Pour que cette notion puisse être utilisée correctement, les points de l'échantillon doivent appartenir à un même voisinage, ce qui fait le lien avec le traitement d'image. Nous avons vu qu'une telle solution permettait de définir des opérateurs simples sur des voisinages de taille importante, contrairement à l'approche 2 à 2 des distances couleur. Par ailleurs, la fonction de similitude nous semble plus simple à manipuler.

Ces deux approches sont à confronter avec celle utilisant les notions algébriques sur les espaces multispectraux. Pour avoir des notions algébriquement équivalentes dans le cas multispectral, nous avons utilisé les relations d'ordre lexicographique. Outre l'utilisation de ce type d'ordre, nous avons retenu particulièrement l'aisance avec laquelle une telle approche pouvait être portée dans le domaine couleur. En effet, la transposition algébrique permet d'utiliser tous les développements de la MM, qu'ils soient au niveau de la définition des opérateurs ou de leur méthode de calcul algorithmique. Pour cela, le cadre informatique introduit au chapitre 2 était particulièrement bien adapté.

Il existe toutefois deux critiques importantes. Bien que cela ne remette pas en question l'utilisation d'outils mathématiques préservant la structure algébrique de treillis (les relations d'ordre), le choix des relations d'ordre lexicographique est largement discutable tant au niveau du principe lexicographique que des résultats obtenus. Nous avons vu un certain nombre d'opérateurs (gradient en §4.2.1.5, reconstruction morphologique en §4.3, extrema, granulométries), mais aucun ne nous a réellement convaincu de l'apport du traitement lexicographique. L'inconvénient principal de ce type de relation d'ordre est qu'il définit implicitement une hiérarchie entre les canaux, hiérarchie qui revêt trop d'importance par

[(xxxviii)] les opérateurs proposant une telle formulation sont malheureusement rares, car la plupart font intervenir également une notion d'*origine* des valeurs.

rapport à nos objectifs (où nous souhaitions utiliser conjointement plusieurs canaux). En effet, le canal placé en premier fait office de « filtre » pour les canaux suivants, et l'approche perd son sens sur des images naturelles.

Comme nous l'avons mentionné, l'interprétation des résultats est également difficile. À l'exception d'opérateurs très « synthétiques » telle que les granulométries §4.3.2, pour lesquels l'interprétation suit la même démarche que pour le cas scalaire, les résultats (peu nombreux) des opérateurs lexicographiques ne sont exploitables que sur des cas très simples. L'exemple avec lequel nous avons illustré l'opérateur de reconstruction montrait précisément un cas d'utilisation simple, où chaque canal pris séparément présentait des dégradations alors que les canaux pris ensemble permettent de reconstruire un objet de l'image.

• **Statistiques circulaires dans HLS** S'il existe effectivement des approches statistiques au traitement de la couleur (voir par exemple Lerallut [Ler06]) ou l'utilisation de statistiques circulaires (travaux de Hanbury [Han02a]), nos travaux restent toutefois originaux. L'utilisation de la moyenne circulaire pour le traitement de l'espace HLS à des moments supérieurs à 1 n'a pas été, à notre connaissance, utilisée dans la littérature. Par ailleurs, le facteur pondérant de la saturation n'a pas été non plus envisagé. La dispersion, qui est également un facteur très important dans la description statistique de variables aléatoires - au même titre que la variance pour les statistiques « classiques »- n'a, nous semble-t-il, jamais été envisagée.

Grâce à l'ensemble de ces définitions, nous avons été capable de définir des opérations simples et des descripteurs couleur. Parmi les opérateurs simples, nous avons défini les « gradients circulaires d'homogénéité » (pondérés ou non par la saturation). Nous avons souligné le fait que ces gradients réagissaient bien aux zones de transition pour des images de nature variée, et qu'ils étaient peu coûteux en termes de calcul. Parmi les opérations complexes, nous avons utilisé ces descripteurs statistiques dans l'analyse quantitative de la peau sous différents illuminants. Nous avons également mis en valeur ces apports dans le traitement de vidéos issues de caméras dont l'information couleur est plus ou moins dégradée. Pour ces derniers, nous avons vu une manière de stabiliser les informations couleurs, grâce à ces statistiques utilisées dans le temps.

• **Analyse quantitative de la peau** L'analyse quantitative de la peau que nous avons proposée en §4.4 montre également une approche originale, et ce pour plusieurs raisons. Comme nous l'avons mentionné précédemment, elle tire profit des développements sur les statistiques circulaires, ce qui à notre connaissance est inexistant dans la littérature. Cependant, l'originalité majeure réside dans l'approche exhaustive « *multi-illuminant* » pour le choix de l'espace et des invariants.

Nous avons volontairement restreint notre analyse à un certain nombre d'espaces de représentation, sur lesquels nous avons émis des hypothèses très variées (primaires RGB et illuminant de départ). Malgré cette restriction, l'analyse n'a pas perdu de son intérêt dans la mesure où ces espaces sont très représentatifs de ceux utilisés en traitement d'image, et sont des transformations à la fois linéaires et non linéaires du cube RGB de départ.

Bien que le changement de blanc de référence soit une méthode très classique en colorimétrie, son utilisation pour la justification des choix de l'espace et des invariants représente une innovation qui nous semble importante. En effet, la justification est à la fois analytique et quantitative, contrairement aux approches connues qui sont soit issues de méthodes de classification plus ou moins complexes, soit des transformations simples du cube RGB de départ (transformations linéaires essentiellement, voir [MSL01]). Rappelons que l'inconvénient de la première méthode réside dans sa très grande dépendance par rapport aux données d'apprentissage et dans la difficulté à transmettre une « configuration » sans imposer aux utilisateurs le calcul complet des données d'apprentissage.

Enfin, partant d'hypothèses nombreuses, les résultats sur de nombreuses bases de peaux montrent des résultats très intéressants pour le problème mentionné. Évidemment, la détection proposée souffre des inconvénients des traitements proches du pixel: il est impossible de savoir si les pixels détectés sont effectivement peau ou non-peau. On aurait pu pour cela analyser également les pixels non-peau,

mais cette approche aurait été nécessairement biaisée par la base de données[xxxix]. Pour contourner ces problèmes de détection, il est nécessaire de passer à des méthodes de plus haut niveau (détection des pixels de peau et des différents éléments du visage, etc.), ce qui était hors du cadre de ce chapitre.

Pour toutes ces raisons, et bien que la détection des zones de peau soit une discipline largement étudiée en analyse d'image, l'approche que nous avons proposée apporte des éléments de méthodologie, d'analyse et de précision pertinents.

Perspectives

Après de tels développements et remarques, les perspectives sont heureusement très nombreuses. Ce qui nous semble très prometteur est l'approche algébrique. Malgré les résultats et la conclusion dubitative à l'égard des relations d'ordre lexicographique, il serait particulièrement intéressant de continuer dans cette voie par l'utilisation d'autres critères d'ordre. Pour cela, il nous semble nécessaire d'étudier beaucoup plus finement les notions algébriques d'ordre intervenant dans les opérateurs morphologiques, et les critères minimaux auxquels doivent répondre les relations d'ordre. La propriété « *stricte et faible* » pour la manipulation algorithmique des opérateurs lexicographiques, est un exemple de critère. Les algèbres utilisant la notion de « flou » [DH02, Zim91, Nac02, BDCK03] nous paraissent en ceci particulièrement attrayants.

[xxxix] comme expliqué en §4.4.1.3 page 142, les pixels non-peau sont potentiellement partout dans l'espace couleur.

CHAPITRE 5

Inondation

« Humidité : cause de toutes les maladies. »
Dictionnaire des idées reçues, G. Flaubert

La ligne de partage des eaux, ou *lpe*, ou watershed, est une transformation largement utilisée en Morphologie Mathématique. Elle a connu un franc succès dès son introduction par Lantuéjoul & Beucher [BL79]: elle propose en effet une méthode *automatique* de partition de l'image. Des variations de l'algorithme initial ont ensuite permis un contrôle plus fin des partitions, offrant ainsi un certain nombre de moyens pour contourner la sur-segmentation: l'ajout de marqueurs permettant d'avoir une partition semi-supervisée, la sélection des minima par des critères de dynamique [Gri91], les cascades [Beu91], les graphes [MI96], les fusions sélectives de régions [AL03], ... sont quelques exemples. Parallèlement, un cadre mathématique rigoureux s'est établi autour de l'algorithme initial [Naj94, Mey94, NWvdB03]. La première partie de ce chapitre sera consacrée à un rappel algorithmique et théorique de la *lpe*.

Sur le plan informatique, L. Vincent & P. Soille [VS91] ont proposé un algorithme de calcul rapide de la *lpe*, grâce à l'utilisation de files d'attente simples triées. Cet algorithme a eu un impact important sur l'utilisation même de la *lpe*; il a catalysé son utilisation aux disciplines diverses de traitement d'image, au delà de la Morphologie Mathématique. Meyer [Mey91] (et brevet [Mey95]) a ensuite proposé l'utilisation de files d'attente hiérarchiques. Toutefois, l'implémentation de ce dernier algorithme est en vérité assez délicate: certains biais apparaissent lors de la propagation si cette dernière n'est pas effectuée rigoureusement. Dans la seconde partie de ce chapitre, nous ferons la lumière sur ces biais, et proposerons les corrections adéquates. Par ailleurs, grâce à l'ensemble des outils introduits au chapitre 2, nous utiliserons ce même algorithme sur des reliefs de type réel et vectoriel, ce dernier étant possible grâce à l'utilisation d'ordres lexicographiques. Enfin, n'étant pas lié à la dimension des images, nous montrerons également des résultats de segmentation sur des images en dimension quatre.

Il existe pour certaines applications une information *a priori* caractérisant une ou plusieurs régions: une couleur, une forme, etc.. L'algorithme classique de *lpe* ne permet pas d'inclure cette information pendant la création de la partition. La troisième partie de ce chapitre adresse ce problème. Nous proposerons un algorithme permettant de contraindre la propagation de chaque région selon une mesure établie sur l'ensemble des points de la région.

L'utilisation de la *lpe* est parfois infructueuse et ne produit pas les résultats attendus: c'est par exemple le cas lorsqu'elle est appliquée à des images présentant des « imperfections » minimes, dégradant le relief en certains points, mais impactant fortement la partition finale. Quelques approches nouvelles ont été réalisées pour prévenir ce problème, qui appliquent des forces locales au fluide. La

quatrième partie de ce chapitre sera consacrée à un algorithme d'inondation localement contrainte.

5.1 Ligne de partage des eaux

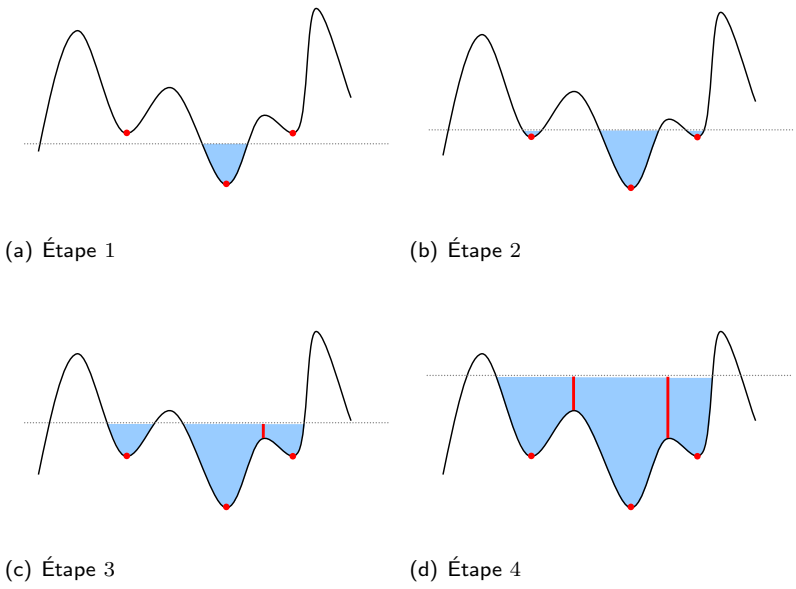
L'algorithme de ligne de partage des eaux est un outil de segmentation et partitionnement puissant. De nombreuses personnes se sont intéressées à son étude et son amélioration, que cela soit dans l'implémentation, la mise en oeuvre matérielle [KLGP95], l'accélération ou la parallélisation.

Après l'étude théorique de la ligne de partage des eaux, nous en énoncerons certaines propriétés et nous décrirons en détail l'algorithme basé sur des files d'attente. Nous étendrons ensuite cet algorithme sur des images de type flottant par l'introduction de files d'attente hiérarchiques génériques.

Le lecteur désireux de se lancer dans une revue des méthodes et des interprétations possibles de la ligne de partage des eaux pourra se référer par exemple à l'article de Roerdink & Meijster [RM00]: les différentes méthodes et leurs algorithmes y sont énoncés. On y trouvera également des discussions sur des implémentations matérielles d'algorithmes dégradés.

5.1.1. Avant propos

Avant d'aller plus loin dans les développements théoriques et algorithmiques, il serait bon pour le lecteur d'avoir une explication générale sur la ligne de partage des eaux. La *lpe* est souvent expliquée à partir de l'inondation d'un relief topographique. La figure 5.1 reprend cette explication.



(a) Étape 1 (b) Étape 2

(c) Étape 3 (d) Étape 4

Fig. 5.1.: LPE - Étapes d'inondation d'un relief. Les minima locaux du relief (points rouges) représentent les sources d'inondation. La ligne grise est la hauteur d'inondation, et les lignes verticales rouges les barrages entre les sources différentes.

Les minima locaux de l'image, les fonds de vallées du relief, représentent les sources d'inondation. Le relief est plongé dans de l'eau s'infiltrant à partir des sources, c'est à dire des minima locaux. L'inondation se produit naturellement à partir des minima les plus bas. La hauteur de l'inondation est progressive, et lorsque les fluides de deux sources différentes se rejoignent, un barrage est érigé au point de contact: l'ensemble de ces barrages forme la *lpe*. On appelle *bassin versant* d'un minimum l'ensemble des points du relief appartenant au lac créé par ce minimum. Puisque deux lacs ne se mélangent pas,

nous obtenons un ensemble de bassins versants complètement disjoints deux à deux, c'est à dire une partition du relief.

L'idée de l'application du watershed sur une image est précisément de créer une partition de cette image en régions: l'image à teinte de gris est considérée comme un relief topographique. Les minima locaux de l'image ont la même signification que celle du relief exposé. Les bassins versants des minima représentent alors chaque région de la partition.

5.1.2. Cadre théorique

L'algorithme présenté dans la section précédente semble extrêmement simple. Cependant la modélisation mathématique du processus d'inondation est loin d'être évidente. Ce problème provient en partie du fait que la *lpe* est définie de manière algorithmique. Les avantages d'une modélisation mathématique sont assez nombreux. Cela a par exemple conduit Najman [Naj94] à une équivalence entre le calcul de la *lpe* et la solution de l'équation eikonale (cf. [Naj94, page 82]) qui est : « trouver la fonction f telle que $\|f\| = g$ ». Le cadre théorique présenté ici nous intéresse particulièrement puisqu'il nous permettra par la suite de modifier la construction originale des bassins versants de la *lpe*, de manière à inclure des données (forces) extérieures au relief inondé.

Nous allons commencer par faire le lien entre le SKIZ géodésique et la partition de Voronoï dans le cas binaire. Nous étudierons ensuite le cas numérique, et nous donnerons le formalisme pour l'expression de la *lpe*. Enfin grâce à ce formalisme, nous énoncerons les méthodes actuelles de modification de la *lpe*. Ces développements s'appuient sur les travaux de Meyer [Mey94], Najman [Naj94], N'Guyen & al. [NWvdB03] et Roerdink & Meijster [RM00].

5.1.2.1. Distance et SKIZ géodésiques

La *géodésie* est une notion *topographique*, et dont l'utilisation en MM est due aux travaux de Lantuéjoul & Beucher [LB81] et Lantuéjoul & Maisonneuve [LM84]. La définition d'arc est préalable à celle de distance géodésique.

Soit $(a, b) \in \mathbf{E}^2$ un couple de points: un *chemin* ou encore *arc* entre a et b dans $\mathbf{E}$ est une application *continue* et *dérivable* $\sigma_{a \rightharpoonup b} : [0, 1] \rightarrow \mathbf{E}$ telle que:

$$\sigma(0) = a,\ \sigma(1) = b$$

Si l'on considère des arcs orientés, a est alors appelé l'origine et b l'extrémité de l'arc. Nous ne considérerons que des arcs non-orientés, et noterons $\sigma_{a \rightleftharpoons b}$ un arc non orienté entre a et b.

Soit à présent un sous-ensemble $\mathbf{E}' \subset \mathbf{E}$, et $(a, b) \in \mathbf{E}'^2$. La notion de géodésie intervient lorsque l'on cherche les arcs reliant a et b, de manière à ce que ceux-ci soient totalement inclus dans $\mathbf{E}'$. Il est cependant plus aisé de manipuler la géodésie à travers la *distance géodésique*.

La longueur d'un arc σ est définie par la quantité suivante:

$$l(\sigma) = \int_0^1 \|\sigma'(s)\|\, ds$$

Cette longueur correspond à la somme des longueurs élémentaires de l'arc. Soit l'ensemble des arcs entre a et b dans $\mathbf{E}'$

$$\Sigma_{a \rightleftharpoons b} = \left\{\sigma_{a \rightleftharpoons b}, (a, b) \in \mathbf{E}'^2\right\}$$

La distance géodésique $d_g(a, b)$ entre a et b est la longueur minimale sur l'ensemble des arcs entre a et b:

$$d_g(a, b) = \begin{cases} \bigwedge_{\sigma \in \Sigma_{a \rightleftharpoons b}} l(\sigma) & \text{si } \Sigma_{a \rightleftharpoons b} \neq \varnothing \\ +\infty & \text{si } \Sigma_{a \rightleftharpoons b} = \varnothing \end{cases}$$

Le diagramme de Voronoï a été défini au chapitre 3 (cf. définition 3.3 page 50). Le *squelette par zones d'influence géodésique* (SKIZ géodésique) de $\mathbf{E}$ est simplement le diagramme de Voronoï de $\mathbf{E}$

selon la distance géodésique. Enfin, notons que la discrétisation de la distance ne pose pas de problème particulier. On exprime souvent un chemin dans le cas discret ($\mathbf{E}'$ est un ensemble dénombrable de points) comme la donnée d'un ensemble $\{p_i, i \in \mathbb{N}_n^*\}$ de n points de $\mathbf{E}'$, telle que $p_1 = a$, $p_n = b$ et que $\forall i \in \mathbb{N}_{n-1}^*, p_{i+1} \in \mathcal{N}(p_i)$, le voisinage étant pris ici dans son sens discret. La distance géodésique discrète entre deux points est alors, comme son homologue continu, la borne inférieure des longueurs des arcs discrets entre ces deux points.

Les développements présentés ici se placent dans un cadre binaire, où les fonctions d'étude et leur support sont confondus. Dans ce cadre, la *lpe* et le SKIZ géodésique sont des notions équivalentes. Or l'explication initiale de la *lpe* est l'inondation d'un relief assimilé à une fonction f, et de ce fait une définition prenant en compte la valuation de f doit être donnée. Par ailleurs, ce point distingue la *lpe* des algorithmes classiques de calcul de Voronoï.

5.1.2.2. Formulation métrique par distance topographique

La distance topographique [Mey94] prend précisément en compte les variations du relief. Soit f la fonction de relief à niveau de gris : $f : \mathbf{E} \to \mathbb{R}$. f est supposée de classe $\mathcal{C}^2$, c'est à dire continue, et à dérivées première et seconde continues. La distance topographique T_f entre deux points $(a, b) \in \mathbf{E}^2$ est exprimée par:

$$T_f(a,b) = \inf_{\sigma \in \Sigma_{a \rightleftharpoons b}} \int_0^1 \|\nabla f(\sigma(s))\| \, ds \tag{5.1}$$

Najman [Naj94, pages 68, 69 et 73] montre d'une part l'existence d'un arc minimisant la valeur sous l'intégrale, d'autre part que la *lpe* est effectivement un squelette par zones d'influence. Pour cela, il considère l'ensemble $\{m_i\}_i$ des minima de f, et définit les bassins versants par:

$$BV_i = \{x \,/\, T_f(x, m_i) + f(m_i) < T_f(x, m_j) + f(m_j), j \neq i\} \tag{5.2}$$

La *lpe* est ensuite donnée par les points n'appartenant à aucun bassin versant (ou par l'union des intersections des bassins versants pris deux à deux, si l'inégalité de l'équation 5.2 est large, comme dans [Naj94, page 73]).

Discrétisation La discrétisation de l'équation 5.1 n'est pas triviale. À notre connaissance, la première formulation est due à Meyer [Mey94]. Elle se base sur la distance de chanfrein d_c [(i)] [**?**] qui vise à approximer la boule euclidienne dans un voisinage discret, et de manière à réduire les biais lors de l'augmentation de la taille du voisinage. Soit un point $x \in \mathbf{E}$, $U(x)$ l'ensemble des points tels que $\forall v \in U(x), f(v) < f(x)$ et contenus dans le masque de chanfrein. La pente basse LS [(ii)] dans $U(x)$ est définie comme:

$$LS : x \mapsto \begin{cases} \vee_{v \in U(x)} \frac{f(x)-f(v)}{d_c(x,v)} & \text{si } U(x) \neq \varnothing \\ 0 & \text{sinon} \end{cases}$$

LS est ensuite utilisé pour approximer le gradient sous l'intégrale de l'équation 5.1. Soit $\pi_{a \rightleftharpoons b} = (p_1, p_2, \ldots, p_n)$ une approximation discrète de l'arc $\sigma_{a \rightleftharpoons b}$, avec $p_1 = a$, $p_n = b$, et tel que $\forall i \in \mathbb{N}_{n-1}^*, U(p_i) \cup U(p_{i+1}) \neq \varnothing$.

$$\|\nabla f\|_{p_i, p_{i+1}} = \begin{cases} LS(p_i) & \text{si } f(p_i) > f(p_{i+1}) \\ LS(p_{i+1}) & \text{si } f(p_i) < f(p_{i+1}) \\ \frac{1}{2}\left(LS(p_i) + LS(p_{i+1})\right) & \text{si } f(p_i) = f(p_{i+1}) \end{cases}$$

Le terme $T_f(a, b)$ est ensuite approché par :

$$T_f^\pi(a,b) = \bigwedge_\pi \sum_{i \in \mathbb{N}_{n-1}^*} \|\nabla f\|_{p_i, p_{i+1}} \cdot d_c(p_i, p_{i+1})$$

(i) de l'anglais « *chamfer* »
(ii) LS pour « *lower slope* »

N'Guyen & al. [NWvdB03] proposent une autre approximation dans le cas $f(p_i) = f(p_{i+1})$, c'est à dire sur les plateaux: la définition initiale de Meyer déplace la *lpe* lorsque le chemin contient un plateau. Selon les auteurs, en prenant $\|\nabla f\|_{p_i,p_{i+1}} = \wedge\{LS(p_i), LS(p_{i+1})\}$ lorsque $f(p_i) = f(p_{i+1})$, la ligne de partage des eaux ne subirait pas de décalage.

Propriétés Puisque la *lpe* est un squelette par zones d'influence, elle n'est pas une notion locale. Il est donc impossible de déduire si un point est *lpe* à partir des informations d'un voisinage.

La distance topographique T_f de l'équation 5.1 ne satisfait pas les propriétés d'une fonction distance. La raison est que T_f est *aveugle* sur les plateaux de f, et ne permet pas de distinguer deux points d'un même plateau. En effet, soit $(x, y) \in \mathbf{E}^2, x \neq y$ deux points d'un même plateau, et un arc $\sigma = \sigma_{x \rightleftharpoons y}$ totalement contenu dans ce plateau:

$$\forall z \in [0,1], f \circ \sigma(z) = f \circ \sigma(0) = f(x) = f(y)$$

f est constant à l'intérieur d'un plateau et donc de dérivée nulle. Nous avons alors $\forall z \in]0,1[, f' \circ \sigma(z) = 0$, d'où $l(f, \sigma) = 0$.

Ceci conduit à l'impossibilité de définir un diagramme de Voronoï directement à l'aide de T_f. Dans [RM00], un algorithme est proposé de manière à éviter ce cas spécifique et permettant de garder la définition de la *lpe* selon une distance topographique. Pour cela, l'image du relief est préalablement transformée par un algorithme de *complétion basse* [(iii)], de manière à ce qu'il n'existe aucun plateau qui ne soit pas un minimum local. La définition de distance topographique satisfait alors la définition de fonction distance, et valide la correspondance entre cette représentation et la *lpe*.

5.1.2.3. Autres formulations de la lpe

L'expression des bassins versants selon l'équation 5.2 n'est liée qu'au relief topographique (ses minima et une fonction dépendant uniquement de ce dernier). Rappelons que dans le problème de la *lpe*, les bassins versants coïncident avec la notion de régions d'intérêt. Dans certains problèmes, des informations de nature autre que celles liées au relief viennent enrichir la notion de régions: la couleur moyenne, la forme de son contour, etc.. On parle alors de *contraintes* sur les régions. Or l'équation 5.2 ne laisse que peu de liberté lorsque l'on souhaite faire intervenir ces contraintes. Il existe principalement deux approches pour contourner cette limitation, les deux proposant une nouvelle formulation de la *lpe*.

Problème d'optimisation La première approche consiste à formuler la *lpe* comme un problème de *minimisation d'énergie*, et donc d'*optimisation*: l'énergie est une fonction sur l'ensemble des régions, et la *lpe* est la partition minimisant cette fonction. Cette approche est utilisée par N'Guyen & al. [NWvdB03] et Beare [Bea06]. Soit $\{\Omega_i\}_{i \in \mathbb{N}_n}$ la famille de régions de manière à ce qu'elle forme une partition de l'espace $\mathbf{E}$ des coordonnées. Dans [NWvdB03], les auteurs qualifient la *lpe* comme la partition minimisant l'énergie suivante:

$$\mathcal{E}(\Omega_1, \ldots, \Omega_n) = \sum_{i=1}^{n} \int_{\Omega_i} (f(m_i) + T_f(\mathbf{x}, m_i))\, d\mathbf{x} \tag{5.3}$$

où T_f est la distance topographique (équation 5.1) et m_i est un minimum local de Ω_i. Cette formulation permet ensuite aux auteurs d'ajouter un terme de régularisation du contour des Ω_i, de la manière suivante:

$$\mathcal{E}(\Omega_1, \ldots, \Omega_n) = \sum_{i=1}^{n} \left(\int_{\Omega_i} (f(m_i) + T_f(\mathbf{x}, m_i))\, d\mathbf{x} + \beta \cdot \int_{\partial\Omega_i} ds \right) \tag{5.4}$$

où β est le facteur de régularisation, $\partial\Omega_i$ le contour de la région Ω_i. La première intégrale est effectuée sur la région, alors que la seconde est le long du contour (selon l'abscisse curviligne s).

[(iii)] de l'anglais « *lower completion* »

La minimisation d'une telle énergie est en fait assez complexe. Les auteurs de [NWvdB03] proposent deux méthodes: la première dérive de cette équation une équation aux dérivées partielles. Cette équation est ensuite utilisée dans l'évolution des contours des régions. Cependant, comme le mentionnent les auteurs, cette méthode ne garantit pas la minimisation de l'énergie. La deuxième méthode consiste à partir d'une partition initiale donnée par la *lpe*, puis de modifier la position des points des contours de manière à abaisser l'énergie. Puisque cette modification est effectuée sur les contours des régions, il s'agit en fait de réassigner les points des contours aux bassins versants adjacents. Enfin, notons que la discrétisation des termes d'énergie n'est pas un problème trivial.

Contours actifs La deuxième approche est liée à la propagation, plus précisément à la manière de construire les bassins versants. Les bassins versants sont considérés comme étant l'intérieur d'un *contour* Γ. Ce contour est la limite d'une suite de contours, dont la construction est récursive et gouvernée par une Équation aux Dérivée Partielle (EDP). Cette méthode est connue sous le nom de *contours actifs*. En ce qui concerne la *lpe*, Maragos [Mar05] propose l'évolution du contour Γ selon l'EDP suivante:

$$\frac{\partial \Gamma}{\partial t} = \frac{c_0}{\|\nabla f\|} \cdot \vec{\mathbf{n}} \tag{5.5}$$

où le vecteur $\vec{\mathbf{n}}$ est la normale au contour Γ dirigée vers l'extérieur de la région, et c_0 est une constante positive. Nous initialisons Γ au temps $t = 0$ par le contour intérieur du marqueur. Selon cette équation, la vitesse de propagation en un point du contour Γ est inversement proportionnelle à la valeur du gradient en ce point. Ceci signifie qu'un point sur un fort gradient évoluera lentement, ce qui est censé simuler le processus de montée des eaux.

La formulation précédente est dite à *hauteur uniforme* (cf. [SM03]) : le liquide *monte* d'une hauteur constante, sur toute l'image et à chaque pas de temps. Sofou & Maragos [SM03] expriment la même évolution lorsque l'augmentation de volume du bassin versant est considérée uniforme à chaque pas de temps. L'EDP devient :

$$\frac{\partial \Gamma}{\partial t} = \frac{c_0}{\mathcal{A}(t)\,\|\nabla f\|} \cdot \vec{\mathbf{n}}$$

où $\mathcal{A}(t)$ est la surface intérieure à Γ au temps t. Nous voyons bien que le modèle fonctionne impeccablement si $\|\nabla f\| \neq 0$, ce qui n'est pas le cas sur les plateaux. Il est bien sûr possible de contourner ce problème selon la même méthode de *complétion basse* que l'on a évoquée précédemment.

Enfin, remarquons qu'étant donnée l'orientation de l'évolution, le contour croît tout autour du marqueur jusqu'à un contour maximal, stoppé par collision avec d'autre contours. Le marqueur est donc nécessairement à l'intérieur du contour final.

Remarques Les problèmes de minimisation d'énergie et de contours actifs sont assez corrélés. Il est en effet possible dans certains cas de déduire une EDP de manière analytique à partir de l'expression de l'énergie. Cette approche est qualifiée de *variationnelle*, puisque l'on s'intéresse à la fonction *dérivée* de l'énergie, dont on cherche ensuite les minima. Cette dérivée est exprimée sous forme d'EDP. La formulation analytique n'est cependant pas triviale. Dans la suite, nous nous servirons de ces formulations lorsque nous contraindrons l'algorithme original de ligne de partage des eaux.

Si la formulation par un problème d'optimisation est élégante et attractive, nous émettons toutefois les réserves suivantes:

1. on ne peut savoir si le minimum de l'énergie contient effectivement les marqueurs. En effet, la partition finale peut dépendre fortement de la fonction de coût « additive » (celle qui n'est pas liée au relief topographique). Une résolution de ce problème d'optimisation pourrait déplacer complètement la partition par rapport aux marqueurs: les régions ne croîtraient plus autour des marqueurs. Par ailleurs, il n'a jamais été question, dans la littérature, d'étudier les fonctions de coût compatibles avec la *lpe*. Enfin, dans [NWvdB03], pour arriver à une minimisation effective de la fonction d'énergie, les auteurs partent finalement d'une partition initiale donnée par la *lpe*. Il est prouvé que cette partition minimise la première intégrale seulement, et non la somme des

deux intégrales. Ainsi, il semblerait que les auteurs cherchent en fait une partition proche de la *lpe*, dont l'énergie serait un minimum local.

2. le problème d'optimisation est bien loin d'une méthodologie algorithmique pour la résolution. S'il donne effectivement un cadre formel au problème, à notre connaissance ce cadre ne sert que de formulation et n'est pas repris dans les schémas d'implémentation (cf. [NWvdB03][(iv)] et plus récemment [Bea06]).

L'approche par évolution de contours est très proche du problème d'optimisation, car comme nous l'avons dit, l'optimisation conduit souvent à une minimisation en calculant la dérivée de l'énergie. Cependant, cette formulation n'est pas tout à fait correcte. Hormis le problème évident de définition sur les plateaux, cette formulation ne peut garantir qu'un point à une hauteur h sera inondé après l'inondation de la totalité des points de hauteur jusqu'à $h-1$. À défaut d'un cadre rigoureux, la formulation donne cependant un support intuitif au processus de propagation qui nous semble adapté pour décrire certaines modifications de l'inondation.

[(iv)] dont la justification face aux EDP de Maragos est plus que douteuse

5.2 Algorithme

Nous allons à présent nous pencher sur l'aspect algorithmique de la transformation de ligne de partage des eaux. Il existe dans la littérature un grand nombre d'implémentations, sur des processeurs généralistes ou dédiés. Les principaux axes actuels de recherche concernent la parallélisation de l'algorithme et l'amélioration des vitesses de calcul, dans les perspectives des architectures, des processeurs dédiés (type FPGA ou ASIC), et de la segmentation en temps réel. Une implémentation hardware de référence se trouve dans [KLGP95], une amélioration récente de ces travaux dans [Dej04] montre l'actualité de ce sujet.

Si l'on se restreint au cadre de processeurs généralistes, la *lpe* dispose d'un certain nombre d'algorithmes. Deux d'entre eux font figures de référence. L'un d'eux, proposé par L.Vincent & P.Soille [VS91], utilise des files d'attente. Il effectue d'abord un tri des points du relief pour calculer ensuite le SKIZ géodésique à l'intérieur de chaque seuil h (ensemble de points de l'image d'altitude inférieure ou égale à h). Pour calculer ce SKIZ, les auteurs utilisent le résultat du SKIZ géodésique au seuil $h-1$ et une propagation à l'aide de files d'attente. L'étape de tri permet d'accéder efficacement aux différents niveaux de seuil. Par ailleurs, l'article original [VS91] souligne certains problèmes liés à la discrétisation, comme celui de l'épaisseur de la *lpe* (figures 10 à 12 page 589) ou celui de déconnexion des bassins versants (figure 13 page 590). Le premier problème est lié à l'implémentation du SKIZ, dépendant du calcul correct de la distance géodésique à chaque niveau de seuil. Le second est lié à la règle déterminant les points appartenant à la *lpe*. Dans certaines configurations « pathologiques », la règle assignant comme point *lpe* tout point ayant strictement plus d'une étiquette dans son voisinage peut conduire à la déconnexion du bassin versant initial. Pour résoudre ce problème, les auteurs proposent de changer cette règle en considérant *lpe* tout point ayant une étiquette strictement plus petite dans son voisinage. Cela entraîne la violation de la symétrie du problème - comme le notent les auteurs - et nécessite également un ordre dans l'espace des étiquettes - ce qui est en fait inutile puisque les étiquettes sont des symboles et non des valeurs. Puisqu'il n'y a pas de meilleure solution, nous ne considérerons pas ce genre de cas dans la suite.

Le second algorithme de référence, celui sur lequel nous allons travailler, a été proposé par Meyer [Mey91]. Il tire profit des files d'attente hiérarchiques, qui sont des structures informatiques incluant des mécanismes de tri des données (voir chapitre 2 page 33). Cet algorithme possède quelques avantages intéressants par rapport à l'algorithme de Vincent & Soille, présentés ci-après:

1. Le tri initial des points de l'image dans [VS91] est relativement coûteux. Selon les auteurs, il nécessite trois passages complets de l'image, ainsi qu'une zone de stockage des points triés de 4 fois la taille de l'image (les points sont stockés sous forme de pointeurs). En revanche, l'algorithme de Meyer propose de ne trier que les données nécessaires à un instant donné de la propagation. Les coûts de stockages sont ainsi grandement réduits.
2. L'algorithme de Meyer effectue, selon l'auteur, un nombre de passages dans l'image moins important que l'algorithme de Vincent & Soille. Il est donc plus performant en termes de temps de calcul.
3. D'un point de vue purement algorithmique, le tri n'est pas écrit de manière explicite dans l'algorithme, mais est effectué à l'intérieur de la structure de *fah*. Ainsi, l'algorithme est disjoint d'une implémentation particulière de *fah*.

L'algorithme de Meyer est le plus performant connu à ce jour. Nous allons en donner les détails puis nous mettrons en évidence un défaut d'isotropie, selon les remarques de Beucher [Beu04]. Nous proposerons ensuite un algorithme corrigé.

Puisque nous disposerons d'un algorithme correct (par rapport à la définition initiale de la *lpe* par Beucher & Lantuéjoul [BL79]), et formulé selon les principes de programmation générique exposés dans le chapitre 2, nous verrons dans quelle mesure il est possible d'opérer sur des images en dimension quelconque, et des images de reliefs en précision à virgule flottante ou « couleur » à l'aide des

treillis lexicographiques. En conjonction avec les reconstructions lexicographiques (§4.3.1 page 130), il nous sera alors possible de proposer une version de la hiérarchie par cascades sur ces mêmes treillis lexicographiques.

5.2.1. L'algorithme par files hiérarchiques d'attente

L'algorithme à base de files d'attente hiérarchiques proposé par Meyer [Mey91] simule un processus d'inondation. Étant donné qu'on ne peut inonder deux fois un même point - chaque point appartient à au plus un bassin versant - le processus d'inondation est une labellisation concurrente de l'image par tous les bassins versants. Nous présentons d'abord l'algorithme original, puis en détaillerons les biais.

5.2.1.1. Description

L'algorithme est un processus de propagation basé sur les files d'attente hiérarchiques. Le principe est le suivant: nous possédons en entrée une image de relief et une image marqueur. Les marqueurs font offices de source d'eau. On déplace progressivement les contours des marqueurs, de manière à faire grandir les régions qu'ils décrivent. Pour cela, on cherche les nouveaux points dans le voisinage du contour. Chaque voisin est ensuite intégré à la région lorsque son altitude le permet.

Comme d'autres algorithmes de propagation (cf. par exemple la reconstruction 4.2 page 133), il se scinde en deux parties distinctes: l'initialisation de la file d'attente et la propagation des points sur l'image. Les détails de l'algorithme sont présentés ci-dessous, et illustrés sur la figure 5.2. Nous considérons uniquement l'algorithme créant une ligne de partage des eaux d'épaisseur 1 [(v)].

L'algorithme original proposé par Meyer [Mey91] est présenté dans l'algorithme 5.1. Il prend en entrée une image de labels « imLabels » et de relief « imRelief », et crée une image de travail « imTravail ». L'image de travail permet de déterminer l'état des points sans recherche à l'intérieur de la *fah*. En effet, la recherche d'un point à l'intérieur de la *fah* est une méthode coûteuse en temps d'exécution, alors que l'accès à un point de l'image de travail est immédiat. Cette méthode a par contre l'inconvénient d'être plus gourmande en termes de consommation mémoire. Chaque point de l'image de travail possède l'un des trois états suivant :

- **candidat** : le point n'a pas de label et peut être inondé (c'est à dire peut prendre un label).
- **empilé** : le point apparaît dans la *fah*. Cet état permet notamment d'éviter les doublons dans la *fah*.
- **lpe** : le point appartient à la ligne de partage des eaux.

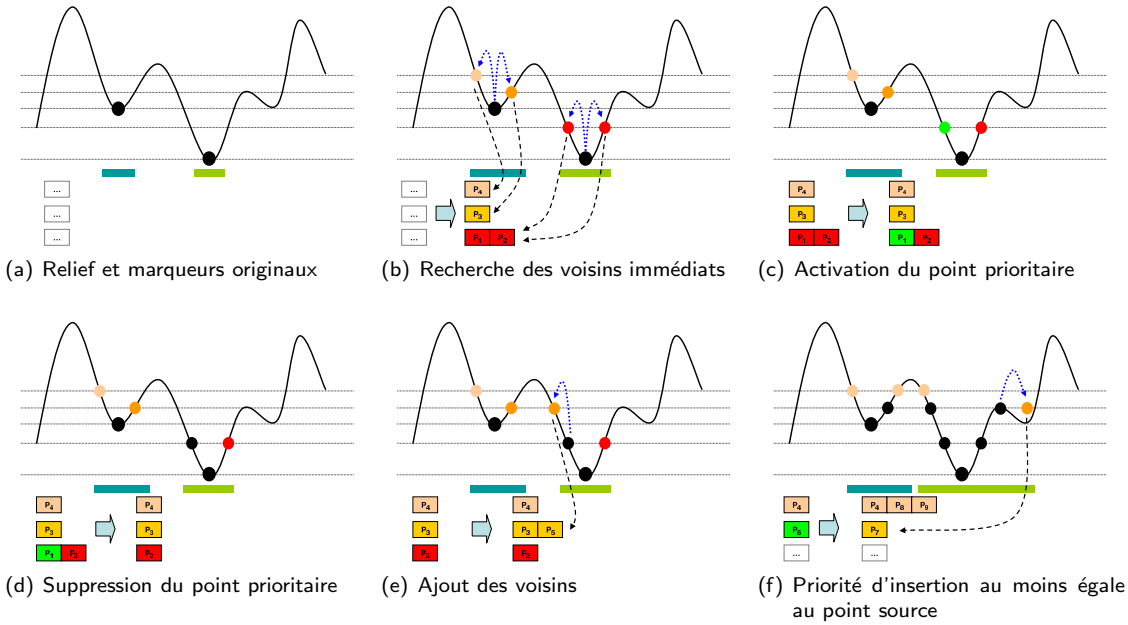
Ainsi, grâce à l'image de travail, il est facile d'éviter les doublons ou l'insertion de point *lpe* dans la *fah*.

L'algorithme possède deux phases distinctes: l'initialisation et le traitement. L'initialisation prépare les données à partir des marqueurs d'inondation. Il s'agit essentiellement de déterminer l'état des points et le contenu de la *fah*. Le traitement propage ensuite séquentiellement les points initiaux, jusqu'à un critère d'arrêt. Voici ces étapes plus en détails.

Initialisation L'algorithme détecte les points des marqueurs d'inondation (les points noirs sur la figure 5.2(a)). La *fah* est supposée totalement vide au départ. Pour des raisons d'illustration, nous avons supposé qu'il n'y a que 4 niveaux d'altitude différents. L'altitude la plus basse correspond au marqueur le plus bas. Pour chaque marqueur, l'algorithme recherche ensuite les voisins qui n'ont pas été marqués. Sur la figure 5.2(b), il s'agit des points voisins des points noirs. Ces points voisins prennent la même étiquette que leur marqueur (labellisation à l'entrée de la *fah*). Ils sont insérés dans la *fah* à la priorité correspondant à leur altitude.

La variante *automatique* de la *lpe* consiste à prendre pour marqueurs les minima locaux du relief. À chacun de ces minima sera ensuite assignée une étiquette unique.

[(v)] Meyer présente également un autre algorithme dans [Mey91] qui ne crée pas de *lpe* mais affecte à chaque point un bassin versant unique



(a) Relief et marqueurs originaux (b) Recherche des voisins immédiats (c) Activation du point prioritaire

(d) Suppression du point prioritaire (e) Ajout des voisins (f) Priorité d'insertion au moins égale au point source

Fig. 5.2.: Illustration de l'algorithme de *lpe* à base de *fah*. (a) Relief original et marqueurs. Les points noirs représentent les marqueurs. Les étiquettes des marqueurs sont données par les rectangles de couleur sous le relief. La *fah* (en bas de la figure) est supposée vide. Le sens de traitement de la *fah* est du bas vers le haut (priorité) et de gauche vers la droite. (b) les voisins des marqueurs et n'étant pas eux-même marqueurs sont insérés dans la *fah*, à priorité correspondante à leur altitude (ici tous les points colorés). (c) le point le plus prioritaire de la *fah* est extrait de celle-ci. (d) Ce point disparaît ensuite de la *fah*, et est étiqueté. (e) Les voisins du point nouvellement sorti sont ensuite insérés à leur tour dans la *fah*, de la même manière que (b). (f) Dans le cas où le point voisin aurait une altitude inférieure à son point source (le couple de points reliés par la flèche bleue), le point voisin est inséré à une priorité égale au point source.

Traitement Tant que la *fah* n'est pas totalement vide, les deux opérations suivantes sont effectuées:

1. Le point le plus prioritaire est extrait de la *fah* (point sortant). Sur la figure 5.2(c), il s'agit du point en vert. Ce point est ensuite détruit de la *fah* (figure 5.2(d)).
2. S'il n'existe qu'une unique étiquette dans le voisinage du point sortant, alors ses points voisins et non étiquetés sont à leur tour insérés dans la *fah*. Leur priorité d'insertion correspond à leur altitude, à l'instar de l'initialisation. Pour une priorité donnée, ces voisins sont introduits à la fin de la file d'attente (sur la figure 5.2(e), P_5 est inséré après P_3). Ils prennent l'étiquette du point sortant (ici encore, il s'agit d'une labellisation à l'entrée de la *fah*).
3. S'il existe strictement plus d'une étiquette dans le voisinage du point sortant, alors aucun voisin n'est inséré dans la *fah*. Le point sortant est ensuite marqué comme appartenant à la *lpe*.

Enfin, notons que dans la description initiale de Meyer [Mey91], il n'y a pas de condition particulière sur la priorité d'insertion d'un point dans la *fah*, en fonction du point source. Par rapport aux implémentations que nous connaissons de cet algorithme, un point ne peut être inséré à une priorité supérieure de son point source, comme l'illustre la figure 5.2(f).

5.2.1.2. Biais

Nous avons rencontré deux types de biais ou erreur. Le premier n'est pas lié à l'article de Meyer que nous avons décrit dans la section précédente, mais est apparu dans certaines implémentations que

Algorithme 5.1 : *lpe* à files d'attente hiérarchiques - Version originale par F. Meyer [Mey91]

Data : imRelief, imLabels, voisinage $\mathcal{N}$
Result : imSortie : image de *lpe*

```
1  imSortie ← imLabels                    // l'image de labels marqueurs est copiée dans l'image de sortie
2  imTravail ← candidat                   // tous les points de imTravail sont initialisés à l'état « candidat »
3  fah ← ∅
4  • Initialisation
5  forall p ∈ imLabels do
6      label-courant = imLabels(p)
7      if label-courant = label-fond then continuer          // passage au point p suivant
8      imTravail(p) = labélisé                               // le point central est fermement labélisé
9      forall v ∈ N_p \ p do
10         Parcours du voisinage sans le point central
11         if imLabels(v) = label-fond;                      // les voisins sans label sont ajoutés dans la fah
12         then
13             if imTravail(v) = candidat then
14                 imTravail(v) = empilé                     // évite les doublons dans la fah
15                 imSortie(v) = label-courant
16                 fah ← ajouter v à la priorité imRelief(v)
17
18
```

nous avons pu voir. Il s'agit pour nous d'une raison *suffisante* pour le souligner. Le second biais est lié à l'ordre de traitement des points d'une même priorité. Ce biais est par contre lié directement à l'algorithme de Meyer.

Erreur lors de l'insertion de nouveaux points Soulignons encore une fois que l'article initial de Meyer [Mey91] ne conduit pas à ce biais. Le but ici est de détailler une erreur due à l'implémentation. Rappelons que les points de la ligne de partage des eaux sont non-inondants. Ceci signifie qu'on ne peut inonder des points à partir d'un point *lpe*, et donc qu'on ne peut remplir la *fah* avec des voisins candidats d'un point *lpe*.

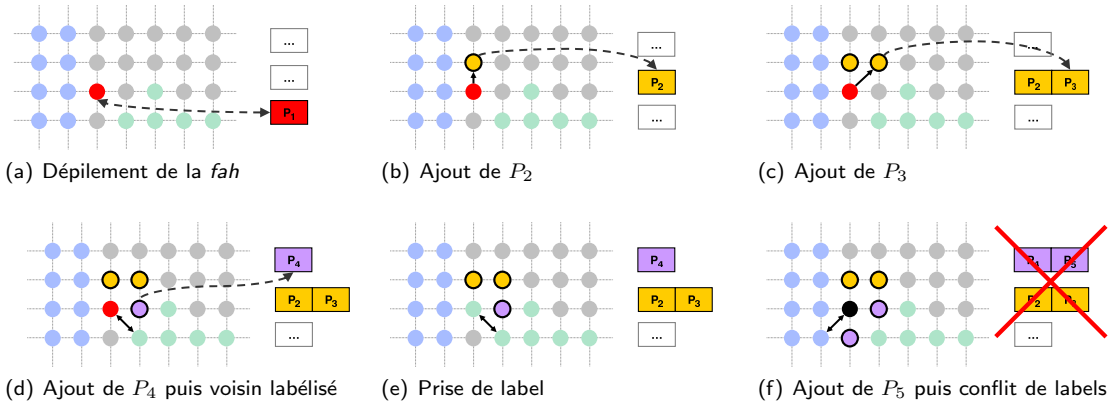
Ce mécanisme suppose la résolution de l'état du point d'intérêt (le point central), avant toute insertion de nouveaux points dans la *fah*. Nous avons eu l'occasion de constater que cela n'était pas toujours effectué de manière correcte. Une erreur que nous avons rencontrée est par exemple l'insertion progressive des voisins pendant leur découverte, et l'insertion est stoppée lorsque le point central devient *lpe*. Un exemple d'une telle erreur est donné en figure 5.3. L'ordre de parcours du voisinage (ie. le graphe de connexité) a un impact important, et la validité de l'assertion « *les points lpe sont non-inondants* » n'est pas vérifiée.

La résolution de ce problème est en fait assez simple. Elle peut se faire en traitant le voisinage de manière **atomique**, en deux passes:

1. test sur la totalité des points concernés par le traitement.
2. validation du traitement.

Cette démarche empêche la propagation (ajout du voisinage dans la *fah*) dans les cas ambigus. Les points ne sont plus traités de manière isolée mais en conformité à une configuration de voisinage adéquate. Par ailleurs, même dans le cas d'une implémentation correcte, certains pixels de l'image peuvent ne pas être traités à cause de configurations particulières empêchant la propagation.

Biais dû au traitement des files d'attente & problème de parité Ce biais est dû à la manière dont sont traités les points d'une même priorité par la *fah*.



(a) Dépilement de la *fah* (b) Ajout de P_2 (c) Ajout de P_3

(d) Ajout de P_4 puis voisin labélisé (e) Prise de label (f) Ajout de P_5 puis conflit de labels

Fig. 5.3.: Biais sur la détection locale des points de la ligne de partage des eaux. Sur ces figures, il existe deux labels: le label bleu et le label vert. Les points en gris sont des *candidats* potentiels, non labélisés et dans aucune file. Le point rouge est le point « central » dont on souhaite déterminer l'état. (a) dépilement d'un point de la *fah*. Le point rouge devient le point central. (b)–(c) Ajout des voisins du point rouge dans la *fah*, selon l'ordre de parcours du voisinage, et à la priorité correspondante (non représentée sur les schémas). (d)–(e) le parcours du voisinage arrive sur le label vert, le point central n'ayant encore aucun label devient donc de label vert (labellisation à la sortie de la *fah*). (f) le parcours du voisinage se poursuit, et un label bleu est découvert. Comme ce dernier est déjà « vert », il y a conflit de label: le point central est donc *lpe*. Puisqu'il est *lpe*, il est non-inondant et ne peut ajouter un quelconque candidat dans la *fah*. Il faut donc supprimer tous les points P_2 à P_5 de la *fah* et remettre leur état à « candidat ».

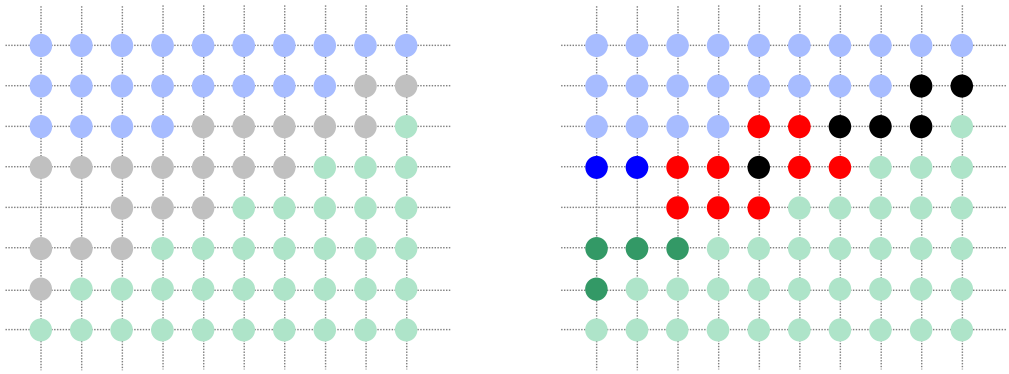
Lorsque nous avons introduit le cadre théorique (§5.1.2), nous avons souligné le fait qu'à l'intérieur des plateaux, l'algorithme de ligne de partage des eaux se comporte comme une fonction distance par rapport aux bords descendants de ce plateau [(vi)]. Ainsi, lorsqu'un plateau est inondé à partir de ses bords avec deux labels différents, la ligne de partage des eaux est placée selon la distance (trame) aux labels. Pour avoir une distance correcte, l'isotropie est nécessaire. Or, l'ordre de traitement des points d'une même priorité et à une certaine étape de l'itération ne doit absolument pas intervenir dans le résultat. Cette condition n'est pas vérifiée dans l'algorithme de Meyer si l'on se réfère à la figure 5.4.

Cette figure illustre le biais, pouvant être considéré comme un *effet de bord*. En effet, deux points équivalents en terme de traitement « interagissent », et invalident ainsi leur équivalence. Ces interactions apparaissent naturellement à la frontière des bassins versants : labéliser un tel point déplacera la frontière du bassin versant concerné en ce point, mais également de ses voisins directs puisque la labellisation valide la propagation sur le voisinage. C'est également un biais parce qu'il dépend de deux phénomènes externes à l'algorithme:

1. l'insertion des points est dépendante du graphe de connexité utilisé pour la propagation.
2. la structure informatique utilisée pour le stockage des points d'un plateau de priorité (c'est à dire l'ensemble des points de la *fah* à priorité constante) ne conserve pas nécessairement l'ordre d'insertion.

La résolution de ce problème est assez délicate. Il faut traiter l'ensemble des points d'un plateau de priorité de manière totalement *atomique*, c'est à dire en considérant l'éventuelle interaction entre les points. La détermination de l'appartenance d'un point à la *lpe* impose alors une vision « préventive » des points de la file d'attente, et à plus grande échelle que le voisinage unité. C'est pourquoi la *lpe* doit être calculée de manière **épaisse**.

[(vi)] Pour être plus précis, il s'agit des bords inférieurs, c'est à dire les bords dont les voisins sont d'altitude strictement inférieure



(a) Grille initiale (b) Propagation d'un pas unitaire

Fig. 5.4.: Problème de parité de l'algorithme à base de *fah*, lié à l'ordre de traitement. Supposons les priorités des points des deux grilles (a) et (b) toutes égales. (a): les labels vert et bleu inondent la grille. Le front de propagation est représenté en gris. (b): les points noirs sont les points de la ligne de partage des eaux, et ne posant pas d'ambiguïté (ie. deux voisins de labels différents, en 8-connexité). Les points bleus et verts foncés sont ceux fermement labélisés. Les points rouges par contre sont ceux acquérant un label différent (bleu, vert ou *lpe*) selon l'ordre de propagation des labels, et tel qu'il a été défini dans l'algorithme de Meyer.

Soit p le point central sur lequel nous effectuons le test de *lpe*. Nous savons que p est dans la file d'attente, et qu'il possède au moins un voisin labélisé. Supposons qu'il ne possède qu'un seul voisin labélisé. Si l'un des voisins v de p n'est pas encore labélisé, mais se trouve également dans la file d'attente, alors v possède un voisin labélisé et le problème mentionné apparaît : un test de collision est alors nécessaire. Si v est en cours d'inondation par le même label que p, alors il n'y a pas de collision, p peut être labélisé fermement. Si par contre v est inondé par un label autre que celui de p, alors p et v sont en collision, bien qu'aucun de ces deux points ne soit directement voisin de deux labels différents. Nous obtenons alors une ligne de partage des eaux *épaisse*, de deux points d'épaisseur (cf. les points rouges sur la figure 5.4(b)).

Conséquences du biais Nous faisons référence ici au second biais exposé, et non au premier qui est en fait une erreur d'implémentation. L'isotropie de l'algorithme garantit le fait que toutes les directions sont traitées avec autant d'importance. De manière indirecte, l'isotropie signifie l'indépendance de l'algorithme par rapport à l'ordre de traitement des points du graphe de connexité utilisé pour la propagation.

Le biais mentionné augmente la taille d'un bassin versant au détriment d'un autre. L'algorithme de Meyer présente donc un *défaut d'isotropie*. Par ailleurs, la *lpe* est déplacée en de nombreux points, et ce déplacement peut avoir des conséquences importantes pour des algorithmes utilisant la *lpe* de manière récursive (comme par exemple l'algorithme des cascades dû à Beucher [Beu91] [(vii)]). De manière indirecte, ce biais invalide le formalisme basé sur les distances topographiques, exposé en §5.1.2.2 (page 180).

Enfin, ajoutons qu'il n'existe pas dans l'algorithme de Meyer de mécanisme garantissant l'isotropie. Cette propriété est simulée en ajoutant les points découverts pendant la propagation à la fin des points déjà connus . Ainsi, un nouveau point p de priorité h sera placé à la fin de la file d'attente de priorité h. Ce mécanisme ne conserve correctement l'isotropie qu'en dimension deux, dans la mesure où l'ordre d'empilement des voisins et de désempilement des points de la file est conservé. Ceci n'est par contre plus vrai en dimension supérieure (trois et plus) [(viii)].

[(vii)] ce biais se propage alors sur chaque nouvelle hiérarchie.
[(viii)] nous avons souligné ce point au §2.2.6 page 33

L'algorithme de Vincent & Soille ne présente pas ce biais. Cependant, étant donné les avantages de l'approche par files d'attente hiérarchiques - que nous avons exposés au début du §5.2.1 - nous souhaitons conserver cette même technique. Nous proposons dans la suite un algorithme corrigé à base de *fah*, et produisant une ligne de partage des eaux épaisse [(ix)].

5.2.2. Algorithme de ligne épaisse de partage des eaux isotrope, à files hiérarchiques d'attente

Les auteurs de [BDB06] prennent en compte les considérations de Beucher [Beu04] afin de résoudre le problème des biais. La méthode de Beucher consiste à distinguer un label temporaire d'un label définitif. Un point possède un label temporaire lorsqu'il est inséré dans la *fah*. Ce label devient ensuite définitif lorsqu'il n'y a pas de collision avec un point voisin. La collision existe lorsque le voisin possède également un label temporaire différent. Ce mécanisme permet de résoudre le problème de parité souligné dans la section §5.2.1.2 précédente.

Cette méthode double le nombre de labels. Or pour les grandes images, ce nombre peut être assez conséquent. Ainsi pour les images actuelles, une représentation de l'image de labels en 32 bits devient courante. Cependant, nous constatons que nous pouvons effectuer ces mêmes opérations en ajoutant un état dans l'image de travail. Rappelons que le recours à une image de travail est nécessaire dans l'algorithme à base de files d'attente hiérarchiques. En effet, cette image permet d'éviter l'insertion de doublons dans la *fah* lors de la propagation (étape de traitement), de déterminer si un point est *lpe* et de ne pas le considérer dans le test de collision des bassins versants. Il serait possible d'éviter une telle image dans la mesure où il existe une redondance entre l'information contenue dans la *fah* et l'image de travail. Cependant, l'accès à ces informations par une image de travail est beaucoup plus rapide (en $O(1)$) qu'une recherche de l'état correspondant dans la *fah* (minoré par $O(\log n)$ avec n le nombre de priorités de la *fah*, auquel s'ajoute la recherche dans une file d'attente simple de complexité généralement linéaire).

Enfin, l'implémentation que nous proposons peut être faite directement avec des files simples de la *STL* et la structure de files d'attente hiérarchiques, comme nous l'avons présenté au chapitre 2. Enfin, nous présentons cet algorithme de manière très détaillée, afin d'accompagner entièrement le lecteur vers une implémentation qui nous semble correcte, tout en évitant quelques raccourcis qui pourraient mener à des biais.

L'algorithme de ligne de partage des eaux **isotrope** et **épaisse** à base de file d'attentes hiérarchiques est présenté en trois sous-algorithmes 5.2, 5.3 et 5.4. Nous reprenons la partie précédente pour la résolution des biais. L'algorithme 5.2 présente la phase d'initialisation, l'algorithme 5.3 la phase de propagation et enfin l'algorithme 5.4 les mises à jour dans les espaces images après propagation.

Remarque D'après ce qui précède, un point possède un nombre d'états limité. Ces états sont :
- **labélisé** le point est fermement labélisé, et son label ne changera pas par la suite, mais il pourra servir à la détermination de l'état d'autres points.
- **lpe** le point fait partie de la ligne de partage des eaux. Son état ne changera pas par la suite, et ne servira pas à déduire l'état d'autres points.
- **candidat** le point ne possède pas encore de label. Par ailleurs, le point ne se trouve dans aucune file d'attente.
- **empilé** le point apparaît dans au moins un niveau de hiérarchie de la file d'attente hiérarchique. Il ne possède pas encore de label, mais possède au moins un voisin fermement labélisé.
- **lpe-temporaire** état intermédiaire entre les états *empilé* et *lpe*. Cet état servira pendant la propagation, pour décrire la collision de deux points d'étiquettes différentes et dans la file courante de traitement.

[(ix)] Le qualificatif « épais » signifie que la ligne de partage des eaux est représentée dans l'image par une ligne d'au moins un pixel d'épaisseur

Nous nous servons naturellement d'une image de travail « imTravail » contenant les états des points à inonder. Les labels sont par contre stockés dans l'image de sortie « imSortie ». Les marqueurs de l'inondation sont donnés par l'image « imLabels ». Dans cette image, nous supposons marqueur toute étiquette qui n'est pas « fond »: étiquette non nulle pour les images scalaires, non noire pour les images couleurs, etc.(x). Le fond sera simplement donné dans l'algorithme par « label−fond ».

Description de l'algorithme d'initialisation 5.2 L'initialisation prépare la file d'attente hiérarchique en vue de la propagation.

Pour chaque point p n'appartenant pas au fond (c'est à dire différent de « label−fond »), nous devons déterminer s'il appartient à la *lpe*. À l'instar de l'algorithme classique de Meyer, le point p appartient à la *lpe* s'il existe au moins un voisin de label différent de lui-même. Ce test est effectué ligne 13. À cette étape, aucun point n'est inséré dans la file.

Si le point n'est pas *lpe*, il est fermement labélisé, à savoir le point correspondant dans l'image de sortie « imSortie » prend son étiquette, et celui dans l'image de travail prend l'état « labélisé ». Les points voisins et candidats peuvent alors être insérés dans la file d'attente globale, en veillant à ne pas insérer de doublon. La règle importante à suivre est qu'un point est inséré dans la *fah* uniquement s'il n'est pas déduit d'un point de la *lpe*. Nous nous servons d'une file intermédiaire « f−file », dans laquelle nous ajoutons les points voisins du point p central, au cours de leur découverte. Ce mécanisme à base de file d'attente simple évitera le parcours multiple d'un même voisinage, car ce parcours peut être relativement coûteux.

La condition de détermination d'un point *lpe* n'est pas suffisante pour l'étape de propagation mais reste valide pour l'initialisation. La configuration de l'initialisation est effectivement beaucoup plus simple puisqu'aucun point ne peuple encore la file d'attente hiérarchique. Il n'y a pas de test de collision entre deux points placés dans la *fah* puisque l'état *empilé* n'existe pas encore.

Description de l'algorithme de propagation 5.3 Cette étape est plus complexe, puisqu'il faut traiter chaque priorité de manière atomique. Elle demande donc de la part du lecteur un peu de concentration.

Nous commençons par extraire l'ensemble « PL » des points de la priorité la plus importante (les points les plus bas du relief topographique), que nous traitons indépendamment du reste de la file d'attente globale « fah−globale ».

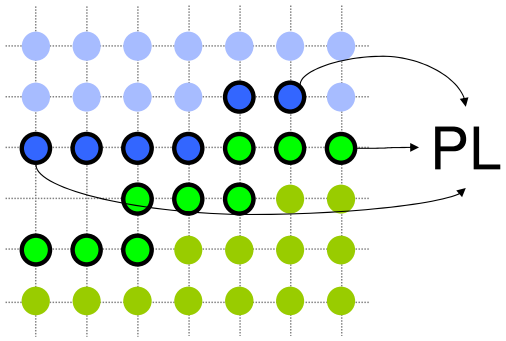


Fig. 5.5.: Illustration du traitement atomique des points de la *fah*. L'ensemble des points « empilés » et de priorité la plus forte sont extraits de la *fah* et placés dans « PL ».

La figure 5.5 illustre le principe: il existe deux étiquettes, verte et bleue. Les points entourés sont des points à l'état « empilé », c'est à dire apparaissant dans la *fah*, possédant une étiquette temporaire,

(x) Des techniques de méta-programmation simples, exposées en §2.3.1.1 page 38 permettent d'obtenir un fond « générique ».

Algorithme 5.2 : *lpe* épaisse isotrope à file d'attente hiérarchique - Initialisation

Data : imRelief, imLabels, voisinage $\mathcal{N}$
Result : imSortie : image de *lpe* épaisse sans biais

```
1  imSortie ← imLabels                          // l'image de labels marqueurs est copiée dans l'image de sortie
2  imTravail ← candidat                         // tous les points de imTravail sont initialisés à l'état « candidat »
3  f-candidat, f-file, f-watershed, f-inonder ← ∅
4  fah-globale ← ∅
5  forall p ∈ imLabels do
6      label-courant = imLabels(p)
7      if label-courant = label-fond then continuer                    // passage au point p suivant
8      est-lpe = faux
9      f-file ← ∅                                                      // « f-file » reçoit les points découverts
10     forall v ∈ N_p \ p do
11         label-voisin = imLabels(v)
12         if label-voisin ≠ label-fond then
13             if label-voisin ≠ label-courant then
14                 est-lpe = vrai
15                 break                                               // sortie de la boucle « for »
16
17         else f-file ← ajouter v
18     if est-lpe then
19         imSortie(p) = lpe
20         imTravail(p) = lpe
21     else
22         imTravail(p) = labélisé                                     // le point central est fermement labélisé
23         forall v ∈ f-file do                                        // les points découverts sont ajoutés dans la fah
24             if imTravail(v) ≠ empilé then
25                 imTravail(v) = empilé                               // évite les doublons dans la fah
26                 imSortie(v) = label-courant
27                 fah-globale ← ajouter v à la priorité imRelief(v)
28
```

mais non définitive. L'étiquette temporaire est égale à l'étiquette définitive dans l'image de sortie, mais l'état dans l'image de travail n'est pas le même. Les zones non marquées sont des points à l'état « candidat ». Le but du traitement est de traiter tous les points entourés de manière atomique, dans le but de leur donner une étiquette définitive ou de les marquer *lpe*. Cette étape a également pour but de trouver de nouveaux points, c'est à dire de propager les labels sur des éventuels candidats.

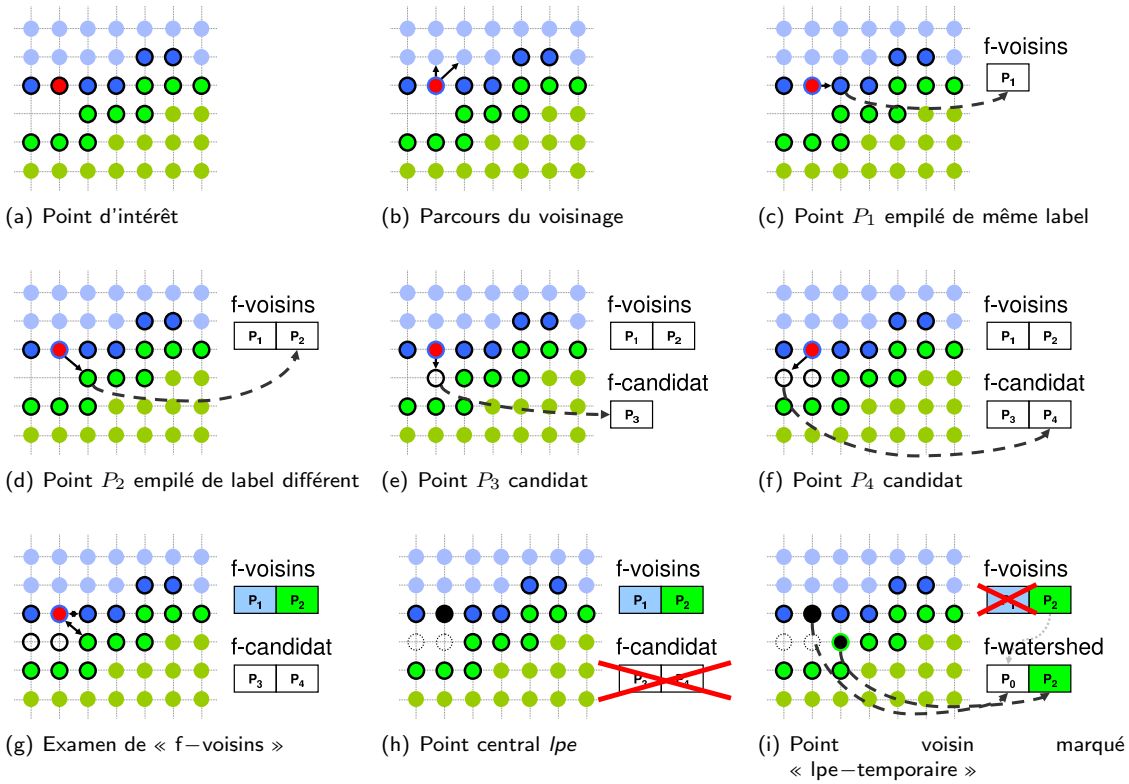
Nous ne pouvons introduire aucun nouveau point dans « fah−globale » avant le traitement complet de tous les points de « PL », car comme nous l'avons vu précédemment la propagation ne serait plus isotrope. Il faut également être très vigilant lorsque nous modifions un point directement dans l'espace image pendant cette phase de l'algorithme; pour contourner les éventuels biais, il faut être capable de dire que le changement d'état d'un point a eu lieu pendant la même étape de propagation de l'ensemble « PL ». Or ceci n'est pas possible en ne prenant que les états des points mentionnés dans l'algorithme de Meyer: un état supplémentaire est nécessaire et c'est précisément l'état « lpe−temporaire ». Nous allons donc effectuer la plupart des traitements de « PL » à l'aide de files d'attente simples intermédiaires. Nous nous servirons de l'état « lpe−temporaire » dans des configurations particulières que nous allons également décrire.

Pour chaque point p de « PL », nous extrayons l'ensemble de ses voisins v (ligne 12). À l'instar de l'étape d'initialisation, les voisins sont classés selon leur état dans « imTravail », mais nous distinguons dans la propagation un état supplémentaire, « empilé », qui complique le traitement. Récapitulons:

- Le voisin v est « candidat »: nous l'insérons dans une file intermédiaire « f−candidat » jusqu'à détermination de l'ensemble du voisinage.
- Le voisin est « labélisé »:
  - Dans le cas où le point courant p n'a pas encore reçu d'étiquette (« est−labelisé » est « faux »), « label−courant » prend l'étiquette de ce voisin; « est−labelisé » devient « vrai ».
  - Dans le cas contraire, nous devons tester l'étiquette du voisin. Si cette étiquette ne coïncide pas avec l'étiquette courante « label−courant », alors le point central p est *lpe*. La boucle est alors cassée car il n'y a pas lieu de continuer.
- Le voisin est dans l'état « empilé » ou « lpe−temporaire »: il est alors inséré dans une file intermédiaire « f−voisins ». Ceci est dans le but d'un test ultérieur, lorsque les points labélisés et voisins ne permettent pas de déduire si p appartient à la *lpe*.

Si le point central p est déterminé *lpe* à cette étape (ligne 28), il n'y a pas de tests supplémentaires à effectuer. Nous plaçons ce point dans la file « f−watershed », qui correspond à une file « en attente de mise à l'état *lpe* ». Nous devons également être très prudent lorsque nous modifions l'image de sortie ou de travail, car nous nous en servons lors de la propagation pour la détermination des points de « PL ». Une modification au cours du traitement de « PL » risquerait d'introduire des erreurs d'évaluation de l'état des points de « PL ». C'est pourquoi le traitement doit être fait de manière atomique. Nous pouvons voir cela comme si l'environnement dont dépendait « PL » était complètement figé, jusqu'à ce que « PL » soit traité en totalité.

Si le point central n'est pas *lpe* à la fin du parcours de voisinage (ligne 28), nous devons effectuer des tests supplémentaires pour tous les points dans l'état « empilé » et voisins de p. Ce cas est repris sur la figure 5.6. Les voisins empilés se trouvent dans la file « f−voisins ». Si le point central et le voisin ont des étiquettes même temporaires différentes, et appartiennent tous deux à « PL », alors ces deux points entrent en collision lors du traitement de « PL ». Ils appartiennent tous deux à la *lpe*, dont l'épaisseur est supérieure à 1 en ces points.



(a) Point d'intérêt (b) Parcours du voisinage (c) Point P_1 empilé de même label

(d) Point P_2 empilé de label différent (e) Point P_3 candidat (f) Point P_4 candidat

(g) Examen de « f−voisins » (h) Point central *lpe* (i) Point voisin marqué « lpe−temporaire »

Fig. 5.6.: Illustration de l'algorithme de *lpe* pour la détection des lignes épaisses. (a) Le point d'intérêt P_0 (centre) est marqué en rouge, les points entourés sont à l'état « empilé », alors que les autres sont à l'état « labélisé ». Les zones non marquées sont des points à l'état « candidat ». (b) Les voisins sont parcourus. Le centre reçoit le premier label qu'il rencontre (rouge entouré de bleu). (c) Un point P_1 empilé est rencontré. Son label temporaire n'est pas testé et il est placé directement dans la file « f−voisins ». (d) Un nouveau point empilé P_2, cette fois de label différent, est rencontré. Il est également placé directement dans la file « f−voisins ». (e)–(f) Deux points candidats P_3 et P_4 sont rencontrés, ils sont placés dans la file « f−candidat ». À ce stade, le point rouge central ne possède pas deux voisins de labels différents, mais peut potentiellement appartenir à une zone épaisse du fait que la file « f−voisins » n'est pas vide. (g) Les points de la file « f−voisins » sont examinés. P_1, de même label temporaire, ne créé pas de conflit. P_2 par contre possède un label temporaire différent. Le point central ainsi que P_2 appartiennent tous deux à une zone épaisse de la ligne de partage des eaux. (h) Le point central P_0 est marqué *lpe* et les candidats sont supprimés, car le point central est non-inondant. (i) P_0 est ensuite placé dans la file « f−watershed ». Puisque P_2 appartient également à la *lpe*, il est également placé dans « f−watershed ». L'état de P_2 passe à « lpe−temporaire » de manière à éviter un second test sur son voisinage.

Les étiquettes temporaires sont dans l'image de sortie « imSortie ». Nous pouvons déterminer si le point voisin v appartient également à « PL » en testant la valeur du relief en v. À cause du swamping, cette condition se résume à (ligne 31):

« le point voisin est à une hauteur inférieure ou égale à la priorité courante pr »

Cette condition est illustrée sur la figure 5.7. En effet, tous les points de « PL » étant traités à chaque étape, si le point voisin était à une hauteur supérieure, cela signifierait que son traitement n'aura lieu que plus tard (et donc qu'il n'est pas dans « PL »). Dans ce cas, le point courant p est marqué comme *lpe*. La relation donnée par la condition précédente est symétrique: en effet, la priorité courante est une condition globale sur l'ensemble des points et donc à « PL », et nous supposons le voisinage

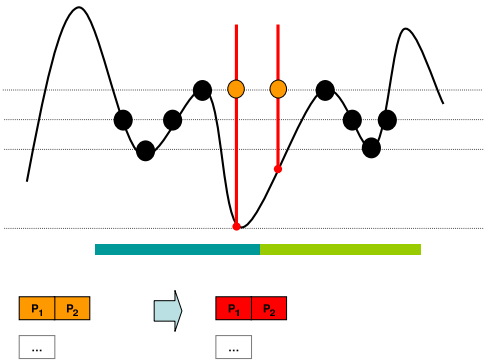


Fig. 5.7.: Illustration de la collision de deux points contenus dans la *fah*. Les deux points oranges sont dans la file d'attente, et en cours de traitement. Leur étiquette (bleue ou verte) est différente. Bien que leur hauteur respective sur le relief soit différente, il est possible de déterminer si ces points rentrent en collision épaisse à l'aide d'un moyen simple. La priorité courante étant connue, si l'un ou l'autre des points est à une hauteur inférieure ou égale à cette priorité signifie que leur traitement va effectivement avoir lieu en même temps. L'image de travail ne permet pas de faire cela directement. Ces deux points, en collision, sont ensuite marqués *lpe*.

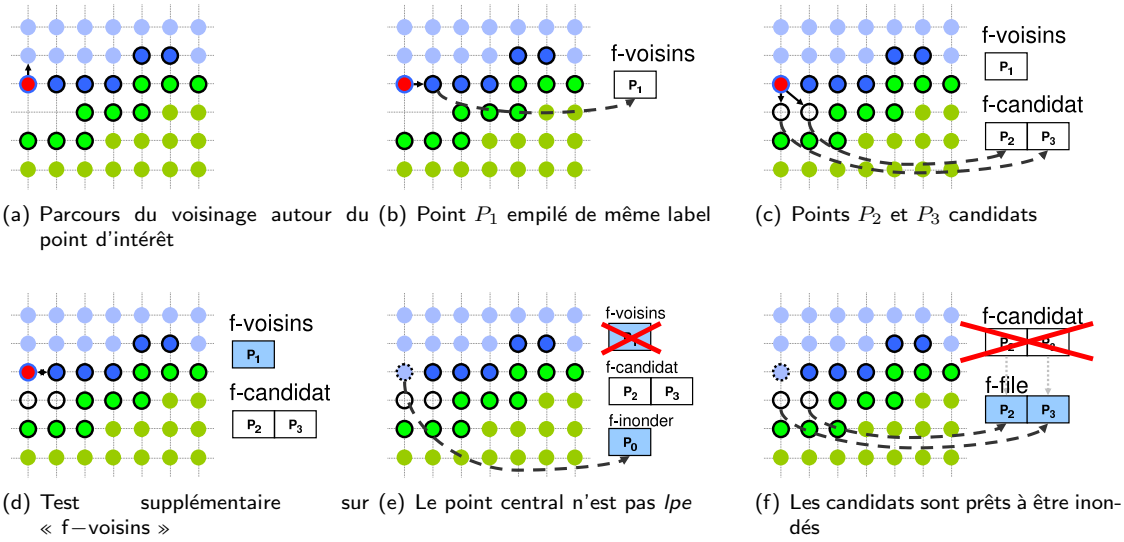
symétrique: $v \in \mathcal{N}_p \Leftrightarrow p \in \mathcal{N}_v$. La symétrie nous assure que cette condition n'est pas dépendante de l'ordre de traitement des points. Étant donné que nous avons découvert le voisin v comme étant *lpe*, nous pouvons placer v parmi la liste des points *lpe* dans la file d'attente « f−watershed ». Nous évitons également un second traitement en le marquant comme un point « lpe−temporaire ». C'est la seule fois où nous modifions l'état de l'image de travail, cet état est considéré comme un état empilé lors du parcours du voisinage (ligne 24).

Contrairement au cas précédent de découverte de point *lpe*, où il était possible de sortir de la boucle dès la détermination de l'état *lpe*, tous les voisins v à l'état empilé (dans « f−voisins ») doivent subir ce traitement. Pour prévenir le traitement double des points « lpe−temporaire », un test en amont (ligne 9) a été ajouté. Afin d'éviter la double insertion dans la file des points *lpe*, nous n'ajoutons un point central p à la file « f−watershed » que si p est labélisé (ligne 44) ce qui n'est pas le cas pour les points vérifiant le test amont. Enfin, puisque la relation de *lpe* épaisse est anti-symétrique (donc non transitive), le test complet du voisin v nouvellement *lpe* est inutile: si v est à son tour siège d'une *lpe* épaisse, ce même test à partir d'autres voisins permet de déduire la même information (relation deux à deux et les voisins de v seront également traités). Enfin, puisque nous parcourons tous le voisinage de p, il est inutile de le marquer également comme « lpe−temporaire ».

Dans le cas où le point central p n'est pas *lpe* (cf. figure 5.8), alors les voisins candidats peuvent être placés dans une seconde file intermédiaire « f− file », qui seront ensuite insérés dans la file globale. Encore une fois, puisque nous ne pouvons modifier les espaces images, nous plaçons dans « f− file » le voisin v, et son étiquette « label −courant ». Le point p est également inséré dans une file intermédiaire « f−inonder », permettant de le labéliser définitivement à la fin du traitement de « PR ».

Algorithme 5.3 : *lpe* épaisse isotrope à file d'attente hiérarchique - Propagation

```
1  ▶ imTravail. États possibles des points: candidat, labélisé, lpe, empilé, lpe-temporaire
2  ▶ imSortie. États possibles des points: label, lpe
3  • Propagation:
4  while fah-globale ≠ ∅ do
5      PL ← file de priorité maximale de fah-globale
6      pr ← priorité maximale de fah-globale
7      forall p ∈ PL do
8          est-labelisé = est-lpe = faux
9          if imTravail(p) = lpe-temporaire then est-lpe = vrai        // condition de lpe épaisse, cf. texte
10         else
11             f-candidat, f-voisins ← ∅
12             forall v ∈ N_p \ p do
13                 if imTravail(v) = candidat then f-candidat ← ajouter v        // nouveaux candidats
14                 else if imTravail(v) = labélisé then                // points labélisés et voisins du centre
15                     if est-labelisé then                            // le centre est déjà labélisé
16                         if imSortie(v) ≠ label-courant then         // conflit de labels
17                             est-lpe = vrai                          // le point est lpe
18                             break                                   // fin de parcours du voisinage
19
20                     else
21                         est-labelisé = vrai
22                         label-courant = imSortie(v)                 // le centre acquiert son label
23
24                 else if (imTravail(v) = empilé) ∨ (imTravail(v) = lpe-temporaire) then
25                     f-voisins ← ajouter v                           // point dans la fah (de conflit potentiel)
26
27             if ¬ est-lpe then
28                 ▶ Le point central est nécessairement labélisé, mais peut être lpe
29                 forall v ∈ f-voisins do
30                     if (imSortie(v) ≠ label-courant) ∧ (imSortie(v) ≠ fond) then
31                         if pr ≥ imRelief(v) then
32                             est-lpe = vrai
33                             if imTravail(v) ≠ lpe-temporaire then
34                                 imTravail(v) = lpe-temporaire
35                                 f-watershed ← ajouter v
36
37
38
39         if ¬ est-lpe then
40             f-inonder ← ajouter p
41             forall v ∈ f-candidat do
42                 f-file ← ajouter paire(v,label-courant)
43         else
44             if est-labelisé then f-watershed ← ajouter p
45     Suite dans l'algorithme 5.4
```



(a) Parcours du voisinage autour du point d'intérêt

(b) Point P_1 empilé de même label

(c) Points P_2 et P_3 candidats

(d) Test supplémentaire sur « f−voisins »

(e) Le point central n'est pas *Ipe*

(f) Les candidats sont prêts à être inondés

Fig. 5.8.: Illustration de l'algorithme de *Ipe* pour la détection d'un point « normal ». (a) De la même manière que sur la figure 5.6, le voisinage du point central (en rouge) est parcouru. Ce dernier acquiert le premier label (bleu) qu'il rencontre lors du parcours. (b) Un point voisin P_1 à l'état empilé est rencontré, il est placé dans « f−voisins ». (c) Des points candidats P_2 et P_3 sont rencontrés, et placés dans « f−candidat ». (d) Le point central n'est toujours pas *Ipe*, mais « f−voisins » n'est pas vide. Les points de cette file sont alors testés, mais il n'y a pas de collision avec le point central. (e) Le point central n'est finalement pas *Ipe*: il peut être labélisé fermement à « bleu ». Puisqu'on ne peut mettre à jour les images, ce point ainsi que son label (bleu) sont placés dans « f−inonder ». (f) Le point central n'étant pas *Ipe*, les candidats découverts peuvent être inondés. Encore une fois, puisque nous ne pouvons écrire leur label temporaire (bleu) dans les images, ces points ainsi que leur label respectif sont placés dans « f−file ».

Description de l'algorithme 5.4 de fin de la propagation: Lorsque tous les points de « PL » ont été traités (ligne 1), les contenus des différentes files d'attente peuvent être placés dans les espaces images. Puisqu'il est possible d'avoir des doublons dans ces files, des tests de blocage supplémentaires ont été mis en place.

- Le test pour les doublons de la file « f−watershed » n'est pas important (ligne 2).
- Le label temporaire des points non *lpe* étant déjà dans « imSortie », la mise à jour de « imTravail » seule suffit (ligne 6).
- Puisque la relation de test de collision est symétrique, il n'est pas possible qu'un point possède des voisins candidats (insérés dans la file de candidats « f− file »), sans qu'il ait été déterminé *lpe*.

Les points de « PL » peuvent être détruits de la *fah*. Cette destruction aurait pu être effectuée à la fin de la boucle de traitement de « PL » (ligne 1): toutes les informations nécessaires aux mises à jour étaient déjà dans les files d'attente intermédiaires.

Algorithme 5.4 : *lpe* épaisse isotrope à file d'attente hiérarchique - Propagation - suite

```
1  forall p ∈ f-watershed do                    // Mise à jour des points ayant reçu une modification
2      if imTravail(p) ≠ lpe then
3          imTravail(p) = lpe
4          imSortie(p) = lpe
5
6  forall p ∈ f-inonder do  imTravail(p) = traité
7  forall v, label-courant ∈ f-file do
8      if imTravail(v) ≠ empilé then
9          imTravail(v) = empilé
10         imSortie(v) = label-courant
11         fah-globale ← ajouter v à la priorité (pr ∨ imRelief (v))
12
13 PL ← ∅
14 ▶ Fin du traitement de PL, retour à la ligne 4 de l'alg. 5.3
```

Remarques Les tests d'appartenance à « PL » de l'algorithme 5.3 peuvent être en partie évités. En effet, si nous parcourons deux fois « PL », et ne faisons que marquer les points de « PL » lors du premier parcours, nous pouvons déduire facilement les points épais de la *lpe*. Nous avons jugé la solution proposée plus astucieuse.

Nous allons dans la suite montrer quelques résultats de l'algorithme dans des espaces particuliers, sur des images en n dimensions ou des images avec des reliefs vectoriels. Ces développements sont possibles grâce au cadre informatique proposé dans le chapitre 2. L'objectif premier est de montrer les possibilités algorithmiques de la méta-programmation, mais également de proposer de nouvelles solutions de traitement de données afin d'étendre le champ d'application de la *lpe*.

5.2.3. Inondation en n dimensions

Bien que souvent mentionnée, nous ne connaissons pas d'exemple d'algorithme de *lpe* capable de travailler en dimension quelconque (n dimensions). Dans cette perspective, le cadre décrit au chapitre 2 permet d'appliquer l'algorithme précédent de *lpe* en n'importe quelle dimension. Nous détaillons ici succinctement dans quelle mesure les moyens offerts par notre cadre informatique permettent à l'algorithme d'évoluer en dimension quelconque.

Nous soulignons le fait que l'algorithme proposé est complètement indépendant des structures de points de l'image. En effet, dans les algorithmes 5.2, 5.3 et 5.4, les points p manipulés sont des

déplacements - *offset* - par rapport au début de l'image. Les algorithmes ne manipulent donc pas directement des coordonnées dans l'espace $\mathbf{E}$ des points de l'image, mais leur représentation sous forme d'entiers.
La structure de voisinage est entièrement cachée dans l'objet de voisinage, en paramètre de l'algorithme. Nous utilisons ce voisinage par exemple à la ligne 12 de l'algorithme 5.3: l'action « $\forall v \in \mathcal{N}_p \setminus p$ » est transcrite de la manière suivante:

```
1       neighborhood.setCenter(p);
2       typename neighborhood_type::iterator n_it = neighborhood.begin(), n_it_end = neighborhood.end();
3       for(; n_it != n_it_end; ++n_it)
4       {
5           v = n_it.Offset();
6           if(v == p) continue;
7
8           // ... suite de l'algorithme
9       }
```

Ici, « neighborhood » représente le voisinage $\mathcal{N}$. Avant tout accès au voisinage, il faut le centrer à l'aide de la méthode « setCenter ».

« neighborhood_type » est le type du voisinage. C'est un champ de type (cf. chapitre 2) qui permet de déduire le type de ses itérateurs. Les itérateurs pointent respectivement sur le début et la fin du sous-ensemble que constituent les points du voisinage. Enfin la méthode « Offset » de l'itérateur permet d'accéder au déplacement du point voisin v.
Comme nous l'avons mentionné dans le chapitre 2, l'offset est une représentation compacte d'une coordonnée exprimée en dimension quelconque. Les positions des voisins sont gérées par la structure de voisinage, et non par l'algorithme. Ce dernier manipule les points sans être lié ni à leur implémentation, ni à leur représentation.

Comme l'algorithme fonctionne uniquement par propagation, c'est à dire de *proche en proche* par voisinage, à **aucun** moment dans l'algorithme une manipulation explicite de coordonnée n'est effectuée. Les coordonnées sont *transportées* de la structure de voisinage $\mathcal{N}$ vers la structure de *fah*, et vice versa. L'algorithme n'a pas explicitement besoin des coordonnées pour adapter ou modifier son comportement ou son déroulement.
À partir de ces remarques, nous disposons des outils nécessaires pour appliquer l'algorithme en dimension quelconque. Il en est de même des algorithmes dérivés tel que l'algorithme des cascades.

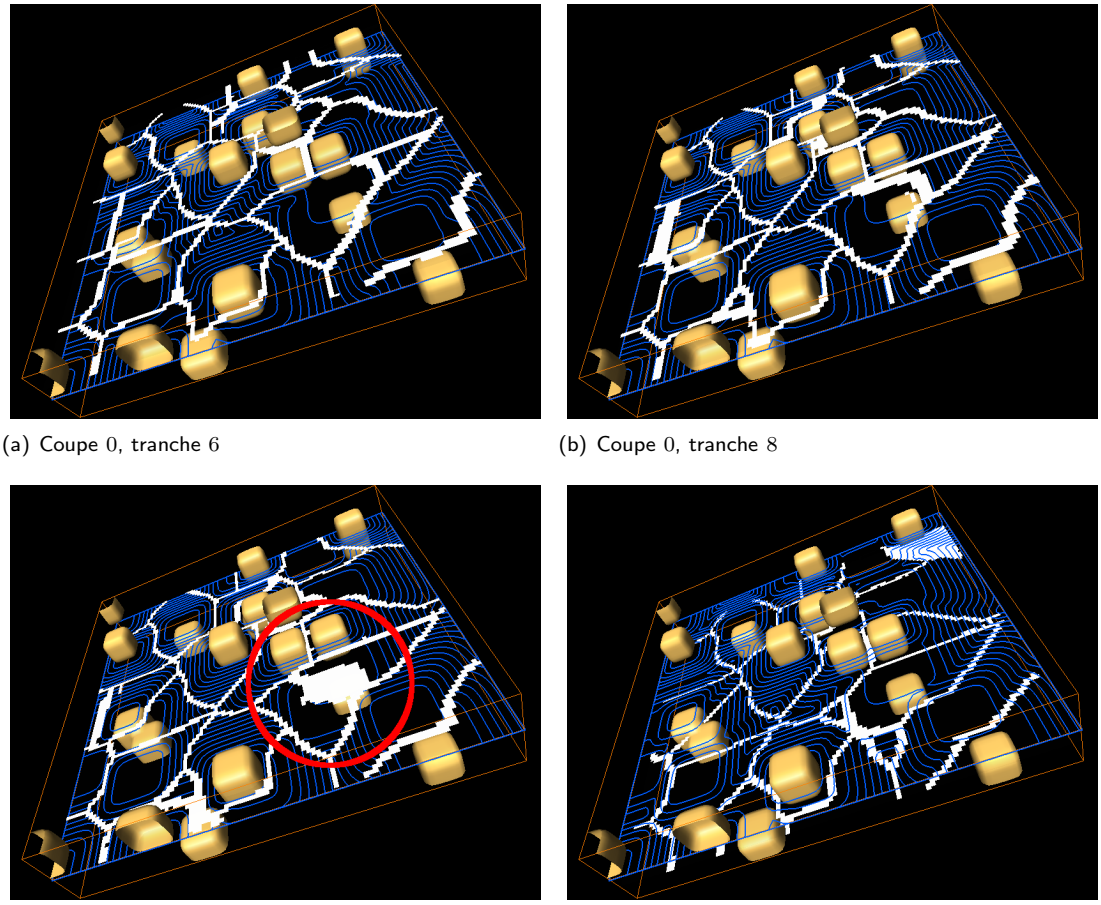
Exemple de ligne de partage des eaux en $4D$: Nous reprenons ici les résultats des fonctions distances exactes présentées dans le chapitre 3.

Les figures 5.9 et 5.10 présentent le résultat de la *lpe* épaisse appliquée à l'image de la fonction distance exacte ℓ^5 (figure 3.6 du chapitre 3, page 64). La taille de l'image est $100 \times 100 \times 20 \times 10$. Nous avons tracé des isosurfaces pour plus de clarté, mais la *lpe* est appliquée sur l'image des distances, et non à ces isosurfaces.

Nous appellerons *coupe* une section $3D$, par suppression de la quatrième dimension (de taille 10). Nous appellerons *tranche* une section $2D$, perpendiculairement au troisième axe (de taille 20). Nous allons examiner le comportement de la *lpe* sur deux coupes, c'est à dire selon la dimension 4, illustrant l'épaisseur de la *lpe*.

Nous nous intéressons aux lieux où la ligne de partage des eaux est épaisse. Une épaisseur particulièrement importante est visible sur la tranche 9 de la coupe 0 (figure 5.9(c), zone à l'intérieur du cercle rouge). Quelques points de la *lpe* semblent déconnectés du reste sur la coupe (tranche 8, même lieu sur la figure 5.9(b)), mais ils sont en fait connexes à la tranche adjacente 9: puisque la zone de partage est importante, une projection selon un axe déconnecte naturellement certains points. Par ailleurs, un même bassin versant peut apparaître sous plusieurs composantes connexes sur une tranche: c'est le cas, dans la même zone, sur la tranche 13 (à rapprocher du bassin versant adjacent de la tranche 9).

À présent, sur la coupe 6 en 5.10, nous remarquons toujours au même lieu, encore une épaisseur de la *lpe*. Cette fois-ci, cette épaisseur sépare deux boules dont l'une n'était pas visible dans la coupe 0.



(a) Coupe 0, tranche 6

(b) Coupe 0, tranche 8

(c) Coupe 0, tranche 9

(d) Coupe 0, tranche 13

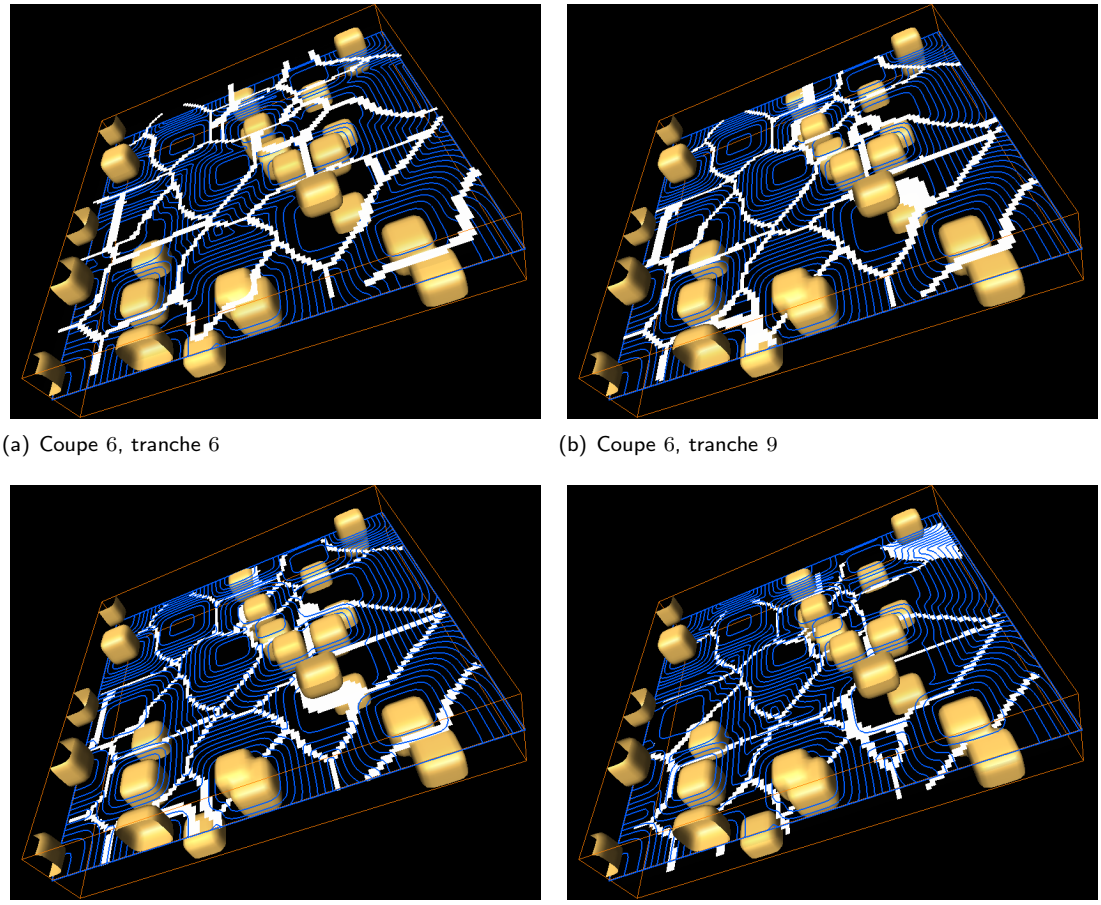
Fig. 5.9.: Ligne de partage des eaux en $4D$ - Coupe 0. Les isosurfaces à d constant sont en jaune (boules selon la métrique ℓ^5 de taille constante). Les lignes bleues symbolisent les isolignes dans un plan $2D$. La *lpe* est surimprimée en blanc. Le cercle rouge met l'accent sur une zone d'épaisseur $4D$.

Finalement, nous constatons que la *lpe* est en forme « d'oblique » suivant les coupes (tranches 9 et 10 respectivement aux figures 5.10(b) et 5.10(c)).

5.2.4. Inondation dans $\mathbb{R}$

L'objectif de cette section n'est pas de proposer un nouvel algorithme, mais de décrire comment l'algorithme sans biais proposé bénéficie de l'architecture logicielle détaillée dans le chapitre 2, de manière à être appliqué sur des images de type « virgule flottante ». À notre connaissance, ce genre de travaux, c'est à dire un algorithme de *lpe* sur une surface topographique en virgule flottante, et à base de *fah*, n'a pas été entrepris dans la littérature. Par exemple, les auteurs de [BDB06] soulignent la « difficulté » de la représentation flottante, et la contournent en repassant à une représentation entière. Nous proposons de rester en représentation « fine », c'est à dire en virgule flottante.

L'intérêt de cette représentation est justifié par le fait qu'un nombre croissant d'applications utilise des images de précision également croissante. L'exemple de la couleur est un cas typique: la couleur est traitée par exemple par une fonction de distance, définissant des gradients morphologiques généralisés (cf. définition 4.2 page 110) dont le résultat est en virgule flottante. Disposer d'un algorithme capable de fonctionner avec cette précision a pour avantage la conservation de précision sur l'ensemble du



(a) Coupe 6, tranche 6

(b) Coupe 6, tranche 9

(c) Coupe 6, tranche 10

(d) Coupe 6, tranche 13

Fig. 5.10.: Ligne de partage des eaux en $4D$ - Coupe 6. Même légende qu'en figure 5.9.

traitement, et certaines conversions d'espace deviennent ainsi évitables.

Nous avons discuté en 2.2.6 (page 33) des files d'attente hiérarchiques, et nous reportons le lecteur à cette partie pour les détails de cette structure. Dans la la partie concernée, nous avons bien précisé que cette structure nécessitait une fonction d'ordre sur l'espace de « *priorités* ». Nous dirons simplement qu'il n'est pas besoin d'écrire plusieurs algorithmes selon les types, mais d'utiliser une *stratégie* d'utilisation de *fah*. La stratégie n'est rien d'autre qu'une structure informatique de contrôle externe: l'algorithme délègue le choix de la *fah* à cette structure. Ceci permet entre autres d'avoir un point d'accès unique à la structure définitive de la *fah*- point d'accès qui est utilisé par tous les algorithmes - et évite la dépendance directe de l'algorithme par rapport à une implémentation particulière de *fah*. Le type de la structure de *fah* est donné par:

```
1 typedef typename pq_strategy<C, T, order_operator>::PQ priority_queue;
2 priority_queue fah_globale; /* fah utilisée dans l'algorithme d'inondation */
```

où « C » est le type des points du relief, « T » est le type « offset »- *coordonnée universelle* - et enfin « order_operator » est l'opérateur d'ordre sur les valeurs du relief, à savoir « order_operator $\equiv <_C$ ».

D'après l'algorithme sans biais, l'ordre lié au relief sur lequel la propagation est effectuée apparaît explicitement deux fois:

1. Dans l'algorithme 5.3 en ligne 31 pour la détection des collisions entre points de la *fah*.

2. Dans l'algorithme 5.4 ligne 11 pour le calcul du supremum. Ceci sert en fait pour le swamping lors de l'ajout d'un nouveau point dans la *fah*, car un point inséré dans la *fah* à une priorité inférieure ou égale à celle du point inondant.

Par ailleurs, la structure de files d'attente hiérarchiques est également initialisée avec la relation d'ordre donnée à l'algorithme. Cependant, tous les mécanismes de tri sont cachés dans la structure de *fah*. Ainsi l'implémentation algorithmique n'est pas sensible à la nature des données du relief inondé, ce qui nous permet de l'utiliser en l'état sur des données plus complexes. Nous l'avons utilisé sur des images en précision à virgule flottante. Nous pouvons à présent l'utiliser sur des reliefs dont les points sont des données vectorielles. Pour cela, nous allons utiliser une relation d'ordre lexicographique, ce qui est l'objet de la section suivante.

5.2.5. Inondation et treillis lexicographiques

La structure d'ordre apparaît naturellement dans l'algorithme de ligne de partage des eaux, par la notion de « relief » topographique: moins l'altitude des points est élevée, plus leur priorité est importante. La hiérarchie liée à la topographie est implicitement gérée par la file d'attente hiérarchique. Dans l'algorithme, et hormis l'utilisation de la file d'attente hiérarchique, la structure d'ordre intervient lors du test de collision entre deux points empilés et voisins (cf. ligne 31 de l'algorithme 5.3). Selon les remarques que nous avons émises lors de la description de l'algorithme (cf. fin du §5.2.2 page 198), ce test de collision peut être évité en marquant les points en cours de traitement, sur une même priorité et un même front, avant la détermination complète de leur état.

Selon les développements des chapitres 2 et 4, nous avons à notre disposition tous les outils nécessaires pour une application de la *lpe* sur des treillis lexicographiques. Il est néanmoins légitime d'ouvrir une parenthèse sur l'intérêt que suscitent de tels treillis. À l'instar des nombreuses remarques faites précédemment, les treillis lexicographiques permettent d'avoir une relation d'ordre plus fine lorsqu'une égalité a lieu entre deux éléments, selon l'ordre classique. Pour un algorithme comme la *lpe*, cela se traduit par des plateaux plus petits, puisque les occurrences d'égalité sont nécessairement moins nombreuses (ordre plus fin).

Le relief utilisé est alors un relief de type vectoriel. Nous utilisons en fait un gradient morphologique lexicographique (cf. §4.2.1.3 page 109) pour le calcul du contraste entre points voisins. Dans le chapitre 4 où nous avions discuté de tels gradients, nous utilisions en fait le module du gradient, défini trivialement à partir d'une métrique couleur. Cependant, le type de relief qui nous intéresse ici est vectoriel. Nous créons alors le gradient vectoriel par recombinaison des canaux du gradient morphologique lexicographique. Plus précisément, soit une image $\mathcal{I}$ dans l'espace couleur HLS; notons $\prec_{\mathcal{R}}$ la relation d'ordre utilisée, $\Delta_{\mathcal{I}} = \delta_{\prec_{\mathcal{R}}}\mathcal{I}$ et $E_{\mathcal{I}} = \epsilon_{\prec_{\mathcal{R}}}\mathcal{I}$ respectivement le dilaté et l'érodé de $\mathcal{I}$ selon la relation $\prec_{\mathcal{R}}$, et enfin l'application « $\star \mapsto \star \cdot \vec{v}$ » est la projection de $\star$ sur la base donnée par le vecteur $\vec{v}$ (par analogie au produit scalaire classique). Le gradient vectoriel est défini de la manière suivante:

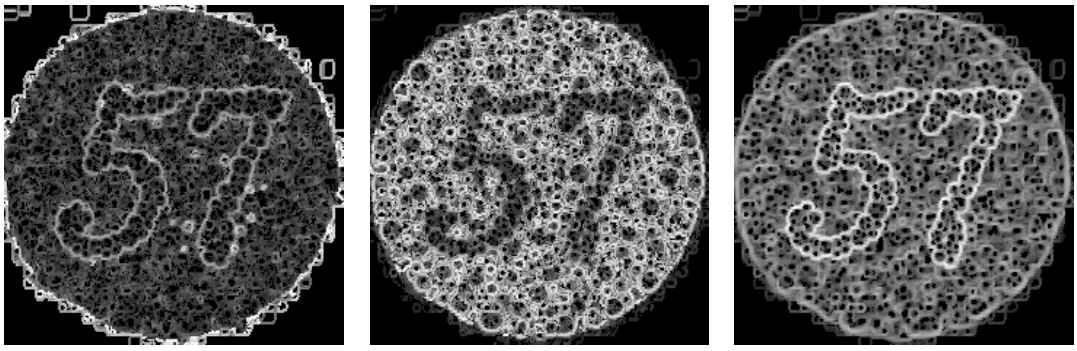
$$\nabla_{vect,\prec} : \mathcal{I} \mapsto \begin{bmatrix} |\Delta_{\mathcal{I}} \cdot \vec{h} & \angle & E_{\mathcal{I}} \cdot \vec{h}| \\ |\Delta_{\mathcal{I}} \cdot \vec{l} & - & E_{\mathcal{I}} \cdot \vec{l}| \\ |\Delta_{\mathcal{I}} \cdot \vec{s} & - & E_{\mathcal{I}} \cdot \vec{s}| \end{bmatrix} \tag{5.6}$$

où h, l et s sont donc respectivement la composante teinte, luminance et saturation. $(\star, \bullet) \mapsto \star\angle\bullet$ est la différence angulaire que nous avons vue dans l'équation 4.4 page 107. Observons ce gradient sur un exemple simple, donné par la figure 5.11.

Nous avons choisi de travailler sur une image modifiée de l'image originale: compte tenu de la nature des disjonctions entre les différentes pastilles de couleur, nous effectuons une dilatation de l'image originale à l'aide d'une relation lexicographique $S \downarrow L \uparrow H \downarrow$ [(xi)]. Nous calculons ensuite l'érodé et le dilaté lexicographique de cette image prise comme référence de traitement, selon la relation d'ordre

[(xi)] ordres naturels pour les canaux de saturation et teinte, ordre inverse pour la luminance. La dilatation sur le treillis défini par cette relation prendra dans un voisinage le supremum de la saturation, puis si égalité l'infimum de la luminance, etc..

(a) Image originale (b) Dilaté de (a) (c) Dilaté de (b) (d) Érodé de (b) (e) Gradient lexicographique dans RGB



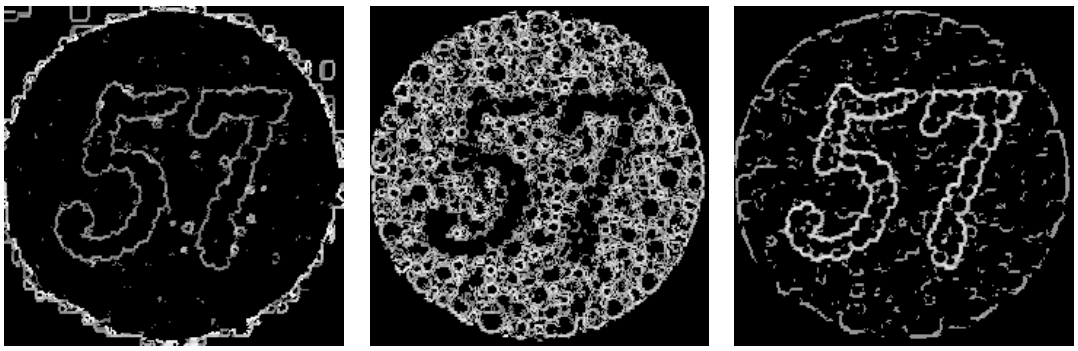
(f) Projection de (e) sur H (g) Projection de (e) sur L (h) Projection de (e) sur S

Fig. 5.11.: Illustration du gradient vectoriel utilisé pour la *lpe* lexicographique. (b) Nous dilatons l'image originale de manière à joindre les pastilles de couleur. L'ordre utilisé est $S \downarrow L \uparrow H \downarrow$. (c)–(d) Respectivement les images dilatée et érodée de (b), avec une nouvelle relation d'ordre $S \downarrow L \downarrow H \downarrow$. (e) Le gradient lexicographique, transformé dans l'espace RGB et difficile à interpréter. (f)–(h) Les projections du gradient (e) sur les différents canaux.

$S \downarrow L \downarrow H \downarrow$. Ces traitements sont illustrés sur les figures 5.11(b) à 5.11(d). Le gradient vectoriel de la figure 5.11(e) et construit selon l'équation 5.6, est d'une interprétation difficile. La projection de ce gradient sur chaque canal (figures 5.11(f)–5.11(h)) fournit toutefois des indications sur la pertinence de l'information contenue dans chaque canal.

Résultats de la lpe lexicographique Sur les images naturelles, l'aspect lexicographique est absolument négligeable. Cela provient du fait que le nombre de points voisins et identiques est pratiquement nul. Pour illustrer l'intérêt de l'approche lexicographique, nous allons dégrader le gradient vectoriel précédent.

Nous pouvons justifier cela par le coût en mémoire important des images couleur, particulièrement si celles-ci sont codées en précision à virgule flottante. Nous réduisons la précision de chaque canal, de manière à pouvoir coder chaque point sur un entier compris entre 0 et 10 par exemple. Ce procédé dégrade fortement les gradients comme le montre la figure 5.12.



(a) Teinte dégradée (b) Luminance dégradée (c) Saturation dégradée

Fig. 5.12.: Dégradation du gradient. De nombreux « trous » apparaîssent et provoquerons naturellement des fuites de la *lpe*.

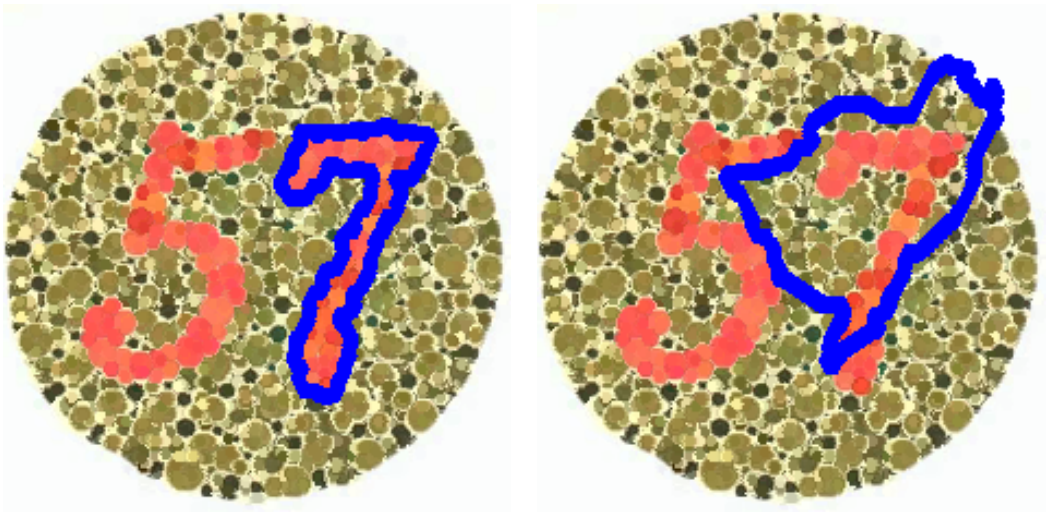
Dans l'exemple suivant, nous avons utilisé deux marqueurs: le premier marque un point unique à l'intérieur du chiffre « 7 », le second est en fait le bord de l'image. Nous appliquons l'algorithme de *lpe* sur le gradient dégradé avec ces deux marqueurs, et en utilisant deux relations d'ordre. La première relation utilise les trois canaux dans l'ordre $H \downarrow S \downarrow L \downarrow$, la seconde n'utilise que le canal de teinte. Le résultat est donné en figure 5.13.

Nous remarquons sur cette figure que la *lpe* tire profit de la relation d'ordre lexicographique (rappelons que nous avons marqué le chiffre « 7 ») sur un gradient que nous avons délibérément dégradé, alors que l'utilisation d'un unique canal conduit à une fuite de la propagation.

5.2.6. Conclusions sur la ligne de partage des eaux

Dans les sections §5.1 et §5.2, nous avons vu quelques développements théoriques de la ligne de partage des eaux, puis son implémentation algorithmique. Nous avons repris l'algorithme de référence existant proposé par Meyer [Mey91]. Nous avons d'une part explicité les biais produits par cet algorithme, par rapport à la définition de la *lpe*, d'autre part nous avons souligné une erreur importante couramment commise. Nous avons ensuite corrigé ces biais et avons proposé un algorithme de *lpe* épaisse, basé sur les files d'attente hiérarchiques.

Grâce au contexte informatique que nous avons développé, nous avons démontré que nous pouvions appliquer cet algorithme sur des images en dimension 4. L'algorithme est en fait indépendant de la représentation des coordonnées, et il est possible de l'utiliser sur des images de dimension quelconque. Nous avons pris une image de distance produite par notre algorithme, proposé au §3.2.



(a) Relation $H \downarrow S \downarrow L \downarrow$ (b) Canal de teinte uniquement

Fig. 5.13.: Application de la *lpe* lexicographique sur le gradient 5.12. La ligne de partage des eaux, selon les marqueurs décrits dans le texte, est symbolisée en bleu.

Toujours grâce au contexte informatique, nous pouvons appliquer notre algorithme sur des reliefs à précision flottante, ou des reliefs *vectoriels*. Ce point semble également être une nouveauté. En effet, la structure de file d'attente hiérarchique ne dépend que de la relation d'ordre sur l'espace des points du relief. Une fois une relation d'ordre définie, l'algorithme et les structures sous-jacentes utilisent cette relation sans aucune modification algorithmique.

5.3 Injection d'information a priori

L'algorithme de *lpe* ne fait aucune distinction entre les différents labels. Or, on est souvent amené à séparer des éléments de l'image sémantiquement différents. Le placement des marqueurs de segmentation suit d'ailleurs cette démarche: des marqueurs sont extraits selon les propriétés propres aux objets que l'on veut séparer, ils marquent ces objets et la partition finale contient au moins ces points, pour lesquels l'incertitude est faible. La ligne de partage des eaux classique empêche le prolongement de cette démarche dans le processus de segmentation, dans la mesure où elle ne permet pas de poursuivre l'utilisation de certaines propriétés des objets lors de la propagation. Cette section adresse ce problème.

5.3.1. Problème ouvert

Les moyens pour parvenir à l'injection d'information *a priori* dans le processus d'inondation sont très nombreux.

Le point de vue adopté jusqu'ici concerne essentiellement le post-traitement de la partition obtenue à l'aide d'une *lpe* « aveugle », non-supervisée à partir des minima locaux. La partition obtenue peut être représentée sous forme de graphe d'adjacence. La valuation des nœuds dépend de l'application. Le graphe est ensuite simplifié par fusion de nœuds: c'est à ce moment qu'intervient une connaissance externe. Cette méthodologie est utilisée par exemple dans [AL03, ZT01, Gom01, MI96]. Les possibilités offertes par la fusion de régions sont encore loin d'avoir été explorées, et des travaux sont en cours de réalisation, notamment ceux sur la structure mathématique de graphe et le traitement spectral (cf. Lerallut [Ler06]).

Il est discutable d'affirmer qu'une méthode est meilleure qu'une autre. Dans ce qui suit, nous allons utiliser une toute autre méthode: nous allons modifier l'algorithme original de *lpe* pour y inclure une connaissance *a priori*.

5.3.2. Nature de la connaissance a priori

Le terme *connaissance a priori* était jusqu'ici assez vague. Voyons ci-après les types d'information qu'il est possible de trouver:

Information de type point l'information concerne les valeurs prises par les points. La conformité de la valeur de chaque nouveau point est évaluée lors de son insertion dans une des régions, pendant le processus de propagation. L'évaluation est une fonction du point seul.

Information de type région l'information concerne non plus le point, mais une contrainte sur la totalité des points de la zone. Il s'agit donc d'une contrainte globale sur la région inondée. Une contrainte possible serait par exemple l'homogénéité de la région, par rapport à une mesure de variance.

Information de type géométrique Il s'agit de la forme de la région *désirée*. De nombreuses techniques, notamment les techniques de contours actifs géodésiques [PD98, GKRR01], snake [XP98], ligne de niveaux [(xii)] [Set96, TO03], EDP [Moi97] abordent ce problème. Le processus d'inondation est déterministe, mais la prévision du résultat sans la réalisation du processus, est à notre connaissance impossible. Par exemple, si nous souhaitons obtenir une ellipse, cela signifierait que les étapes successives de l'inondation seraient contraintes à des ellipses. Aussi, les contraintes géométriques s'appliquent généralement sur d'autres critères. Il est courant de voir une contrainte de régularité du front de propagation, pour ne citer que cet exemple. Nous différons ce point à la section suivante, où il sera question d'inondation à contrainte locale.

Puisque les connaissances *a priori* sont issues de paramètres statistiques, nous confondrons dans la suite les termes *labels* et *classe*.

[(xii)] de l'anglais « *level sets* »

5.3.3. Méthode proposée

Nous allons travailler sur la priorité des points insérés dans la file d'attente. Nous justifions cette démarche par une analogie avec l'approche classique: la priorité traduit la ressemblance d'un point nouvellement inondé (labélisé) avec la priorité courante pr. Un point d'une hauteur égale à pr ressemble donc aux points en cours de traitement. Le traitement d'un point à une hauteur supérieure est reporté jusqu'à ce qu'il y ait « ressemblance » entre la pile de traitement courant (la priorité pr) et la hauteur du point.

Notons Γ le front de propagation. Pour un algorithme tel que la ligne de partage des eaux, il s'agit des points n'appartenant pas encore à un bassin versant, et qui codent l'extérieur immédiat du bassin [(xiii)]. Chacun des points du front contient à la fois une coordonnée et une priorité associée. Dans le cas classique, cette priorité correspond à la hauteur du relief à la coordonnée considérée. Dans notre cas, nous modifions la manière de calculer la priorité d'insertion dans la *fah*: nous ajoutons au terme de hauteur un terme de mesure, qui est fonction de la ressemblance entre le point courant et la connaissance *a priori*.

Cette mesure étant dépendante de la région, il est possible de dire que chaque classe se propage sur son propre relief. Le processus est toutefois légèrement plus complexe puisqu'il permet l'interaction de toutes les classes sur une même partition. Enfin, si le nombre de classes est peu élevé, il est alors - humainement - possible d'associer à chacune d'elle une fonction de mesure différente.

Formulation À partir du cadre théorique, la formulation la plus simple pour exprimer ce problème est celui des contours actifs (cf. §5.1.2.3 page 181). La formulation générale, selon la modification proposée, prend la forme suivante:

$$\frac{\partial \Gamma}{\partial t} = \frac{c_0}{\|\nabla f\| + \alpha \cdot \mu_\Gamma} \cdot \vec{\mathbf{n}} \tag{5.7}$$

Rappelons que le vecteur $\vec{\mathbf{n}}$ est la normale à Γ dirigée vers l'extérieur. μ_Γ est une **mesure** sur le contour Γ, α est une constante pondérant l'influence de cette mesure, et c_0 est une constante quelconque que nous fixerons définitivement à 1. Bien qu'elle ne soit pas complètement rigoureuse [(xiv)], nous voyons rapidement à l'aide de cette formulation les grandes différences par rapport à la *lpe* classique.

La mesure μ_Γ peut être dépendante de l'intérieur de Γ [(xv)]. Par ailleurs, cette formulation est dite *explicite*, dans la mesure où l'expression de $\frac{\partial \Gamma}{\partial t}$ dépend explicitement de Γ. Dans le cas particulier des informations de type *point*, le problème est grandement simplifié puisque μ_Γ est indépendant de l'intérieur de Γ et même du contour. En effet, l'ajout d'un point à la région délimitée par Γ ne change pas la mesure d'appartenance d'un point à la classe considérée. Ce n'est par contre pas le cas pour les informations de type *région*, où la mesure évolue en fonction du temps.

La forme de μ_Γ est *a priori* quelconque, la seule contrainte - de cohérence - étant que μ_Γ décroisse avec la dissemblance du point par rapport à la connaissance. Enfin, on peut considérer $\mu_\Gamma > 0$, de manière à éviter un cas de division par 0 (sur les plateaux).

Remarque Contrairement aux problèmes de classification, l'inondation contrainte n'est pas équivalente à une partition de $\mathbf{F}$ en plusieurs classes à partir de la mesure μ_Γ. Cela est bien mis en évidence dans l'équation 5.7. Intuitivement, d'autres notions apparaissent, telle la connexité du bassin versant. À notre connaissance, il n'est pas possible de résoudre la partition sans effectuer le processus de propagation, que cela soit dans le cas de connaissance de point ou de région.

[(xiii)] Par rapport aux algorithmes présentés, ce serait les points à l'état « empilé », c'est à dire dans la file d'attente hiérarchique.

[(xiv)] voir à ce sujet la partie §5.1.2.3 concernée

[(xv)] c'est à dire la région délimitée par Γ.

Dans les deux paragraphes suivants, nous allons aborder les deux premiers types de connaissance. Rappelons que la première est une connaissance *a priori* sur l'appartenance des points à une classe, la deuxième traduit une connaissance *a priori* sur la région.

5.3.4. Information de type « point »

Soit $\mathbf{F}$ l'espace des valeurs des points (pixels de l'image). L'hypothèse est que nous avons une connaissance *a priori* sur une classe de points à valeur dans $\mathbf{F}$. Soit $\mathcal{C}$ cette classe. Inclure cette connaissance dans le processus d'inondation signifie:

1. pour un nouveau point c, évaluer l'inférence $c \in \mathcal{C}$.
2. utiliser cette évaluation dans le processus d'inondation.

L'évaluation est généralement donnée par une mesure de probabilité, $P(c \in \mathcal{C})$, ce que nous avons nommé précédemment par mesure μ_Γ. Nous allons voir dans la suite les modifications qu'une telle contrainte engendre dans l'algorithme de *lpe* initial.

Algorithme L'algorithme est pratiquement le même que pour le cas classique, à un appel de fonction près. Par rapport à l'algorithme classique, seule change l'évaluation de la priorité d'insertion des points dans la file d'attente hiérarchique. Les modifications par rapport à l'algorithme de *lpe* sont présentées pour l'initialisation 5.5 et la propagation 5.6. « imContrainte » est une liste d'image, dont le nombre est le nombre de labels présents dans l'image marqueur. Cette liste est donc indicée par le label. Lorsqu'un point est inséré dans la *fah*, la mesure μ est d'abord récupérée en fonction du label « temporaire » du point, donné par la variable « label −courant ». Les images de contrainte, donnant la mesure μ, n'étant dépendantes que du point en question, il est possible de les calculer avant l'appel à l'algorithme. En fonction du résultat de la propagation, c'est à dire lorsque la classe du pixel définie, la mesure adéquate est récupérée parmi ces images, et ajoutée à la priorité d'insertion.

Algorithme 5.5 : *lpe* épaisse isotrope à contrainte sur les points - Initialisation

Data : imRelief, imLabels, imContrainte, voisinage $\mathcal{N}$
Result : Lpe avec contrainte sur les points

1 ▶ *Modification de la ligne 27 de l'algorithme 5.2*
2 fah-globale ← ajouter v à la priorité imRelief$(v) + \alpha \cdot$ imContrainte [label-courant](v)

Algorithme 5.6 : *lpe* épaisse isotrope à contrainte sur les points - Propagation

1 ▶ *Modification de la ligne 11 de l'algorithme 5.4*
2 fah-globale ← ajouter v à la priorité $(pr \vee$ imRelief $(v)) + \alpha \cdot$ imContrainte [label-courant](v)

Comparaison par rapport à une modification de relief Étant donné la relative simplicité du problème, il est naturel de se demander si ce genre de méthode ne s'apparente pas à une modification du relief inondé antérieurement à une inondation classique. Nous avons utilisé cette méthode dans [DED04] : la modification du relief était faite selon une métrique couleur. Une image de distance $\mathcal{I}_d$ était générée de la manière suivante:

$$\forall p \in E, \mathcal{I}_d(p) = d(\mathcal{I}(p), \Omega_{skin})$$

où $\mathcal{I}$ est l'image couleur originale, d est une métrique couleur, et Ω_{skin} l'ensemble des valeurs apprises pour la classe des points *peau* d'intérêt pour ce problème. Cette image de distance est ensuite utilisée pour supprimer dans le gradient initial les pixels présentant une forte distance à la classe de peau. L'idée sous-jacente est de faciliter la propagation du front d'onde sur les régions qui ne sont pas d'intérêt.

Cette méthode est attractive par sa simplicité, mais elle conduit rapidement à des problèmes sur le gradient: ce dernier est détruit en certains points, et l'inondation présente des fuites sur des régions indésirables.

5.3.5. Information de type « région »

Le formalisme s'apparente fortement au cas précédent, la différence porte sur la fonction d'évaluation. Si la connaissance *a priori* sur les points permet de considérer chacun d'eux de manière indépendante dans le processus d'inondation, ce n'est plus le cas lorsque la connaissance concerne la région. Soit donc une région de points $\mathcal{Z} = \{z_i\}_{i \in J}$, avec J dénombrable, c un nouveau point. L'évaluation devient:

$$P(c \in \mathcal{C} \mid \mathcal{Z})$$

Nous supposons que l'ordre des points de $\mathcal{Z}$ ne compte pas. Bien évidement, la difficulté est qu'il faut mettre $\mathcal{Z}$ à jour lors de l'ajout d'un nouveau point à $\mathcal{Z}$. Dans la mesure où cette évaluation influe sur la priorité à laquelle est inséré le point dans la *fah*, il ne fait pas encore partie de $\mathcal{Z}$. La mise à jour de $\mathcal{Z}$ concerne uniquement les points fermement labélisés. Cela entraîne plusieurs stratégies pour les points du front de propagation:

1. la mise à jour de $\mathcal{Z}$ entraîne la réévaluation de tous les points $\Gamma^{\mathcal{Z}}$ du front de $\mathcal{Z}$.
2. la mise à jour de $\mathcal{Z}$ n'a d'incidence que sur les nouveaux points insérés dans son front de propagation $\Gamma^{\mathcal{Z}}$.

Le deuxième cas n'est en fait qu'un cas particulier du premier, pour lequel la réévaluation des priorités sur les points du contour n'est pas effectuée.

5.3.5.1. Détails de l'algorithme

Il n'y a pas de grande difficulté dans l'élaboration de l'algorithme: par rapport à l'algorithme classique de *lpe*, il faut simplement :

1. associer à chaque région (label) les points labélisés.
2. associer à chaque région les points du front d'inondation.
3. effectuer et mettre à jour une fonction de mesure sur chaque région.
4. utiliser cette fonction pour modifier la priorité lors de l'ajout de ces nouveaux points.

Il faut principalement veiller à ne pas créer de doublon dans les listes de points, ce qui fausserait nécessairement la mesure associée à la région. Enfin, les points du front devenant points de la *lpe* doivent également être gérés correctement, de manière prioritaire.

L'algorithme se décompose en trois sous-algorithmes, décrits ci-après:

1. l'algorithme 5.7 présente l'initialisation du fonctionnement. À l'instar de l'algorithme classique, il prépare la file d'attente hiérarchique globale en vue de la propagation. Il initialise également les objets permettant de calculer la modification de priorité. Il existe un tel objet pour chaque étiquette/label présente dans l'image des marqueurs « imLabel ». Sont également concernés les listes de points du front de chaque label.
2. la partie algorithmique liée à la propagation des points (algorithme 5.8), comme son homologue classique, dépile les points de la file d'attente hiérarchique pour effectuer la propagation. Il est très similaire, pour ne pas dire presque totalement conforme, à la partie correspondante de l'algorithme classique. La différence principale est le traitement des points sur les zones d'épaisseur de la *lpe*: dans le cas présent, puisque la mise à jour des points *lpe* doit être faite précisément, nous avons jugé plus simple la détection des points de la *lpe* sans changer l'état des voisins. Lorsqu'un point est *lpe* dans une zone épaisse, ses voisins, participant à l'épaisseur de la zone, ne sont pas modifiés (contrairement à l'algorithme 5.3). C'est pourquoi la description de cet algorithme sera très succincte.

3. l'algorithme 5.9 met à jour les différentes modifications de priorité. Lorsque la totalité des points à une priorité et une étapes données est traitée, chacun de ces points devient soit fermement labélisé, soit *lpe*. La propagation effectue ce traitement, et place les points traités dans des files distinctes. Ces files sont ensuite utilisées dans cet algorithme, et permettent d'effectuer les modifications des mesures sur chaque région selon ces nouvelles informations.

Description de l'algorithme d'initialisation 5.7 : L'initialisation concerne ici les points de l'image de labels « imLabels », mais également les différentes listes et mesures mentionnées dans l'introduction de cette même section. Nous ajoutons à l'algorithme classique deux parties: la première initialise les objets représentant les mesures sur les régions, ainsi que les listes de points associées à la fois aux fronts et au contenu des régions (nous avons besoin du contenu pour initialiser les mesures sur les régions).

Nous supposons que ces objets sont munis des méthodes suivantes:

créer crée un objet de mesure contenant des valeurs par défaut.

initialiser initialise l'objet avec les points de la région; ces points sont donnés sous forme de liste

priorité produit une valuation de la distance entre le point donné en argument et la mesure interne

ajouter ajoute la liste de points en argument, et modifie la mesure en conséquence

« listes −fronts » est une liste de listes. Le premier index (écrit sous forme « listes −fronts[l] », où « l » est le label) identifie l'étiquette de la région concernée, le deuxième est simplement une liste de points. « listes −fronts » contient en fait les points des fronts de propagation de chaque région. Lors de l'initialisation et de la propagation, nous avons accès à toutes ces informations: en effet, les points du front sont les points à l'état empilé, dans la file d'attente. Par ailleurs, ces points possèdent un label (image de sortie « imSortie ») temporaire jusqu'à labellisation définitive. Il faut éviter l'insertion de doublons dans le front, ce qui est fait par un mécanisme de blocage (ligne 19).

« listes −regions » est très similaire à la liste précédente [(xvi)]. Il est également possible de connaître la totalité des points de chaque étiquette à la sortie de l'initialisation: ce sont les points fermement labélisé. Cette liste est remplie à la ligne 17 de l'algorithme.

Il n'est pas possible à ce stade, c'est à dire pendant la découverte des points des régions, d'insérer les points du front directement dans la *fah* globale. Il faut d'abord calculer les contraintes initiales sur chaque région (ligne 26). Ce n'est qu'ensuite qu'il est possible d'insérer les points du front dans la *fah*. Pour chaque label, nous avons une liste de points du front. Nous pouvons utiliser l'objet de mesure de ce label. Avant insertion dans la *fah*, nous calculons la distance entre le point du front et la mesure effective sur la région, ce qui modifie la priorité d'insertion du point: un point dont la valeur est « proche » de la mesure aura une priorité proche de celle donnée par le relief. Un point « loin » de cette mesure sera considéré comme étant à une altitude plus élevée, et son traitement sera artificiellement retardé (ligne 32).

Description de l'algorithme 5.8 : Comme son homologue classique, la propagation traite chaque priorité de manière *atomique*. Les points de priorité maximale de la file globale sont extraits dans l'ensemble « PL ». Chaque point de « PL » est ensuite soit fermement labélisé, soit *lpe*. Les points *lpe* et labélisés nourrissent respectivement les files « f−watershed » et « f−inonder ». Enfin, les points candidats à la propagation sont placés dans « f−inonder ». Cette dernière file contient des paires de valeurs, l'un des champs étant la coordonnée du point, l'autre le label temporaire du point (déduit du label inondant ce point).

Description de l'algorithme 5.9 : L'algorithme utilise le contenu des files d'attente de l'algorithme 5.8 précédent. Les points déterminés *lpe* sont dans un premier temps marqués, et supprimés de la liste

[(xvi)] seules les structures de données sont différentes, ce que nous préciserons en remarque par la suite

Algorithme 5.7 : *lpe* épaisse isotrope à contrainte sur les régions - Initialisation

Data : imRelief, imLabels, voisinage $\mathcal{N}$
Result : Lpe avec contrainte globale sur les régions

```
1  Initialisation des objets pour les calculs des priorité
2  forall label l de imLabels do
3      objet-mesure[l] ← créer
4      listes-regions[l] ← initialiser avec une liste vide
5      listes-fronts[l] ← initialiser avec une liste vide
6  forall p ∈ imLabels do
7      label-courant = imLabels(p)
8      ...
9      ▶ Détermination de l'état de p selon l'algorithme 5.2
10     ...
11     (reprise à partir de la ligne 18 de l'algorithme classique 5.2)
12     if est-lpe then
13         imSortie(p) = lpe
14         imTravail(p) = lpe
15     else
16         imTravail(p) ← traité
17         listes-regions[label-courant] ← ajouter p
18         forall v ∈ f-file do
19             if imTravail(v) ≠ dans-la-file then
20                 imTravail(v) = dans-la-file
21                 imSortie(v) = label-courant
22                 listes-fronts[label-courant] ← ajouter v
23
24 Initialisation des objets de mesure
25 forall label l de imLabels do
26     objet-mesure[l] ← initialiser avec listes-regions[l]
27 Ajout des points du front à des priorités modifiées
28 forall label l de imLabels do
29     liste-courante = objet-fronts[l]
30     forall p ∈ liste-courante do
31         m = objet-mesure[l].priorite(p)
32         fah-globale ← ajouter p à la priorité imRelief(p) + α · m
```

Algorithme 5.8 : *lpe* épaisse isotrope à contrainte sur les régions - Propagation

```
1  while fah-globale ≠ ∅ do
2      PL ← plateau de priorité maximale de fah-globale
3      pr ← priorité maximale de fah-globale
4      forall p ∈ PL do
5          ▶ Détection des points similaire à l'algorithme classique 5.3
6          f-watershed ← points lpe de PL
7          f-inonder ← points fermement labélisés de PL
8          f-file ← points candidats à la propagation et voisins des points de PL
9      Suite dans l'algorithme 5.9
```

des points du front. Pour cela, nous utilisons le label temporaire à partir de l'image de sortie (ligne 2), ce qui nous permet de déterminer la liste dans laquelle supprimer le dit point (ligne 5).

Ensuite, les points fermement labélisés sont traités (contenus dans la file « f−inonder »). Il n'est pas besoin de mettre à jour l'image de sortie puisque ce dernier contient le label temporaire, qui est égal maintenant au label ferme. Ce label est récupéré et placé dans « l ». Puisque le point fermement labélisé ne fait plus partie du front de la région concerné, il est supprimé de la liste des points du front (ligne 11). Enfin, ce point est ajouté à la liste des points de la région concernée (« listes −regions[l] » ligne 12), permettant ensuite la mise à jour de la mesure sur cette région.

Ceci suppose que cette mise à jour peut être effectuée à partir de la mesure précédente, et de la donnée des nouveaux points.

L'étape suivante consiste à mettre à jour les images par rapport aux candidats à la propagation (dans la file « f−file »). Cette étape supprime également les doublons de cette file, par l'utilisation d'une file temporaire « f−file−temporaire ».

Ensuite, les objets représentant les mesures sont mis à jour avec la donnée des points fermement labélisés. Rappelons que ces nouveaux points se trouvent dans « listes −regions ». Seules les labels ayant reçu une modification sont affectés par cette mise à jour, cette file ayant été préalablement initialisée (à la ligne 7), puis renseignée uniquement avec les modifications issues de « PL ». Le parcours du contenu de « listes −regions » (ligne 23) est effectué uniquement sur les labels existants dans « listes −regions ».

L'étape suivante est optionnelle, et distingue l'algorithme avec et sans mise à jour des points du front. Pour l'algorithme sans mise à jour, ce bloc n'existe pas. Cette étape parcourt l'ensemble des régions ayant reçu une modification de mesure, par le biais de l'ajout de points. Pour ces régions, la priorité de l'ensemble des points est réévaluée, avec la nouvelle mesure. À noter que cette étape est susceptible de créer un nombre important de doublons dans la file d'attente globale, puisque chaque point du front peut être inséré de nouveau après cette réévaluation.

Enfin, les points candidats à la propagation sont insérés dans la file d'attente hiérarchique globale, de la même manière que pour l'étape d'initialisation.

5.3.5.2. Exemple de fonction de contrainte

Les modifications de priorité appliquées à chaque région peuvent être de nature différente, c'est à dire qu'il est possible d'appliquer une force différente à chaque étiquette/classe. Dans la suite, et pour des raisons de simplicité, nous avons appliqué la même force pour tous les marqueurs.

Nous nous sommes servis des développements effectués dans le chapitre 4, concernant les mesures sur les espaces circulaires (§4.2.2 page 119). Nous avons utilisé les deux mesures suivantes :

$$\begin{cases} \eta & = & \bar{\mu_0}\left(\mathcal{Z}\right) \\ m & = & \exp\left(\frac{|x\angle\eta|}{\pi}\right) - 1 \end{cases} \tag{5.8}$$

$$\begin{cases} \eta & = & \bar{\mu_0}^w\left(\mathcal{Z}\right) \\ m & = & \exp\left(\frac{|x\angle\eta|}{\pi}\right) - 1 \end{cases} \tag{5.9}$$

Rappelons que $\bar{\mu_0}$ et $\bar{\mu_0}^w$ sont les moyennes en teinte, respectivement sans et avec pondération par la saturation. L'application $(a, b) \mapsto |a\angle b|$ est bien une fonction de distance mesurant l'angle aigu entre deux éléments. Ces mesures sont significatives de l'homogénéité en teinte de chaque région inondée: plus la région est homogène en teinte, et moins l'écart de la propagation par rapport au relief sera important. L'idée sous-jacente est bien sûr d'obtenir des régions de teinte uniforme.

Enfin, nous soulignons encore le fait que ces deux mesures sont effectuées sur une image autre que celle du relief topographique, mais dont le support $\mathbf{E}$ est le même. Selon le paradigme d'encapsulation objet, la gestion de cette image est totalement interne à « *l'objet informatique* » de mesure. L'algorithme de propagation ne voit que l'objet de mesure, dont les accès sont standards - les méthodes *créer*,

Algorithme 5.9 : *lpe* épaisse isotrope à contrainte sur les régions - Mise à jour

forall $p \in$ *f-watershed* **do**
  if *imTravail*$(p) =$ *lpe* **then** continuer $l =$ imSortie(p)
  imTravail$(p) =$ lpe
  imSortie$(p) =$ lpe
  listes-fronts$[l] \leftarrow$ suppression de p

▶ *mise à jour des points des régions*
forall $l \in$ *listes-regions* **do** listes-regions$[l] \leftarrow \varnothing$
forall $p \in$ *f-inonder* **do**
  $l =$ imSortie(p)
  imTravail$(p) =$ traité
  listes-fronts$[l] \leftarrow$ suppression de p
  listes-regions$[l] \leftarrow$ ajouter p

▶ *mise à jour des points inondés et suppression des doublons*
f-file-temporaire $\leftarrow \varnothing$
forall $(p, l) \in$ *f-file* **do**
  if *imTravail*$(p) \neq$ *dans-la-file* **then**
    imTravail$(p) =$ dans-la-file
    imSortie$(p) = l$
    f-file-temporaire $\leftarrow$ ajouter la paire (p, l)

f-file $\leftarrow$ f-file-temporaire

▶ *Calcul des nouvelles mesures*
forall $l \in$ *listes-regions* **do**
  L'ajout des nouveaux points met à jour la contrainte en conséquence
  objet-mesure$[l] \leftarrow$ ajouter les points de listes-regions$[l]$

▶ *Étape optionnelle de mise à jour des points du front*
forall $l \in$ *listes-regions* **do**
  liste-courante $\leftarrow$ listes-fronts$[l]$
  forall $p \in$ *liste-courante* **do**
    $m =$ objet-mesure$[l]$.priorite(p)
    fah-globale $\leftarrow$ ajouter p à la priorité imRelief$(p) + \alpha \cdot m$

▶ *Ajout des points nouvellement inondés*
forall $(p, l) \in$ *f-file* **do**
  $m =$ objet-mesure$[l]$.priorite(p)
  fah-globale $\leftarrow$ ajouter p à la priorité imRelief$(p) + \alpha \cdot m$
  listes-fronts$[l] \leftarrow$ ajout de p

initialiser, *priorité*, *ajouter* décrites précédemment. L'image sur laquelle est effectivement calculée la mesure est interne à l'objet de mesure. Cet objet « cache » l'image à l'algorithme. Cette *encapsulation* évite une dépendance éventuelle de l'algorithme avec un type de mesure particulier: il est alors possible de changer la mesure de contrainte, sans toucher à l'algorithme. Cette méthode de programmation est totalement **générique**, et permet ainsi l'implémentation d'un grand nombre de mesure sans modification de l'algorithme de propagation.

5.3.5.3. Résultats

Nous allons présenter les résultats de l'inondation avec information *a priori* sur les régions, selon les deux mesures précédemment citées. Par la nature même de ces mesures, l'impact sera significatif sur les images assez colorées, c'est à dire assez riche en saturation [(xvii)].

La présentation sera la suivante: sur des exemples choisis, nous allons dans un premier temps évaluer l'influence du terme de mesure, en faisant varier le paramètre α (cf. équation 5.7). Nous comparerons également les deux mesures. Nous effectuerons ensuite la comparaison entre les algorithmes avec et sans mise à jour des points du front. Le relief utilisé pour l'inondation sera le gradient du canal de luminance.

Influence de α : La figure 5.14 présente le résultat de la segmentation (sans mise à jour) sur l'image « Parrots », à partir des marqueurs de l'image 5.14(b). La deuxième ligne présente les résultats de la mesure sans pondération, la troisième ligne celle avec pondération. Nous remarquons que pour chaque valeur de α, les partitions des deux mesures sont pratiquement égales.

Pour α faible (image 5.14(c) et 5.14(g)), la partition devrait ressembler à celle fournie par la *lpe* classique (en vert sur les figures). Or ce n'est pas le cas. Nous attribuons ceci aux deux phénomènes suivants:

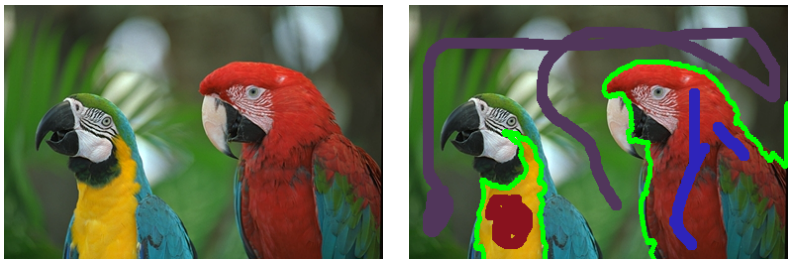
1. la tête du perroquet rouge est grignotée par le marqueur du fond. Curieusement, dans ce cas précis une petite brèche inonde la partie supérieure de la tête du perroquet, alors que la distribution couleur (dans les tons verts) ne correspond pas. Inversement, le fond ne se propage pas suffisamment vite à l'intérieur du perroquet de gauche, ce qui conduit accidentellement à une bonne segmentation de l'animal (au sens visuel).
2. l'algorithme proposé n'effectue pas de swamping du relief, contrairement à la *lpe* classique. La raison de ceci est que le swamping devrait se faire sur des données indépendantes du temps uniquement (comme le relief), alors que la mesure de ressemblance utilisée (et indirectement la priorité à partir d'une ligne de crête) n'est précisément plus indépendante du temps.

Pour des valeurs de α plus élevées, le comportement est conforme à ceux attendus. Le perroquet rouge est segmenté dans sa totalité (homogénéité en rouge), et la partie jaune du perroquet de gauche est également segmentée. Cependant, ces résultats ne diffèrent pas de la *lpe* classique.

Sur la figure 5.15, les résultats sont plus intéressants. Encore une fois, les résultats sont très similaires pour les versions avec et sans pondération par la saturation. Pour une valeur faible de α, la partie basse droite de l'image (images 5.15(c) et 5.15(f)) présente une partie mieux découpée que celle avec le gradient. Le contour se place en effet sur la frontière entre la partie bleue et marron. Le reste de l'image est conforme à la *lpe*.

Pour des valeurs de α plus élevées, le terme d'attache à l'homogénéité de zone possède un poids plus important. Ceci permet à la propagation de dépasser la ligne de gradient dans la zone rouge intermédiaire (au dessus du personnage). Le chien au dessus de la personne, dont la teinte est intermédiaire entre celle du personnage et celle de la zone rouge, passe alors dans la région associée au personnage. Par contre, la zone à gauche du personnage ne possède plus de frontière sur la partie rouge. En fait, le nouveau contour est déplacé sur une frontière bleu-vert présente dans cette partie.

[(xvii)] Rappelons que plus la saturation est élevée, plus le champ de teinte est pertinent.



(a) Image originale (b) Marqueurs et *lpe* classique

(c) Contrainte de teinte - $\alpha = 10^{-4}$ (d) Contrainte de teinte - $\alpha = 0.2$ (e) Contrainte de teinte - $\alpha = 0.6$ (f) Contrainte de teinte - $\alpha = 2$

(g) Contrainte de teinte pondérée - $\alpha = 10^{-4}$ (h) Contrainte de teinte pondérée - $\alpha = 0.2$ (i) Contrainte de teinte pondérée - $\alpha = 0.6$ (j) Contrainte de teinte pondérée - $\alpha = 2$

Fig. 5.14.: Illustration de la contrainte globale sur l'image « Parrots »- $\alpha \in \{10^{-4}, 0.2, 0.6, 2\}$. (b) : marqueurs originaux et *lpe* classique (en vert). (c)–(f) : *lpe* à contrainte globale (en rouge) selon un angle moyen sans pondération par la saturation. (g)–(j) : *lpe* à contrainte globale (en rouge) selon un angle moyen prenant compte de la pondération par la saturation.

Comparaison avec et sans mise à jour du front : Les deux algorithmes produisent des résultats si similaires qu'il est assez difficile de distinguer les partitions obtenues. Cependant, sur les partitions présentées en figure 5.16, nous pouvons dire que la version sans mise à jour produit un meilleur résultat, et qu'elle n'apporte aucune amélioration sensible malgré les calculs beaucoup plus longs. Remarquons bien que ces conclusions ne sont valables qu'avec les mesures considérées: les résultats pourraient être sensiblement différents avec d'autre type de force (géométrique par exemple).

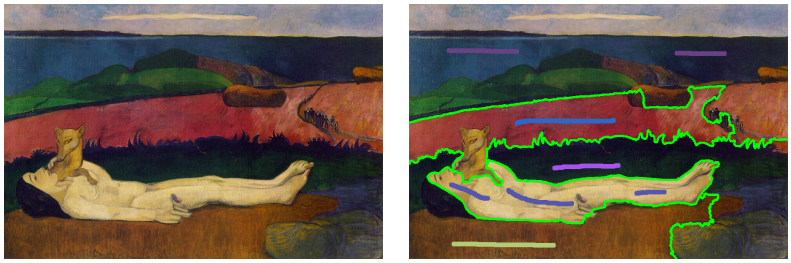
5.3.6. Quelques détails d'implémentation

Dans notre première implémentation de l'algorithme, nous avons codé les fronts des différentes régions à l'aide de listes. La liste est un objet fourni par la *STL* permettant des insertions et suppressions en $O(1)$ [(xviii)]. Les coûts de parcours sont par contre plus élevés que pour un simple vecteur. L'inconvénient majeur du vecteur, pour une telle application, est que si une suppression doit avoir lieu ailleurs qu'aux extrémités du vecteur (premier et dernier élément), une copie totale de celui-ci doit être effectuée.

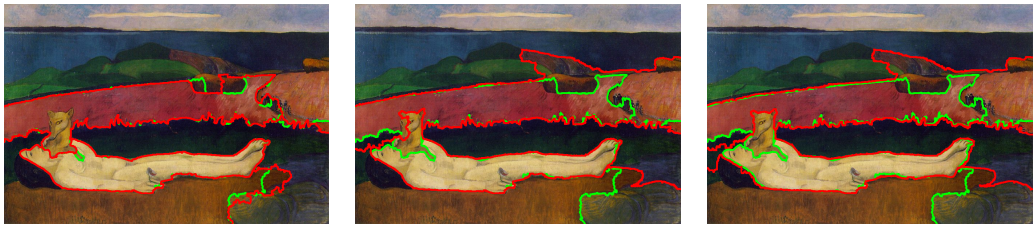
Lors de la détermination de l'état d'un point du front, il faut mettre à jour le front de la région concernée par ce changement. Ceci implique la recherche du point dans le front puis sa suppression. Si le front est codé à l'aide de liste ou de vecteur, il faut parcourir chaque point et tester si effectivement, il s'agit du point concerné par ce changement. Cette démarche ralentit considérablement l'algorithme.

Puisqu'il faut *chercher* les points, nous avons effectué une implémentation des points du front à l'aide de « dictionnaire », ici plus précisément à l'aide de l'objet « set » de la *STL*, dont le comportement est identique. Rappelons que les dictionnaires ordonnent les éléments selon leur type « clef ». La structure

[(xviii)] voir à ce sujet la partie introductive §2.1.2 page 11



(a) Image originale (b) Marqueurs et *lpe* classique



(c) Contrainte de teinte - $\alpha = 10^{-4}$ (d) Contrainte de teinte - $\alpha = 0.2$ (e) Contrainte de teinte - $\alpha = 0.6$

(f) Contrainte de teinte pondérée - $\alpha = 10^{-4}$ (g) Contrainte de teinte pondérée - $\alpha = 0.2$ (h) Contrainte de teinte pondérée - $\alpha = 0.6$

Fig. 5.15.: Illustration de la contrainte globale sur l'image « La Perte de Virginité » (Gauguin). (b) : marqueurs originaux et *lpe* classique (en vert). (c)–(e) : contrainte selon la teinte seule (en rouge). (f)–(h) : contrainte selon la teinte pondérée par la saturation (en rouge).

interne est complexe, et généralement codée sous forme d'arbre binaire de recherche. Les temps de recherche sont garantis en $O(\log(N))$, N étant ici le nombre de clefs.

Pour coder le front de chaque région, nous avons utilisé l'offset des points comme type clef de l'objet set [(xix)]. Nous avons alors des coûts de recherche croissants selon le logarithme de la taille du front, au lieu d'une recherche de coût *linéaire*, comme c'est le cas pour les listes et vecteurs.

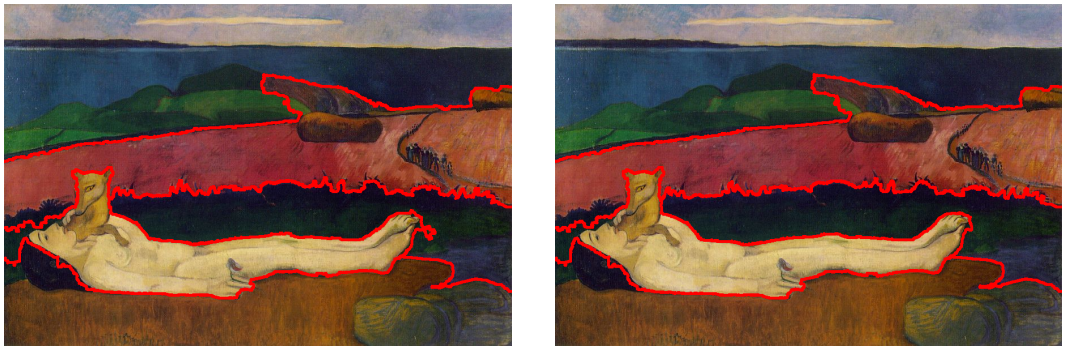
L'accélération des traitements qui en résulte est très importante, et rapproche les temps de calcul de l'algorithme *sans mise à jour* des temps de calcul de la *lpe* classique. Nous n'avons pas chiffré précisément cette accélération: pour une image telle que « La perte de la virginité » (figure 5.15), le temps de calcul tombe d'environ 25 minutes à environ 1 minute.

5.3.7. Conclusion et perspectives

Résumé

Nous avons étudié dans cette partie l'ajout d'informations *a priori* dans le processus d'inondation. Cette information modifie la manière dont les points sont ajoutés à la file d'attente globale du processus d'inondation; elle s'exprime simplement comme distance sur le point en cours d'ajout à une région, et une mesure globale sur la région. Nous avons proposé deux algorithmes: le premier considère à chaque

[(xix)] contrairement à l'objet dictionnaire « map », le champ de donnée n'est pas important dans l'objet « set », seule la présence ou non d'un élément compte.



(a) Avec mise à jour du front (b) Sans mise à jour du front

Fig. 5.16.: Comparaison entre les algorithmes avec et sans mise à jour des points du front de propagation ($\alpha = 0.2$ et teinte pondérée). Pour les deux images, la *lpe* classique est représentée en vert, alors que l'inondation à contrainte externe est représentée en rouge. Les deux partitions sont très similaires, seule une petite boucle apparaît au pied du personnage dans l'algorithme avec mise à jour.

ajout la modification de la mesure globale sur la région, et modifie en conséquence la priorité des points du front, c'est à dire ceux n'appartenant pas encore à la région mais directement connexes. Le deuxième algorithme ne considère la modification de la mesure que pour les points futurs du front. Il s'en suit une accélération considérable des calculs.

Perspectives

Les premiers travaux venant à l'esprit sont dans la lignée de ce qui a été présenté, en utilisant des mesures plus fines et précises en fonction des besoins et des applications. Nous avons utilisé des marqueurs assez grossiers, conduisant à une partition de l'image en régions selon une mesure unique. L'algorithme que nous avons mis en œuvre n'est pas limité par le nombre de mesures possibles, indépendamment sur chaque région. Dans une perspective de segmentation non-supervisée, il est envisageable d'utiliser plusieurs types de mesures : par exemple une région pour le fond posséderait une mesure ayant un impact faible sur sa propagation (fond très général), alors que la mesure sur la forme recherchée se servirait des informations extraites précédemment. Les mesures peuvent également être améliorées, en ajoutant un facteur multi-modal aux distributions de couleurs par exemple. Enfin il serait également intéressant d'utiliser des mesures de nature mixte, associant contenu (couleur) et géométrie des formes en cour d'inondation.

Sur un plan purement informatique, des nombreuses améliorations sont également possibles, améliorations principalement en termes de temps de calcul. Le premier exemple de ce genre d'amélioration a été donné dans cette partie même par le deuxième algorithme. Bien que produisant, en théorie [(xx)] des régions sur lesquelles les mesures sont moins précisément appliquées, les résultats des deux approches, nous ont semblé comparables. Une série de tests sur de petites régions serait un complément appréciable à cette observation. Les coûts de gestion des points de chacun des fronts ne sont pas exactement négligeables. En effet, lors de la labellisation d'un point, il faut trouver la région à laquelle appartient ce point, il faut ensuite trouver le point en question parmi tous les points du front, etc.. Les temps de recherche ne sont pas faibles. Il serait possible de dégrader d'avantage l'algorithme de manière à effectuer cette recherche uniquement toutes les N passes, avec $N > 1$ [(xxi)]. Ceci capitaliserait les coûts de recherche.

Enfin, nous avons proposé une analogie de l'approche proposée avec la formulation par EDP d'évolution de contours. Cette analogie permet de mieux comprendre la différence essentielle de ce nouvel algorithme avec la *lpe*, et sous une forme très compacte.

[(xx)] nous n'avons essayé que très peu en pratique, précisément à cause des temps de calcul titanesques.
[(xxi)] nous nommons par « passe » ce que nous avons précédemment nommé « plateau de priorité ».

5.4 Inondations localement contraintes

La méthode proposée dans la section §5.3 précédente consiste à modifier le comportement de l'inondation par l'intervention d'une information sur les points inondés. L'objectif initial de cette méthode est de limiter les défauts du relief inondé lors de l'inondation en un nombre faible de régions, en modulant la vitesse de propagation selon une mesure de similitude par rapport aux informations *a priori*.

La méthode exposée dans cette section propose une nouvelle approche consistant à contraindre l'inondation du fluide selon la configuration locale du front de propagation. Certaines approches existantes proposent le nom d'*inondation à l'aide de fluide visqueux*, par analogie avec les forces internes d'un tel fluide.

Cette section s'articule de la manière suivante: nous introduirons la problématique générale de l'inondation localement contrainte. Nous présenterons succinctement les approches existantes, et décrirons ensuite notre approche.

Dans un deuxième temps, nous décrirons en détail l'algorithme implémentant notre approche. Nous en soulignerons certaines propriétés et limitations. Nous présenterons ensuite les résultats sur des images synthétiques et réelles.

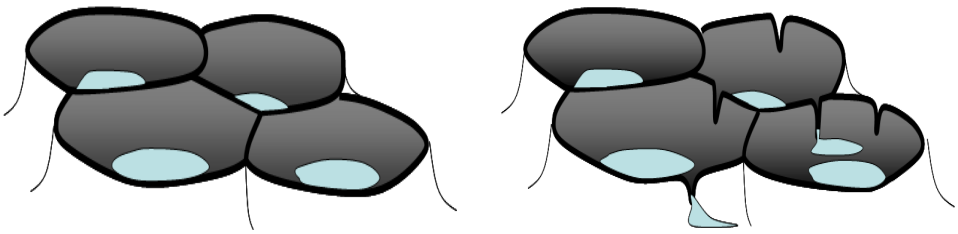
Nous effectuerons ensuite une digression autour de la contrainte locale. Nous rapprocherons notre modèle de celui, plus classique, basé sur la courbure locale du front de propagation. Cette expression nous permettra de compléter l'analogie par EDP de l'algorithme que nous proposons. Nous conclurons cette section en insistant sur les perspectives envisageables à l'algorithme.

5.4.1. Contrainte locale et viscosité

5.4.1.1. Origine du problème

La motivation pour l'utilisation d'un fluide visqueux est double:

– Comme nous l'avons largement constaté sur nos séquences, lors de l'inondation d'un relief avec un faible nombre de classes inondantes, la variabilité sur les frontières des partitions est très importante. Ce point est illustré sur la figure 5.17. La qualité du relief inondé étant par ailleurs difficilement contrôlable, une méthode d'inondation moins sensible à ce genre de défaut serait d'un grand secours.



(a) Cas idéal (b) Cas réel

Fig. 5.17.: Défaut de gradient entraînant une instabilité lors du partitionnement en un faible nombre de régions. (a) : les contours sont correctement marqués entre tous les bassins versant. Lors du processus d'inondation, les fluides évoluent progressivement à l'intérieur des bassins versants. (b) : le gradient présente certains défauts. Les fluides s'échappent entre les brèches et évoluent rapidement dans des directions non désirées.

– la *lpe* ne permet pas d'influencer la forme des contours des régions obtenues. Or certains objets d'intérêt présentent souvent des contours réguliers (faible courbure). De manière intuitive, les fluides visqueux présentent un écoulement tel que le front, c'est à dire l'interface du fluide dans le sens de l'écoulement, est également régulier.

L'inondation à l'aide d'un fluide visqueux permettrait en théorie d'ajouter une contrainte de régularisation à la ligne de partage des eaux: la « viscosité » se manifesterait par les forces sur l'interface du

fluide, et qui modifierait localement son écoulement. Par analogie avec nos algorithmes d'inondation, l'interface du fluide est le *front de propagation*, et l'écoulement la *vitesse* à laquelle avance le front. Si le front de propagation est une notion bien acquise, ce n'est pas le cas en ce qui concerne la *vitesse*. Nous pourrions faire l'hypothèse que la vitesse de propagation est directement liée à la priorité des points: en effet, lorsqu'un point est traité, ses voisins sont mis en file d'attente ce qui simule la propagation. Si ensuite les points voisins sont traités avant tous les autres, localement le front « avance », c'est à dire se propage, plus vite dans le voisinage de ce point que partout ailleurs. Cette hypothèse semble assez cohérente avec ce que nous souhaitons obtenir, nous allons la mettre en pratique dans la suite.

Voyons d'abord les approches existantes.

5.4.1.2. Les approches existantes

Un certain nombre d'approches existent déjà dans la littérature. Les deux principales approches à notre connaissance, sont celles de Vachier & Meyer [MV02] et N'Guyen & al. [NWvdB03].Nous allons présenter et critiquer ces approches.

Vachier & Meyer [MV02] Pour la simulation de la viscosité, les auteurs modifient la fonction de relief par un ensemble de fermetures de manière à fermer ou « lisser » les brèches. Les fermetures sont réalisées de manière à préserver les contours forts, et à fermer les zones de gradient faible. Les fermetures sont donc de tailles décroissantes avec l'altitude. Ceci s'explique par le fait qu'un contour fort est très représentatif d'une véritable transition, alors qu'un contour faible peut être assimilé à du bruit. Puisqu'elle agit sur le relief de départ et non sur le processus d'inondation, cette approche a l'inconvénient de connecter des régions qui *a priori* ne doivent pas l'être.

N'Guyen, Worring & Boomgaard [NWvdB03] Les auteurs établissent une analogie entre la *lpe* et un problème de minimisation de fonctionnelle. Rappelons que nous avons discuté de ce rapprochement dans la partie §5.1.2.3 (page 181). Cette formulation sous forme d'optimisation permet aux auteurs d'ajouter des termes de régularisation par rapport aux contours.

Si l'approche est élégante, la mise en pratique d'une telle méthode est en réalité assez délicate. La fonctionnelle d'énergie doit d'abord être discrétisée, ce qui peut poser des problèmes. Ensuite, la minimisation d'une telle fonctionnelle est un problème difficile. Pour contourner la difficulté, les auteurs partent d'une *lpe*, et réassignent ensuite les points sur des contours entre les bassins versants. Ce réassignement est effectué de manière à réduire la fonction d'énergie, qui par ailleurs doit être calculée à chaque modification. Finalement, cette méthode ne garantit en rien la minimisation globale de l'énergie, et leur méthode semble capturer la partition finale dans un minimum local de l'énergie.

5.4.1.3. Approche proposée

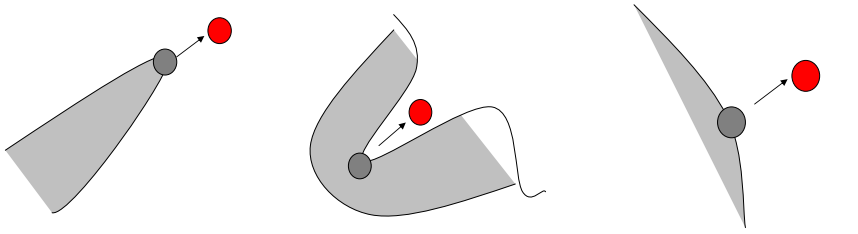
Notre approche n'utilise aucune des deux méthodes proposées précédemment et se veut plus simple et tout aussi efficace.

Selon les remarques émises pour la méthode de Vachier & Meyer [MV02], nous souhaitons appliquer une contrainte locale simulant la viscosité du fluide, pendant le processus d'inondation. Nous proposons l'approche suivante pour effectuer cette simulation.

Contrainte à partir du rapport de surface Dans notre schéma, la viscosité se manifeste par une contrainte locale du front, de manière à ne plus considérer les points du front comme isolés, mais comme des points soumis à l'action de forces physiques fonction du voisinage immédiat. Le comportement désiré d'un point du fluide est le suivant:

- **ralenti** lorsqu'il se trouve dans une zone convexe du front (« pic »).
- **accéléré** lorsqu'il se trouve dans une concavité du front (« creux »).
- **classique** au sens de la *lpe* lorsqu'il se trouve sur un front « plat » ne présentant pas de courbure particulière

Ces contraintes sont résumées dans la figure 5.18.



(a) Ralentissement (b) Accélération (c) Comportement classique

Fig. 5.18.: Comportements d'un fluide visqueux.

Contrairement à l'inondation classique, on se « retient » d'inonder une zone à partir d'une brèche étroite du gradient et de manière « précipitée ». Les forces agissant à l'interface du front déplaceront naturellement la ligne de partage des eaux, c'est la raison pour laquelle nous ne parlerons pas de *ligne de partage des eaux* - définie de manière précise - mais simplement d'*inondation contrainte*.

Soit Γ le contour du bassin versant. Il est possible de déterminer la configuration locale, en un point $\mathbf{x}$ de Γ, de manière simple. Soit $\mathcal{B}$ une boule de rayon $r_{\mathcal{B}}$ et $\mathcal{B}_{\mathbf{x}}$ cette même boule centrée en $\mathbf{x}$. Par la suite, nous nommerons $\mathcal{B}$ la *boule de contrainte*. Soit L l'intérieur de la région délimitée par Γ, c'est à dire la classe à partir de laquelle nous inondons $\mathbf{x}$ (les zones grises sur la figure 5.18). Il est possible de connaître la configuration locale par le rapport suivant:

$$\mathcal{A}_{\Gamma}(\mathbf{x}) = \frac{\mathcal{A}(L \cap \mathcal{B}_{\mathbf{x}}) - \mathcal{A}(L^c \cap \mathcal{B}_{\mathbf{x}})}{\mathcal{A}(\mathcal{B})} \tag{5.10}$$

où L^c est le complémentaire de L, et $\mathcal{A}$ l'application calculant la surface. D'après l'expression de $\mathcal{A}_{\Gamma}$, nous avons effectivement :

- $\mathcal{A}_{\Gamma} < 0$ dans les zones convexes
- $\mathcal{A}_{\Gamma} > 0$ dans les zones concaves
- $\mathcal{A}_{\Gamma} \approx 0$ pour les zones où le front de propagation est « plat »

L'intérêt de cette mesure, que nous nommerons *force de contrainte*, réside principalement dans la facilité de calcul. Par ailleurs, elle est strictement monotone par rapport à $\mathcal{A}(L)$, ce qui nous permettra d'effectuer des mises à jour très simple de la mesure pendant la propagation (changement de la configuration dans le voisinage). Nous verrons cela en détail lors de la description algorithmique.

Simulation de la viscosité par modification de priorité: Pour des raisons similaires à l'algorithme avec information *a priori*, la viscosité du front de propagation sera mise en œuvre par une modification de la priorité. Si l'on revient sur la formulation de la propagation à l'aide d'EDP, nous avons:

$$\frac{\partial \Gamma}{\partial t} = \frac{1}{\|\nabla f\| + \mathcal{A}_{\Gamma}} \tag{5.11}$$

$\mathcal{A}_{\Gamma}$ est dépendant de Γ. Étant donné cette équation, il faudra veiller à mettre correctement à jour $\mathcal{A}_{\Gamma}$ à chaque modification de Γ (c'est à dire à chaque pas de temps) pour effectuer correctement la propagation. Enfin, remarquons que cette mesure aura un impact fort à l'intérieur des plateaux (zones pour lesquelles $\|\nabla f\| \approx 0$). Rappelons que sur les plateaux, une fonction distance implicite impose la forme des fronts de propagation. Cette distance est liée à la définition même de la *lpe*, à partir des SKIZ. Reprenant l'équation 5.11, cette fonction distance et notamment son *isotropie*, dépendra de la fidélité de l'implémentation par rapport à l'équation. Nous discuterons de ce point lors de la description de l'algorithme, en illustrant par quelques exemples de biais.

5.4.2. Algorithme de propagation avec contrainte locale

5.4.2.1. Description

L'algorithme complet d'inondation à contrainte locale est assez long. Nous avons tenu à être assez précis dans cette description, car les pièges conduisant à des biais importants sont nombreux. Ces biais se manifestent notamment sur les zones plates, et ils ne sont ainsi pas facilement décelable sur des images réelles. Nous avons scindé l'algorithme en quatre parties:

1. l'algorithme 5.10 présente l'initialisation. Il s'agit essentiellement de la recherche des sources d'inondation. Contrairement à l'inondation classique, nous ne pouvons pas calculer directement les priorités des points entrants dans la *fah* globale.
2. l'algorithme 5.11 présente ce que nous avons appelé la « *relaxation* » de l'étape d'initialisation. Pendant cette étape, les priorités avant insertion des points dans la *fah* globale sont calculées.
3. l'algorithme 5.12 présente la propagation. La partie concernant la recherche de point est analogue à l'algorithme classique d'inondation et ne sera pas reprise. La différence concerne la mise à jour des points qui ne peut être faite directement.
4. l'algorithme 5.13 concerne la mise à jour des priorités. Les priorités des points du front d'un point nouvellement ajouté doivent être mises à jour, et la priorité d'un nouveau point est calculée de manière analogue à l'étape d'initialisation.

Description de l'algorithme 5.10: L'étape d'initialisation est assez similaire à celle décrite pour la ligne de partage des eaux classiques (algorithme 5.2). Pour chaque point p du marqueur, nous mettons ses voisins *non-marqueurs* dans une file intermédiaire « f−candidat ». Si après parcours de l'ensemble du voisinage immédiat p n'est pas *lpe*, nous insérons les points de « f−candidat » (ie. les voisins) dans une seconde file intermédiaire « f− first −pass ».

La raison de la seconde file est la suivante: si nous marquons les points de la *fah* globale directement, nous risquons de créer des inter-actions entre les points déjà parcourus (labels temporaire) et les points à parcourir (non encore labélisé). La labellisation temporaire est ainsi reportée à la fin de parcours.

Cette seconde file servira à faire par la suite la distinction entre les points entrant dans la file d'attente hiérarchique globale « fah−globale ». La position du point du front et son label temporaire (« imLabels(p) » ligne 35) sont introduit dans « f− first −pass », pour être ensuite utilisé dans l'étape où on met effectivement les labels du front d'onde dans l'image de sortie (ligne 40 de l'algorithme).

Description de l'algorithme 5.11: Cet algorithme présente la partie de « relaxation » de l'initialisation. Après l'étape de découverte des points devant être introduit dans la *fah* globale et avant leur insertion effective dans la *fah* globale, nous devons assigner une priorité à ces points. Cette priorité est fonction de la configuration locale.

L'algorithme 5.11 présente précisément la manière dont sont calculées ces priorités. Pour chacun des points du front d'onde, nous centrons la boule servant au calcul du rapport des surfaces. Nous parcourons ensuite les points intérieurs à cette boule, et comptons les points fermement labélisés et du même label que celui qui a donné le label temporaire du point central. La priorité est modifiée en prenant en compte uniquement les points fermement labélisés. Les points qui ne sont pas fermement labélisés sont ceux du front d'onde, et ils ne comptent pas dans le calcul de la priorité: les considérer conduit rapidement à des effets d'avalanche et d'instabilité numérique.

Dans le cas particulier de l'initialisation, les points du front n'inter-agissent pas entre eux, puisque leur état les exclus du calcul de la priorité. Ce ne sera par contre pas le cas lors de la propagation. Enfin « imPriorité » est une image recevant les mêmes priorités que la *fah* globale dans le cas de l'initialisation. Il servira dans l'étape de propagation lorsque les priorités seront mise à jour, et nous reviendrons sur ce point.

Algorithme 5.10 : Inondation par fluide à contrainte locale - Initialisation

Data : imRelief, imLabels, imAlpha
Result : imSortie : Partition à partir d'une inondation à contrainte locale

```
f-candidat, f-watershed, f-inonder
fah-globale
• Initialisation :
f-first-pass ← ∅
imTravail ← non-traité
imPriorité ← 0
imSortie ← imLabels
Test over the neighborhood :
forall p ∈ imSortie do
    if imSortie(p) = label-fond then continuer boucle
    Pass 1: test for watershed line:
    f-candidat ← ∅
    est-lpe ← faux
    forall v ∈ N_p \ p do
        label-neighbor = imSortie(v)
        if label-neighbor ≠ label-fond then
            if label-neighbor ≠ imSortie(p) then
                est-lpe ← vrai
                break

        else
            point that can be flood
            f-candidat ← ajouter v

    Pass 2: if the point does not belong to the watershed line, we can append the neighbors into the
    queue but we should wait for the first pass to finish completly
    if est-lpe then
        f-watershed ← ajouter p
        imTravail(p) = lpe
    else
        imTravail(p) ← traité
        the discovered neighbors : they are put inside another queue, for later insertion into the HQ
        forall v ∈ f-candidat do
            ensure not already inside the queue
            if imTravail(v) = non-traité then
                f-first-pass ← ajouter paire(v, imLabels(p))
                imTravail(v) = dans-la-file
Pass 3: Temporary label of queued and watershed points
forall v ∈ f-watershed do imSortie(v) = lpe
forall v, l ∈ f-first-pass do imSortie(v) = l
```

Algorithme 5.11 : Inondation par fluide à contrainte locale - Initialisation - Suite

```
1  A_ball ← total non-null area of the ball
2  Pass 4: Computing of the exact priority before insertion into the HQ
3  forall p, l ∈ f-first-pass do
4      A = 0
5      Counting the points of interest
6      forall v ∈ {N_p^B} do
7          if (imTravail(v) = traité) et (imSortie(v) = l) then A = A + 1
8      Modifying the priority
9      p_h = imRelief(p)
10     p_a = imAlpha(p)
11     p_fah = p_h − p_a × (2∗A − A_ball) / A_ball
12     fah-globale ← ajouter p à la priorité p_fah
13     imPriorité(p) ← p_fah
```

Description de l'algorithme 5.12: Il s'agit de l'étape de sortie des points de la *fah* globale. Nous testons d'abord si la priorité du point correspond bien à la dernière priorité connue du point. « imPriorité » contient cette information. Si tel n'est pas le cas, le point n'est pas traité.

L'algorithme doit ensuite déterminer pour chacun de ces points p s'il appartient à la « ligne de partage » des eaux, si on peut encore parler d'une telle ligne. La démarche est très similaire au cas classique, et nous reportons le lecteur à l'algorithme 5.3 pour les détails. La différence est ce qu'il se passe sur les points sortant du front, c'est à dire fermement labélisés, et les nouveaux points découverts.

Dans les deux cas, il est impossible de les ajouter immédiatement à la file, ni à l'image de sortie « imSortie ». Les points fermement labélisés p sont insérés dans une file intermédiaire « f − first − pass ». Cela s'explique selon les mêmes remarques que lors de l'initialisation. Les points nouvellement découverts sont ajoutés à une file intermédiaire « f − first − pass − front ». Ils ne peuvent pas être marqués directement dans « imSortie » car nous avons besoin de « imSortie » intacte, c'est à dire ne contenant que des points fermement labélisés, lors du calcul des priorités. Pour le calcul des contraintes, nous avons également besoin du label temporaire affecté à un point v du nouveau front car nous ne comptons que les points fermement labélisés de même label. C'est pourquoi v et son label temporaire donné par « imSortie(p) » sont insérés dans la file temporaire « f − first − pass − front » (ligne 15 de l'algorithme).

Lorsqu'enfin tous les points de la priorité courante ont été parcourus, il est possible de mettre à jour l'image de sortie (ligne 18).

Description de l'algorithme 5.13 Chaque point ajouté à l'image de sortie, à la fin de l'algorithme précédent, modifie la contrainte locale des points qui n'ont pas été traité (voir figure 5.19). Ces points sont en effet fermement labélisés, et sont donc considérés dans le calcul de la contrainte locale.

Les points fermement labélisés sont toujours dans la file intermédiaire « f − first − pass » (ligne 3). Nous centrons la boule $\mathcal{B}$ en chacun de ces points, et cherchons, parmi tous les voisins, les points du front de même label. Nous avons l'information de label courant l dans la file intermédiaire « f − first − pass ». Les points du front possèdent également un label, temporaire.

Lorsque nous trouvons un de ces points, nous mettons à jour l'image des priorités. La mise à jour est ici très simple. Puisque nous ajoutons un unique point au voisinage, la variation de priorité est donné par $\frac{-2}{\mathcal{A}_{ball}}$ avec le facteur de « imAlpha ». Il faut ensuite remettre ce point dans la *fah* globale, mais nous passons encore une fois par une file intermédiaire « f − file ». En effet, il se peut qu'il y ait plusieurs mise à jour d'un même point à partir de voisins différents (cf. figure 5.19(c)). Nous ajoutons le point v dans « f − file » s'il n'est pas déjà dans cette file (ligne 8). Ce mécanisme réduit le nombre d'insertion dans la *fah* globale, déjà encombrée par la création de doublons.

Enfin, les points nouvellement découverts, c'est à dire ceux constituant le nouveau front de propa-

Algorithme 5.12 : Inondation par fluide à contrainte locale - Propagation

```
1  • Détermination de l'état des points du front:
2  while fah-globale ≠ ∅ do
3      PL ← plateau de priorité maximale de fah-globale
4      pr ← priorité maximale de fah-globale
5      forall p ∈ PL do
6          if imPriorité(p) ≠ pr then continue
7          ...
8          Détermination de l'état de p similaire à l'algorithme 5.3
9          ...
10         if est-lpe then
11             imSortie(p) ← lpe
12             imTravail(p) ← lpe
13         else
14             f-first-pass ← ajouter paire(p, imSortie(p))
15             forall v ∈ f-candidat do f-first-pass-front ← ajouter paire(v, imSortie(p))
16
17     The previous front points get their final label
18     forall v, l ∈ f-first-pass do
19         imTravail(v) ← labélisé
20         imSortie(v) ← l
21     Suite de l'algorithme en 5.13
```

gation, sont dans la file intermédiaire « f−first−pass−front ». Les éléments de cette file contiennent à la fois la position p du point, ainsi que le label temporaire l. L'image de travail et de sortie sont mise à jour, et les éléments nécessaires au calcul de la priorité du point sont réunis. Le calcul est le même que celui présenté dans l'algorithme 5.11.

La mise à jour des priorités après une étape de propagation justifie l'existence de l'image de priorité « imPriorité ». Il est difficile et coûteux de chercher des points dans la *fah*, et encore plus coûteux de modifier la priorité de tels point. Nous choisissons au prix de doublons dans la *fah* cette méthode de mise à jour, qui s'est par ailleurs avérée plus rapide que la manipulation de la *fah*[xxii]. La modification des priorités est prise en compte au début de l'algorithme de propagation 5.12 (ligne 6).

Remarque : Le choix de création de doublons entraîne malheureusement une augmentation très importante du nombre de clefs et de points dans la file. Ceci a pour conséquence un algorithme rapide au début, et qui ralentit au cours de la propagation. Ce phénomène s'accompagne d'une consommation mémoire anormalement élevée. Un moyen de contourner ce problème est de remettre à jour entièrement la *fah*, après un certain nombre d'itérations. Nous nous servons de l'image de travail « imTravail » et de l'image des priorités « imPriorité ». La mise à jour de la *fah* a lieu toutes les n boucles de la boucle de propagation principale. Dans nos implémentations, une valeur de $n = 100$ ou $n = 200$ réduit considérablement le nombre de points et le de clefs présents dans la file. La réduction dépend de la taille de la boule des contraintes; plus la boule est grande, plus le nombre de doublons est élevé, et plus le facteur de réduction est important. Pour un rayon de 40 pixels et une mise à jour toutes les 200 itérations, nous avons observé une réduction du nombre de points de l'ordre d'un facteur de 10 à 100, et d'un facteur de 2 à 10 (parfois même 100 !) pour le nombre de clefs. Il est difficile de quantifier l'avantage réel d'autant plus que sans cette mise à jour, ces doublons s'accumulent. Mais ce fait souligne l'importance de cette mise à jour pour garder des temps de calcul raisonnables, ou pour mieux dire tolérables !

[xxii] La modification - suppression puis insertion - dans la *fah* coûte $2 \times \log(N)$ opérations, avec N le nombre de priorités. L'insertion d'un doublon coûte $\log(N+1)$ puisque dans le pire des cas, nous augmentons le nombre de priorités. L'accès à l'image des priorités est considéré en $O(1)$.

Algorithme 5.13 : Inondation par fluide à contrainte locale - Propagation - Suite

```
1  Update of the priority of the points still in the queue
2  f-file ← ∅
3  forall p, l ∈ f-first-pass do
4      forall v ∈ {N_p^B} do
5          if (imTravail(v) = dans-la-file) et (imSortie(v) = l) then
6              p_a = imAlpha(v)
7              imPriorité(v) = imPriorité(v) − 2∗p_a / A_ball
8              if v ∉ f-file then f-file ← ajouter v
9
10 forall v ∈ f-file do fah-globale ← ajouter v à la priorité imPriorité(v)
11 forall p, l ∈ f-first-pass-front do
12     if imTravail(p) ≠ non-traité then continue
13     imTravail(p) ← dans-la-file
14     imSortie(p) ← l
15     Insertion des nouveaux points dans la fah, selon l'algorithme 5.11
16     ...
```

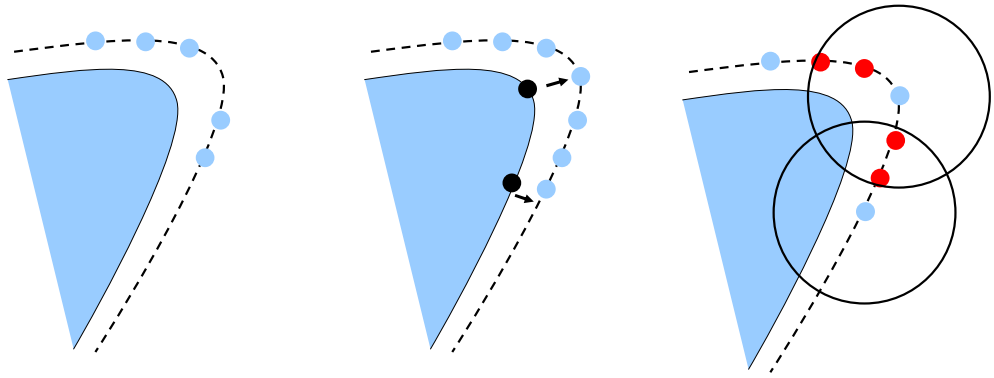
5.4.2.2. Quelques remarques à propos de l'inondation contrainte

Nous détaillons à présent quelques propriétés concernant l'inondation à contrainte décrite dans la section précédente. Il nous a semblé important d'insister sur la propriété d'**isotropie**. En effet, il s'agit souvent d'une propriété qu'on « oublie » soit involontairement, soit par souci de « clarté » et « concision ». Si effectivement cette propriété alourdit fortement l'algorithme, elle reste néanmoins propriété fondamentale de l'approche. C'est pourquoi nous jugeons nécessaire de préciser le comportement de l'algorithme proposé vis-à-vis de cette propriété dans la suite. Nous mentionnerons également une différence importante par rapport à la *lpe* classique: le swamping. Enfin nous émettrons quelques remarques sur le contenu de la *fah*.

Comportement sur les plateaux L'isotropie se manifeste naturellement sur les plateaux. Selon l'intuition, le type de contrainte locale que nous utilisons dans l'algorithme devrait donner à la propagation des propriétés de distance euclidienne.

Considérons donc un relief plat, avec comme labels de départ ceux de la figure 5.20(a) : nous souhaitons évidemment que les deux carrés de la figure évoluent de manière complètement symétrique. Les étapes de l'inondation sont données sur la figure 5.20. Nous remarquons sur cette figure que les deux labels se propagent de manière exactement symétrique et que l'isotropie ainsi que la symétrie sont respectés. Nous remarquons également que lors de la propagation, le front d'onde n'est pas totalement lisse comme il était souhaité au départ. Les premières étapes de la propagation montrent d'ailleurs une influence de l'élément structurant utilisé. Puisque les propagations se font de manière *atomique* et indivisible, il est en fin de compte normal d'avoir une incidence de la forme de l'élément structurant.

Sur la figure 5.21, nous montrons une approche flux de l'algorithme telle que nous l'avions envisagée initialement. Dans cette approche, les priorités des points voisins dans la boule de contrainte sont mises à jour immédiatement lors de la labellisation des points sortant de la file. Les points nouvellement découverts sont également insérés dans la file, avec jetons. Le jeton est un mécanisme bloquant, permettant de reporter à la fin du traitement du plateau courant l'insertion effective des nouveaux points. Il émane des remarques faites pour l'algorithme classique de *lpe*. Il va de soi que sans ce jeton, l'algorithme se conduit de manière totalement anarchique, favorisant une direction ou une autre de manière arbitraire. Malgré le mécanisme de jeton, la mise à jour de l'image de priorité pendant la labellisation conduit à des anisotropies comme le montre la figure 5.21. Un des labels possède alors une priorité plus faible que l'autre, et le traitement ne se consacre qu'à l'un deux. Nous remarquons



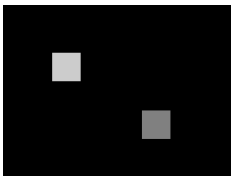
(a) Configuration initiale (b) Découverte de nouveaux points (c) Mise à jour locale

Fig. 5.19.: Propagation et mise à jour locale de la priorité des points. À partir de la configuration initiale (a), de nouveaux points (en bleu sur (b)) sont ajoutés. Ceci modifie la configuration localement, et la priorité des voisins à l'intérieur de la boule de contrainte doit être mise à jour (en rouge sur (c)).

par contre que les lignes de front sont plus lisses. Malgré cette propriété attractive, nous n'avons pas retenu cette solution.

Swamping Toujours par analogie au cas classique, l'algorithme de *lpe* effectue un swamping du relief lorsque les sources ne correspondent pas aux minima locaux. Ce swamping peut être effectué soit par modification du relief initial par une simple reconstruction, soit directement sur une astuce de la file d'attente, en insérant un point à la priorité « priorité_courante $\vee$ priorité_du_point ». Cependant, le swamping n'est pas utilisé l'algorithme que nous proposons. La raison de cela tient au fait qu'il est difficile d'effectuer du swamping alors que nous pouvons accélérer ou ralentir le front, en augmentant ou diminuant la priorité.

File d'attente hiérarchique: Les priorités de la file d'attente hiérarchique ne sont plus dans l'espace $\mathbf{F}$ - image - du relief inondé, mais sur un espace plus fin. Si l'on fait abstraction de l'image « imAlpha » de contrôle, le pas le plus fin est donné par $\Delta p = \frac{2}{A_{ball}}$. Pour des images entières, une priorité réelle simple précision est largement suffisante, il est même envisageable d'avoir une file d'attente à priorité discrète, mais étant donné le nombre important de priorités possibles, l'avantage d'une telle représentation reste discutable. L'intervention de l'image « imAlpha » nous force par contre à passer en représentation réelle double précision. Sans une file d'attente à hiérarchie réelle, telle que celle que nous avons présentée en §2.2.6 page 33, il n'est pas possible d'implémenter l'algorithme.



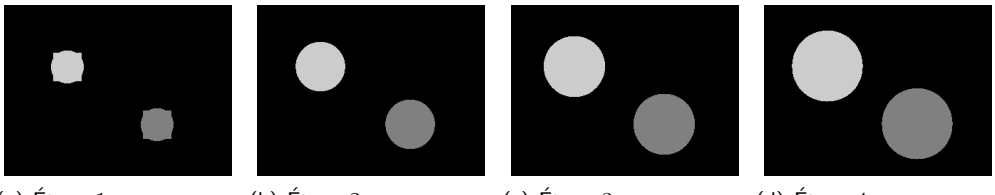
(a) Labels initiaux

(b) Étape 1 (c) Étape 2 (d) Étape 3 (e) Étape 4

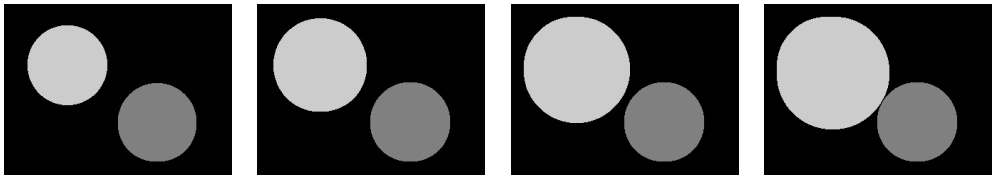
(f) Étape 5 (g) Étape 6 (h) Étape 7 (i) Étape 8

(j) Étape 9 (k) Étape 10 (l) Étape 11 (m) Étape 12

Fig. 5.20.: Illustration de l'isotropie de l'inondation contrainte. (a) : labels initiaux. (b) à (m) : les étapes de la propagation. Les labels se propagent de manière complètement symétrique jusqu'à ce qu'ils entrent en contact, soit entre eux, soit avec le bord de l'image.



(a) Étape 1 (b) Étape 2 (c) Étape 3 (d) Étape 4



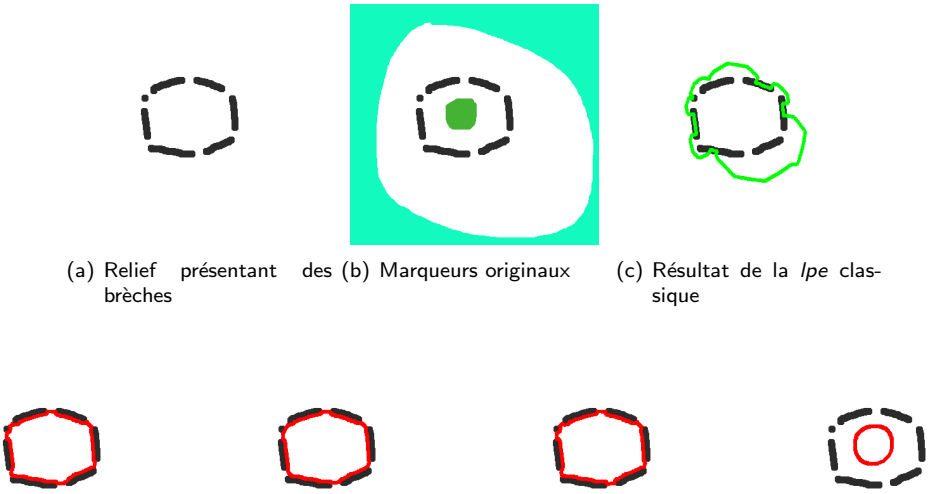
(e) Étape 5 (f) Étape 6 (g) Étape 7 (h) Étape 8

Fig. 5.21.: Illustration de la non-isotropie de l'inondation contrainte. (a) : labels initiaux. (a) à (h) : les étapes de la propagation. (e) : l'ensemble de priorités d'un des labels prime sur l'ensemble des priorités de l'autre. Ensuite seul un des deux est propagé.

5.4.2.3. Résultats

Nous présentons ici essentiellement deux types de résultats: ceux sur des images synthétiques, et ceux sur des images réelles. Dans la suite, l'image « imAlpha », c'est à dire l'image pondérant l'influence de la contrainte locale sur l'inondation, est constante. Les gradients utilisés sont normalisés dans $[0, 1]$.

Images synthétiques Présenter les résultats sur des images synthétiques permet en outre de vérifier le comportement de l'algorithme. La figure 5.22 présente des résultats pour un relief synthétique simple. Comme nous pouvons l'observer, le comportement de l'inondation contrainte suit nos attentes puisque le front du marqueur intérieur s'arrête aux trous du relief. Un paramètre α assez faible (0.2) permet de limiter l'influence de la configuration locale du front. Nous obtenons des résultats absolument équivalents pour $\alpha \in \{10^{-5}, 0.2, 0.6\}$. Pour $\alpha = 0.8$, une partie du contour intérieur se place à l'extérieur de l'ellipse pour un rayon faible.



(a) Relief présentant des brèches (b) Marqueurs originaux (c) Résultat de la *lpe* classique

(d) Résultats pour $\alpha = 0.2$ et $r = 6$ (e) Résultats pour $\alpha = 1.5$ et $r = 6$ (f) Résultats pour $\alpha = 0.2$ et $r = 10$ (g) Résultats pour $\alpha = 1.5$ et $r = 10$

Fig. 5.22.: Illustration de l'inondation visqueuse sur un relief présentant des « brèches » sur ses contours. (a) : relief utilisé pour l'inondation. Il s'agit d'une sorte d'ellipse dont le contour présente des trous. Le relief ne présente qu'une unique valeur. (b) : marqueurs utilisés pour l'inondation. Un marqueur est à l'intérieur de l'ellipse mentionnée, l'autre à l'extérieur. (c) : la *lpe* classique (en vert) traverse le relief sans aucune retenue et suit une distance trame sur le plateau. (d): l'inondation contrainte (en rouge, « imAlpha » constante) ne traverse pas les différents trous du relief. (e): l'augmentation de la valeur de α ne change pas le résultat. La boule utilisée réduit en fait l'effet de la concavité du marqueur extérieur (cf. suite). (f) : même résultat que (d) avec un rayon de contrainte de $r = 10$ pixels. (g): la valeur importante de $\alpha = 1.5$ conduit cette fois à l'échec de la segmentation. Le marqueur extérieur est initialement avantagé puisque sa forme est concave. Son front de propagation possède donc une priorité plus importante que le front du marqueur intérieur, d'où le résultat.

La figure 5.23 présente le résultat pour les mêmes marqueurs que ceux de la figure 5.22, le relief étant cette fois-ci lissé à l'aide d'une convolution par une gaussienne. Le résultat est assez similaire à celui de la figure 5.22, cependant dans ce cas, le marqueur intérieur à l'ellipse donne lieu à une région ressemblant à une ellipse lisse dont les contours sont proches des gradients forts du relief. Nous constatons que dans ce cas, le contour est très lisse ce qui montre l'influence de la contrainte locale.

(a) Relief lisse et watershed classique (b) Résultat $\alpha = 0.2$ et $r = 6$ (c) Résultat $\alpha = 0.2$ et $r = 10$

Fig. 5.23.: Illustration de l'inondation visqueuse sur un relief présentant des « trous » lisses. (a) : relief utilisé pour l'inondation et *lpe* (en vert). Il s'agit du relief de la figure 5.22(a) convolué par une gaussienne. Les marqueurs utilisés pour l'inondation sont les mêmes que ceux de la figure 5.22(b). La *lpe* classique (en vert) traverse une partie du relief. (b) : résultat pour une image « imAlpha » constante avec $\alpha = 0.2$ et un rayon de contrainte de 6 pixels. L'inondation contrainte confine le marqueur intérieur à l'intérieur du contour. (c) : le résultat est très similaire à celui de (b).

Images réelles : Les figures 5.24 et 5.25 présentent deux cas d'utilisation réelle de l'inondation contrainte. Ce genre d'application est à la base du développement de l'inondation à contrainte locale, et sont fidèles à nos attentes.

L'image « Nicolas » en figure 5.24, présente trois zones de fuite : les deux côtés du visage, où le gradient est prononcé mais fin et arrivant rapidement sur des zones très plates (la saturation de la vitre à droite de la personne et le dossier du siège); et la partie haute de la personne où les cheveux et le plafond de l'habitacle possèdent une luminance équivalente.

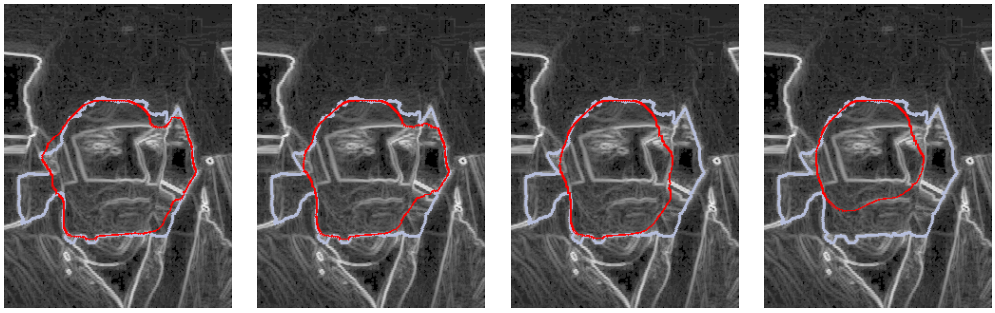
Les marqueurs que nous avons placés pour cet essai sont assez grossiers, mais représentatifs des objets que nous souhaitons séparer. Nous avons cependant fait attention à ce que ce marqueur déborde des deux gradients forts crées par l'ombre portée sur le visage. La *lpe* classique déborde sur les deux zones transversales (la vitre et le dossier du siège) mais s'arrête à la limite entre les cheveux et le visage, où le gradient est suffisamment prononcé pour empêcher les fuites.

L'inondation contrainte présente des résultats différents, mais conformes à ce que nous cherchons à obtenir. Deux tailles de boule de contrainte sont illustrées : 6 et 10 pixels. Pour une boule de contrainte petite en 5.24(c), l'inondation s'infiltre plus rapidement dans les petites structures ce qui entraîne le débordement sur la zone vitrée. Par contre, le débordement sur la zone du siège est évité, ce qui signifie que cette zone présente une fuite de taille faible. Les résultats sont toutefois meilleurs pour une taille de boule de 10 pixels. Pour $\alpha = 0.6$ en 5.24(d), le débordement sur la zone vitrée n'est pas évité mais une partie de la ligne de séparation suit le menton de la personne, malgré la faiblesse du gradient. Les autres parties sont satisfaisantes, mais expliquées par une *lpe* satisfaisante également.

Pour une pondération de la contrainte légèrement plus importante $\alpha = 0.8$ en 5.24(e), le résultat est encore plus intéressant puisque le débordement de la zone vitrée est évité. Ceci signifie que le gradient présente une fuite large et peu contrastée. Les autres contours sont similaires à ceux de $\alpha = 0.6$. Enfin pour une pondération encore plus importante de la contrainte locale en 5.24(f) la partie basse du visage est perdue. Cela s'explique par le fait que le marqueur intérieur est ralenti par les zones de gradient autour de la bouche de la personne. Le gradient sur le menton étant lui-même assez faible, le marqueur extérieur déborde à l'intérieur du visage.

Sur l'image « Raffi » en figure 5.25, la ligne de partage des eaux donne déjà un résultat assez intéressant. Contrairement au cas précédent, les parties transversales du visage sont contrastées avec l'habitacle. La zone de fuite concerne la partie haute de la tête, où en terme de luminance les cheveux

(a) Image originale (b) Marqueurs

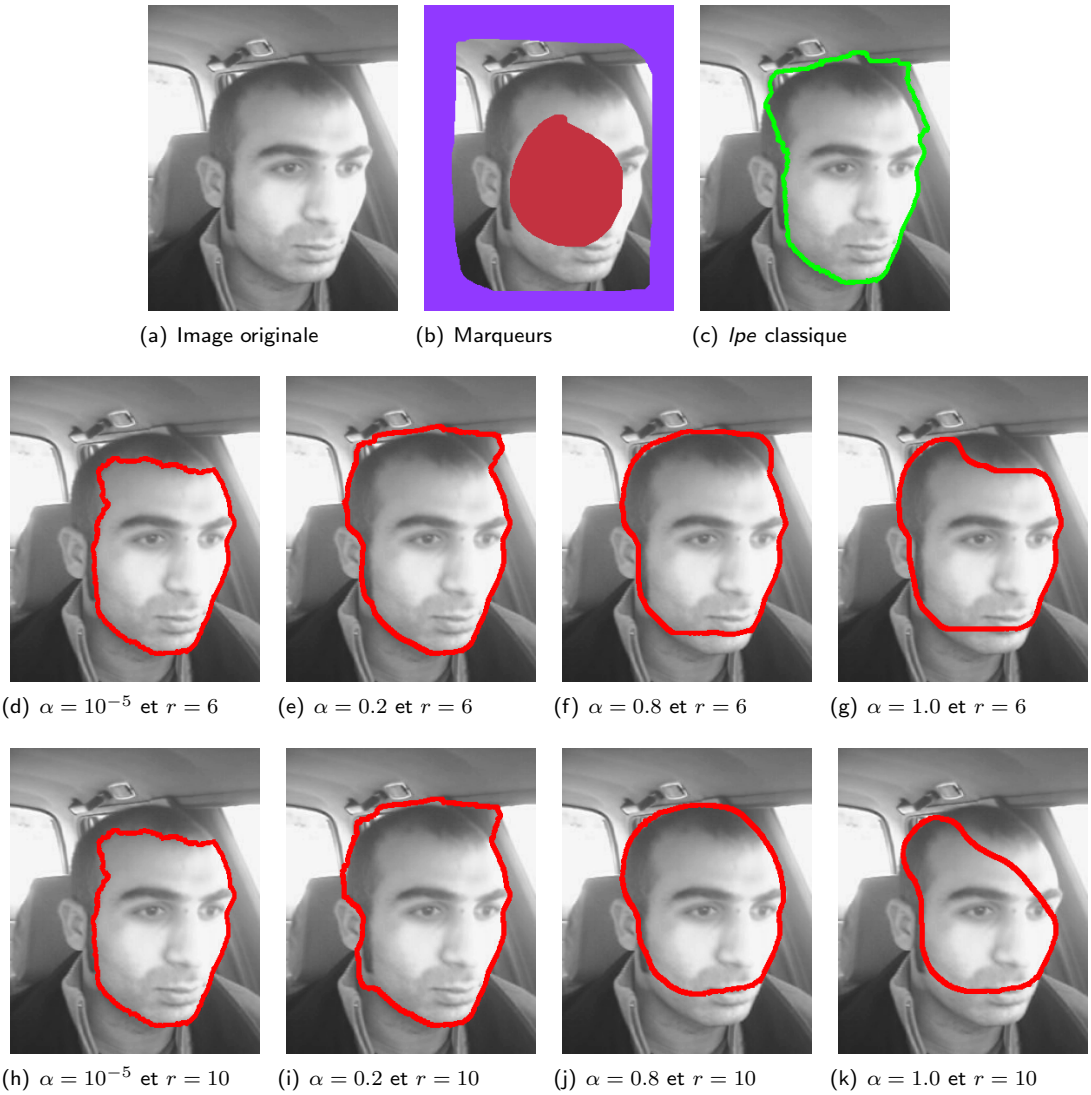


(c) Résultat $\alpha = 0.8$ et $r = 6$ - image gradient (d) Résultat $\alpha = 0.6$ et $r = 10$ - image gradient (e) Résultat $\alpha = 0.8$ et $r = 10$ - image gradient (f) Résultat $\alpha = 1.0$ et $r = 10$ - image gradient

Fig. 5.24.: Illustration de l'inondation contrainte sur l'image « Nicolas ». En bleu le résultat de la ligne de partage des eaux classique à partir des marqueurs de (b), sur le gradient de (a).

et le plafond présentent un contraste faible. Pour les quatre résultats présentés, la partie basse du visage est moins bien segmentée que pour la *Ipe* classique. Ceci s'explique de la même manière que précédemment: les gradients autour de la bouche ralentissent le marqueur intérieur, ce qui permet au marqueur extérieur de s'infiltrer dans le visage. L'influence de ces gradients est toutefois moins importante pour $\alpha = 0.8$ et un rayon petit ($r = 6$) de contrainte (figure 5.25(f)). Le fait que la structure de contrainte soit petite permet au front d'avancer malgré la présence des irrégularités autour de la bouche. Notons toutefois que le marqueur extérieur de départ possède une zone importante assez plate au niveau du cou, et un contraste faible avec le menton, ce qui lui permet « d'avancer » plus vite. Pour cette taille de boule de contrainte, la fuite au niveau des cheveux n'est pas évitée. Avec une pondération plus importante de la contrainte ($\alpha = 1.0$ en figure 5.25(g)) les résultats sont assez similaires pour la zone du menton. Le front du marqueur intérieur s'arrête par contre au niveau de la zone de contraste visage-cheveux: le marqueur intérieur s'arrête et le marqueur extérieur pénètre dans la zone de cheveux. Pour une boule plus grande et $\alpha = 0.8$ (figure 5.25(j)), la zone de fuite des cheveux est évitée et la ligne de séparation est très correcte. La zone du menton subit les mêmes problèmes que précédemment, puisque le front intérieur avance encore moins vite que pour la petite boule de contrainte. Une influence plus grande de la pondération (figure 5.25(k)) conduit à une accélération du front extérieur au niveau des cheveux à cause de la concavité du marqueur. Ce marqueur pénètre ensuite dans le visage, alors que le front intérieur semble être ralenti par le contraste entre les cheveux et le visage.

Sur la figure 5.26, nous constatons d'excellents résultats. À partir d'un marquage assez grossier (un marqueur intérieur à la fleur et un autre extérieur, figure 5.26(b)), nous obtenons d'excellentes segmentations pour un grand nombre de α différents et pour les deux tailles de boule mentionnées précédemment. Par ailleurs, plus α est faible, plus les pétales de la fleurs sont bien détourées. La *Ipe* classique présente également un bon résultat (figure 5.26(c)) mais le contour s'arrête sur l'ombre portée sur la fleur car possédant un gradient significatif. Il y a curieusement une zone de fuite pour deux



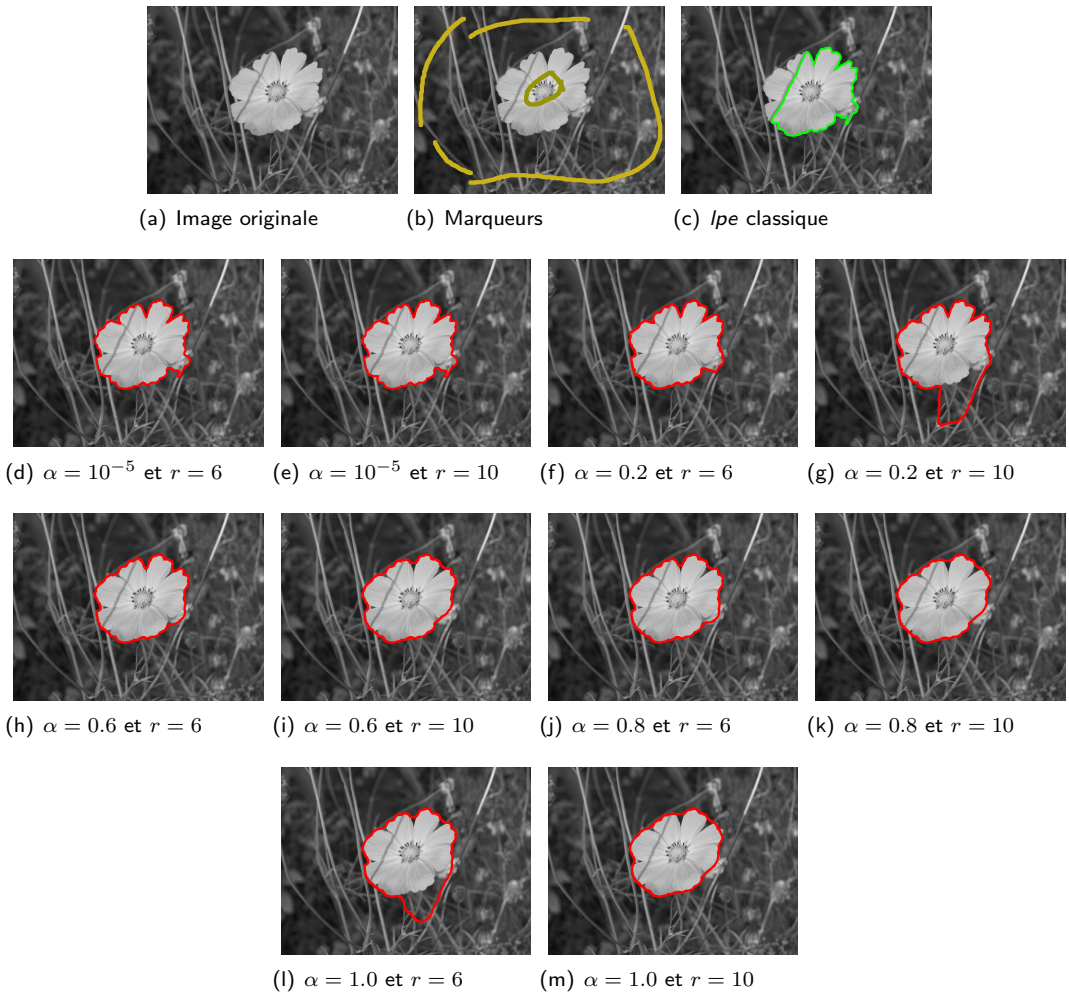
(a) Image originale (b) Marqueurs (c) *Ipe* classique

(d) $\alpha = 10^{-5}$ et $r = 6$ (e) $\alpha = 0.2$ et $r = 6$ (f) $\alpha = 0.8$ et $r = 6$ (g) $\alpha = 1.0$ et $r = 6$

(h) $\alpha = 10^{-5}$ et $r = 10$ (i) $\alpha = 0.2$ et $r = 10$ (j) $\alpha = 0.8$ et $r = 10$ (k) $\alpha = 1.0$ et $r = 10$

Fig. 5.25.: Illustration de l'inondation contrainte sur l'image « Raffi ». (c) En vert le résultat de la ligne de partage des eaux classique à partir des marqueurs de (b), sur le gradient de (a). (d) – (k) Inondation à contrainte locale pour différents rayons r et poids α.

couples de paramètres, pour lesquels nous avons un débordement sur la partie basse de la fleur. Après analyse, cela provient plutôt de l'incapacité du marqueur extérieur à avancer le long de la tige (partie basse de la fleur).



(a) Image originale (b) Marqueurs (c) *Ipe* classique

(d) $\alpha = 10^{-5}$ et $r = 6$ (e) $\alpha = 10^{-5}$ et $r = 10$ (f) $\alpha = 0.2$ et $r = 6$ (g) $\alpha = 0.2$ et $r = 10$

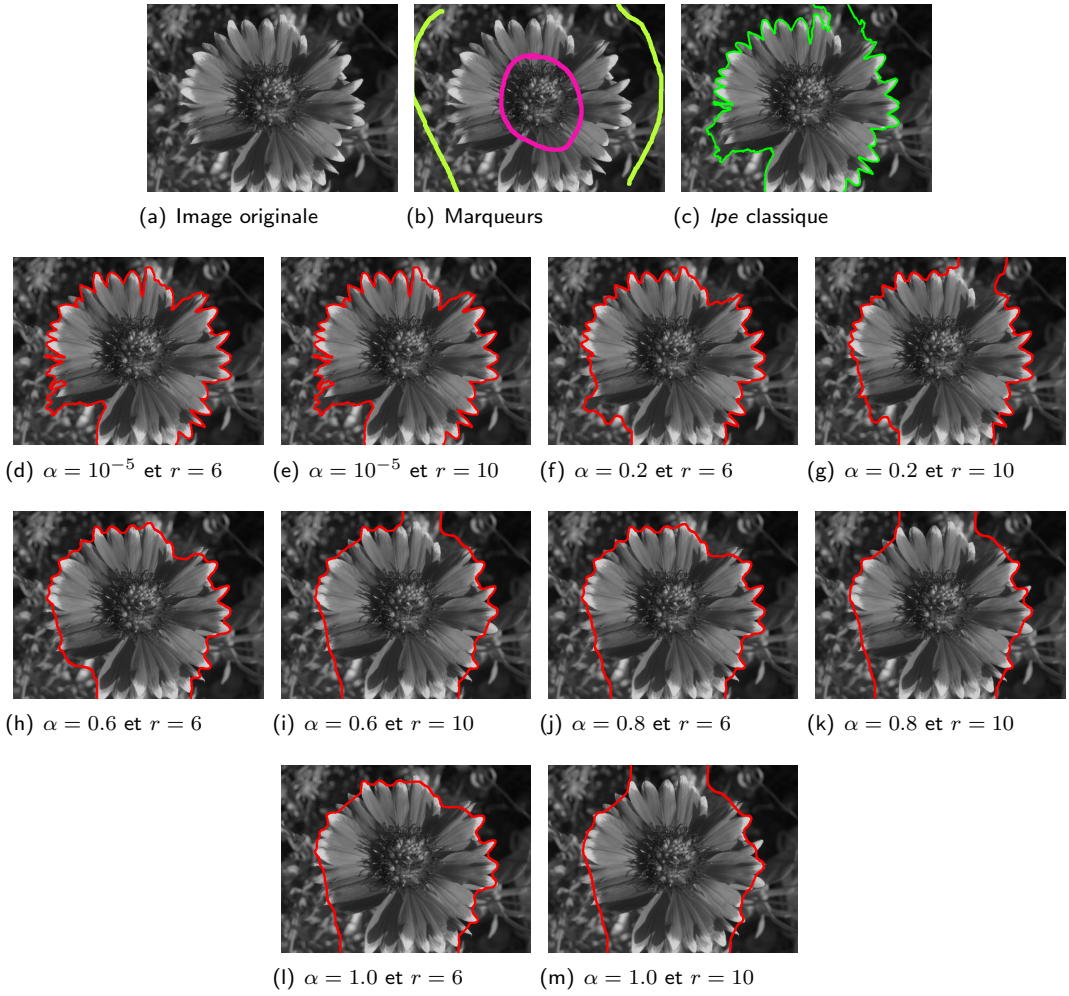
(h) $\alpha = 0.6$ et $r = 6$ (i) $\alpha = 0.6$ et $r = 10$ (j) $\alpha = 0.8$ et $r = 6$ (k) $\alpha = 0.8$ et $r = 10$

(l) $\alpha = 1.0$ et $r = 6$ (m) $\alpha = 1.0$ et $r = 10$

Fig. 5.26.: Illustration de l'inondation visqueuse sur l'image « fleur roumanie » (même légende que sur la figure 5.25).

Nous observons des résultats également très satisfaisants sur la figure 5.27. L'image originale présente deux zones de fuites: la partie haute-droite et basse de la fleur. La *Ipe* classique déborde sur la partie haute, mais se comporte bien sur la fuite basse. L'inondation visqueuse se maintient très bien à l'intérieur de la fleur pour des valeurs de α petites. Lorsque α croît, une boule de contrainte petite permet au front de ne pas s'échapper à l'extérieur de la fleur. Mais ceci encore une fois est dû au relief vu par le marqueur extérieur : une structure dont le gradient est important est présente sur le front du marqueur droit, et l'empêche ainsi d'avancer.

Conclusion : Les résultats présentés sont pertinents et assez conformes à ce que nous souhaitions. Nous remarquons toutefois que cette motivation soit teintée par la difficulté que nous avons parfois à paramétrer correctement les tailles de boule et l'influence de la contrainte. Le choix n'est effectivement pas trivial: le rayon de la boule de contrainte est à la fois représentatif de la taille des structures de l'image et de la régularité du front. En effet, sur les quelques résultats que nous avons décris, une boule de taille faible permettait au « liquide » de s'infiltrer dans des structures petites alors qu'une boule



(a) Image originale (b) Marqueurs (c) *lpe* classique

(d) $\alpha = 10^{-5}$ et $r = 6$ (e) $\alpha = 10^{-5}$ et $r = 10$ (f) $\alpha = 0.2$ et $r = 6$ (g) $\alpha = 0.2$ et $r = 10$

(h) $\alpha = 0.6$ et $r = 6$ (i) $\alpha = 0.6$ et $r = 10$ (j) $\alpha = 0.8$ et $r = 6$ (k) $\alpha = 0.8$ et $r = 10$

(l) $\alpha = 1.0$ et $r = 6$ (m) $\alpha = 1.0$ et $r = 10$

Fig. 5.27.: Illustration de l'inondation visqueuse sur l'image « fleur Roumanie 2 » (même légende que sur la figure 5.25).

de taille plus importante bloquait la propagation du front dans ces mêmes structures. De la même manière, l'influence du paramètre α (amplitude de la modification de priorité) n'est pas trivialement liée au résultat de la segmentation. Le seul élément que nous avons observé est le suivant: plus α est important, plus le front semble régulier. Il semble donc que les influences de ces deux paramètres ne sont pas « continues », ce qui compromet une optimisation éventuelle du problème par rapport à ces paramètres.

5.4.2.4. Remarques

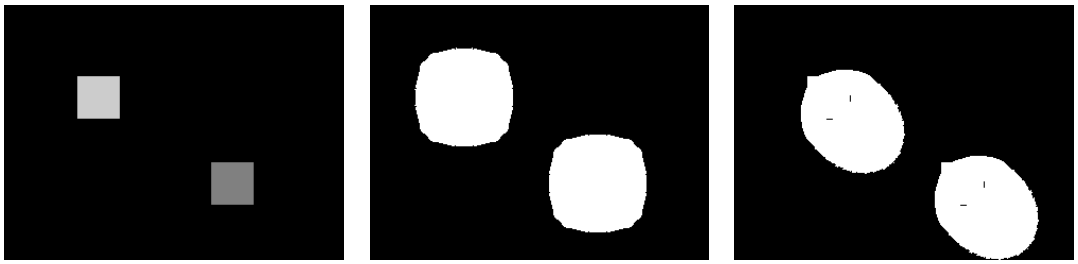
Les remarques sont de différente nature et concernent la terminologique, les cas d'utilisation et enfin les limites de l'approche proposée.

Terminologie Cette remarque n'est pas capitale, mais mérite toutefois d'être énoncée. Le front de propagation évolue sur un relief modifié. Il est difficile de prévoir le relief précisément « vu » par le front. Dans ce sens et bien que dans le principe l'algorithme d'inondation à contrainte locale présente de fortes analogies, il s'écarte de l'algorithme de *ligne de partage des eaux*. La *lpe* est en effet complètement déterminée par le relief et les marqueurs initiaux. La modification que nous avons apportée entraîne le fait que le relief effectivement « vu » par le front d'inondation n'est pas déterminé à l'avance, mais pendant le processus d'inondation.

Sur les plateaux, les distances par rapports aux bords des plateaux et aux temps géodésiques sont respectées. Par contre, dans les configurations où la *lpe* passerait par une ligne de crête, la force locale déplace la position effective des lignes de partage.

Il serait encore pire de vouloir essayer des méthodes de simplification de partition comme les cascades puisque les contours des partitions n'ont pas de sens topographique.

Sensibilité par rapport à l'isotropie de la boule L'algorithme est extrêmement sensible à l'isotropie de la boule utilisée pour le calcul des contraintes locales. La figure 5.28 montre un exemple de biais lorsque la boule utilisée pour les contraintes n'est pas correctement centrée. L'erreur est « minime » (décalage de 1 pixel) mais la conséquence est très importante.



(a) Labels initiaux (b) La boule de contrainte est correctement centrée (c) La boule de contrainte n'est pas correctement centrée

Fig. 5.28.: Illustration du problème d'isotropie liée au centrage de la boule de contrainte. (c) : la boule utilisée pour le calcul des contraintes est décentrée d'une unité vers la droite; cette direction est ensuite favorisée lors de la propagation (il y a plus de points à gauche de la boule, les priorités sont favorisées vers la droite). Les traits noirs visibles dans les parties inondées sont en fait des points ayant été marqués comme étant dans la *fah*. Leur priorité est plus faible que tous les autres points du front et seront traités plus tard. (b) : la propagation correcte à la même étape que (c).

Sensibilité par rapport à la forme de départ des marqueurs L'algorithme privilégie essentiellement la propagation sur les concavités. De ce fait, les formes des marqueurs d'inondation influencent assez la partition finale obtenue lorsque le gradient est plat. Ce cas est flagrant sur l'image 5.22(g) (page 228): le marqueur externe est de forme concave et le marqueur interne est de forme convexe. L'influence de

la viscosité est relativement importante (par rapport aux autres exemples de la même figure). Du fait de la concavité initiale de l'un des marqueurs, la propagation est déséquilibrée: les priorités du front du marqueur concave sont plus importantes que celles du marqueur convexe. Le marqueur convexe reste alors bloqué dans son état initial.

Indépendance face à la dimension L'algorithme proposé est totalement indépendant de la dimension de l'image. Il se distingue ainsi de la plupart des algorithmes agissant sur la régularité du contour des régions, en se basant sur la courbure locale. En effet, la courbure locale est une valeur difficile à estimer. Ainsi, cet algorithme est bien adapté pour des problèmes de segmentation en régions « lisses » sur des images $3D$. Par ailleurs, nous remarquons qu'en l'absence de relief (sur les plateaux) le front de propagation prend une forme proche de celle donnée par la distance euclidienne. Il serait alors intéressant de l'appliquer sur des espaces de classification (de dimension faibles et > 3).

Améliorations possibles L'algorithme présente quelques défauts:

– *Interaction locale dans le calcul des forces:* Le calcul des forces locales présentées présente un biais pour certaines configurations locales. Un exemple de configuration est donné en figure 5.29.

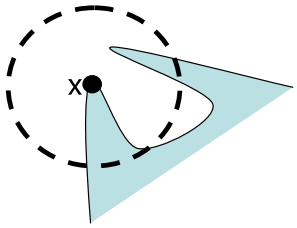


Fig. 5.29.: Problème lié à la mesure sur un voisinage. La mesure utilisée pour la modification de la priorité présente quelques biais dans le cas où plusieurs éléments du label inondant se trouvent dans le $\mathcal{B}$ au voisinage du point x.

– *Calcul des forces pour des sources de petite taille:* La ligne de partage des eaux est très utilisée avec des sources d'inondation très ponctuelles (ie. de l'ordre du pixel). Or notre algorithme présente une dépendance forte à l'initialisation par rapport à la taille des classes inondantes. Si en effet nous l'initialisons avec deux classes d'inondation, selon le cas suivant: la première classe a une source ponctuelle, la seconde possède une source de surface de l'ordre de la boule, alors la deuxième classe est largement privilégiée par rapport à la première. En effet, les points du front de propagation de ce dernier sont accélérés, alors que les points du front de la première classe sont ralentis (le point est considéré comme un pic sur le front d'onde et selon le schéma 5.18(a)).

5.4.3. Expression par EDP et courbure locale

De la même manière que pour les inondations avec information *a priori*, le cadre d'équation d'évolution de courbe à l'aide de EDP est adapté à la formulation de l'algorithme. Cette adéquation découle essentiellement de la modification algorithmique proposée, à savoir la priorité d'insertion des points dans la *fah* selon une mesure locale.

Équation d'évolution Reprenons le cadre des EDP de contours. Soit Γ le contour d'un bassin versant à un instant donné, nous avons:

$$\frac{\partial \Gamma}{\partial t} = \frac{1}{\|\nabla f\| + \alpha \cdot \mathcal{A}_\Gamma}$$

avec α indépendant du temps (constante de pondération), $\mathcal{A}_\Gamma$ la mesure locale modifiant la propagation (dépendante donc de Γ) et f la fonction de relief inondé.

Nous allons voir dans la suite que la mesure $\mathcal{A}_\Gamma$ peut être exprimée comme une fonction de la courbure de Γ. L'intérêt d'une telle démonstration est de mettre en relief une certaine équivalence des deux forces - rapport de surface d'une part et courbure locale d'autre part. La courbure locale est une mesure utilisée assez couramment dans les techniques à base d'EDP, permettant d'imposer une régularité au contour recherché (voir par exemple [Set96]). D'un point de vue numérique, le rapport de surface présente l'avantage d'être plus stable qu'une estimation locale de la courbure.

Courbure locale Nous nous plaçons dans un domaine continu; nous cherchons donc à exprimer $\mathcal{A}_\Gamma$ en fonction de la courbure, ou plus précisément du rayon de courbure de Γ. Soit $\mathbf{x}$ un point du contour Γ, la définition initiale de $\mathcal{A}_\Gamma$ nous permet d'écrire:

$$\begin{aligned} \mathcal{A}_\Gamma(\mathbf{x}) &= \frac{\mathcal{A}(L\cap\mathcal{B}_\mathbf{x})-\mathcal{A}(L^c\cap\mathcal{B}_\mathbf{x})}{\mathcal{A}(\mathcal{B})} \\ &= \frac{2\cdot\mathcal{A}(L\cap\mathcal{B}_\mathbf{x})-\pi\cdot r_\mathcal{B}^2}{\pi\cdot r_\mathcal{B}^2} \\ &= \frac{2}{\pi\cdot r_\mathcal{B}^2}\cdot\mathcal{A}(L\cap\mathcal{B}_\mathbf{x})-1 \end{aligned} \tag{5.12}$$

où L est la zone inondée (l'intérieur de Γ), L^c son complémentaire, $\mathcal{A}$ la fonction de surface, et $\mathcal{B}_\mathbf{x}$ la boule de contrainte centrée en $\mathbf{x}$. Nous notons également $r_\mathcal{B}$ la rayon de la boule $\mathcal{B}$ supposé constant.

Nous avons supposé dans notre algorithme qu'il n'existait qu'une seule composante connexe L dans le voisinage de $\mathcal{B}$. Un contre exemple à cette supposition fut décrit en figure 5.29. Cependant nous ne nous intéressons qu'à la composante connexe de $L\cap\mathcal{B}_\mathbf{x}$ contenant le point $\mathbf{x}$ (ce qui ne correspond pas tout à fait à l'algorithme, et est une hypothèse simplificatrice).

Localement et au premier ordre, le contour Γ peut être approximé par un cercle de rayon de courbure r_Γ; plaçons nous dans le repère de Frenet en $\mathbf{x}$. Par rapport à l'équation 5.12, il faut calculer la surface bleue de la figure 5.30.

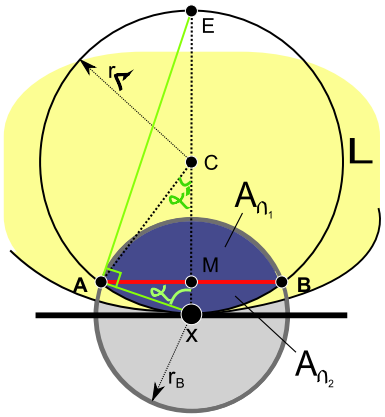


Fig. 5.30.: Détails du calcul du rapport des surfaces. La zone bleue correspond à la force locale de régularisation imposée. L'intérieur de la région, L, est indiquée par la partie jaune.

À partir de la figure 5.30, le disque gris centré en $\mathbf{x}$ représente la boule de contrainte locale. La surface que nous devons calculer est la surface de l'intersection des deux disques, de rayons respectifs $r_\mathcal{B}$ et r_Γ. Notons $\mathcal{A}_\cap = \mathcal{A}(L\cap\mathcal{B})$ la surface de l'intersection en un point $\mathbf{x}$ quelconque de Γ.

Le cercle osculateur de Γ, en noir et de rayon r_Γ, est centré en C. A et B sont les points de l'intersection des deux cercles, M est le milieu du segment $[AB]$, et le segment $[E\mathbf{x}]$ est un diamètre du cercle osculateur. Le triangle $E\mathbf{x}A$ est rectangle en A, l'angle $\alpha = \widehat{E\mathbf{x}A}$ peut être exprimé par son cosinus:

$$\cos\alpha = \frac{r_\mathcal{B}}{2\cdot r_\Gamma}$$

$L\cap\mathcal{B}$ est l'union de deux portions de disque séparées par la corde $[AM]$ (partie bleue sur le schéma). Notons $A_{\cap_1}$ et $A_{\cap_2}$ ces portions et $\mathcal{A}_{\cap_1}$ et $\mathcal{A}_{\cap_2}$ leur surface respective. Nous choisissons $A_{\cap_1}$ comme

étant la portion ne contenant pas le point $\mathbf{x}$. Nous avons $\mathcal{A}_\cap = \mathcal{A}_{\cap_1} + \mathcal{A}_{\cap_2}$. Calculons $\mathcal{A}_{\cap_1}$. Nous avons:

$$\begin{aligned}
\mathcal{A}_{\cap_1} &= \frac{r_\mathcal{B}^2}{2} \cdot 2 \times \widehat{M\mathbf{x}A} - \mathcal{A}(AXB) \\
&= r_\mathcal{B}^2 \cdot \widehat{E\mathbf{x}A} - 2 \times \mathcal{A}(MXB) \\
&= \alpha \cdot r_\mathcal{B}^2 - 2 \times \frac{AM \times MX}{2} \\
&= \alpha \cdot r_\mathcal{B}^2 - (r_\mathcal{B} \cdot \sin\alpha) \times (r_\mathcal{B} \cdot \cos\alpha) \\
&= r_\mathcal{B}^2 \cdot (\alpha - \cos\alpha \cdot \sin\alpha) \qquad (5.13)
\end{aligned}$$

De manière analogue, nous avons $\mathcal{A}_{\cap_2} = r_\Gamma^2 \cdot (\alpha' - \cos\alpha' \cdot \sin\alpha')$, avec $\alpha' = \widehat{\mathbf{x}CA}$. L'expression de α' est dépendante de α; nous pouvons remarquer :

$$\sin\alpha' = \frac{MA}{CA} = \frac{r_\mathcal{B} \cdot \sin\alpha}{r_\Gamma} \qquad (5.14)$$

$$\begin{aligned}
\cos\alpha' &= \frac{MC}{CA} = \frac{XC - MX}{r_\Gamma} \\
&= \frac{r_\Gamma - r_\mathcal{B} \cdot \cos\alpha}{r_\Gamma} = 1 - \frac{r_\mathcal{B}}{r_\Gamma} \cdot \cos\alpha \\
&= 1 - \frac{r_\mathcal{B}^2}{2 \cdot r_\Gamma^2} \qquad (5.15)
\end{aligned}$$

Nous avons ensuite, par développement:

$$\begin{aligned}
\mathcal{A}_\cap &= \mathcal{A}_{\cap_1} + \mathcal{A}_{\cap_2} \\
&= r_\mathcal{B}^2 \cdot (\alpha - \cos\alpha \cdot \sin\alpha) + r_\Gamma^2 \cdot \left(\alpha' - \left(1 - \frac{r_\mathcal{B}^2}{2 \cdot r_\Gamma^2}\right) \cdot \frac{r_\mathcal{B} \cdot \sin\alpha}{r_\Gamma}\right) \\
&= \alpha \cdot r_\mathcal{B}^2 + \alpha' \cdot r_\Gamma^2 - r_\mathcal{B} \cdot \sin\alpha \cdot \left(r_\mathcal{B} \cdot \cos\alpha - r_\Gamma \cdot \left(1 - \frac{r_\mathcal{B}^2}{2 \cdot r_\Gamma^2}\right)\right) \\
&= \alpha \cdot r_\mathcal{B}^2 + \alpha' \cdot r_\Gamma^2 - r_\mathcal{B} \cdot \sin\alpha \cdot \left(\frac{r_\mathcal{B}^2}{2 \cdot r_\Gamma} - \frac{2 \cdot r_\Gamma^2 - r_\mathcal{B}^2}{2 \cdot r_\Gamma}\right) \\
&= \alpha \cdot r_\mathcal{B}^2 + \alpha' \cdot r_\Gamma^2 - r_\mathcal{B} \cdot r_\Gamma \cdot \sin\alpha \cdot \frac{2 \cdot r_\Gamma^2}{2 \cdot r_\Gamma} \\
&= \alpha \cdot r_\mathcal{B}^2 + \alpha' \cdot r_\Gamma^2 - r_\mathcal{B} \cdot r_\Gamma \cdot \sin\alpha \qquad (5.16)
\end{aligned}$$

Par ailleurs, le triangle CEA est isocèle en C et le triangle $\mathbf{x}EA$ est rectangle en A. Nous obtenons:

$$\begin{aligned}
\alpha' &= \pi - \widehat{ECA} = \pi - \left(\pi - 2 \cdot \widehat{CEA}\right) && \textit{(triangle } CEA \textit{ isocèle en } C\textit{)} \\
&= 2 \cdot \left(\tfrac{\pi}{2} - \alpha\right) && \textit{(triangle } \mathbf{x}EA \textit{ rectangle en } A\textit{)} \\
&= \pi - 2 \cdot \alpha
\end{aligned}$$

Enfin, nous avons:

$$\mathcal{A}_\cap = \alpha \cdot r_\mathcal{B}^2 + (\pi - 2\alpha) \cdot r_\Gamma^2 - r_\mathcal{B} \cdot r_\Gamma \cdot \sin\alpha \qquad (5.17)$$

Rappelons que nous mesurons $\mathcal{A}_\cap$ dans l'algorithme lors de la propagation et que nous souhaitions exprimer le rayon de courbure du front de propagation, donné par r_Γ dans l'équation 5.17 précédente. Cependant, il ne nous a pas paru possible d'exprimer sous forme analytique r_Γ en fonction de $\mathcal{A}_\cap$ et $r_\mathcal{B}$. Nous pouvons toutefois avoir une idée de cette fonction en traçant le graphe de $\mathcal{A}_\cap$ en fonction de r_Γ. Ce résultat est présenté en figure 5.31. Nous remarquons par ailleurs que pour des valeurs petites de $r_\mathcal{B}$, de faibles variations de $\mathcal{A}_\cap$ entraînent de grandes variations pour r_Γ, ce qui va dans la direction de l'intuition : il serait difficile de déterminer correctement le rayon de courbure à partir d'une très petite observation. Inversement pour $r_\mathcal{B}$ grand, le calcul de la surface d'intersection comptabiliserait des points sur une très large fenêtre du front de propagation, ce qui mettrait à défaut la validité du modèle que nous avons utilisé.

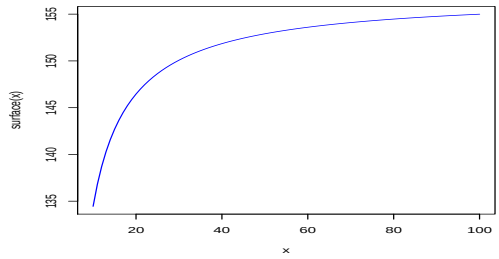


Fig. 5.31.: Surface en fonction du rayon de courbure, pour $r_{\mathcal{B}}$ constant. Sur les axes d'abscisses et d'ordonnés respectivement r_{Γ} et $\mathcal{A}_{\cap}$. Pour cet exemple, nous avons pris $r_{\mathcal{B}} = 10$.

Discussion: Comme le montre l'équation 5.17, l'expression du rayon de courbure du front de propagation en fonction du rapport des surface n'est pas triviale. Cependant nous avons vu dans cette partie qu'il existe effectivement une fonction exprimant la courbure locale en fonction du rapport de surface.

5.4.4. Conclusion

Nous avons vu dans cette section la modification de la *lpe* originale de manière à prendre en compte une configuration locale du front de propagation.

Résumé

Nous avons introduit l'origine du problème, qui provient naturellement d'une limitation de la *lpe* originale. Nous avons ensuite formulé une solution à l'aide de contraintes locales. Ces contraintes ont été mises en œuvre par une mesure simple de rapport de surface, entre une boule de contrainte et la zone inondée. Nous avons également formulé l'algorithme sous forme d'EDP.

L'algorithme est d'implémentation délicate, c'est pourquoi une description exhaustive a été faite. De même, nous avons mis en relief un certain nombre de biais liés soit à des défauts d'implémentation, soit à une mauvaise configuration. Nous avons également souligné certaines limites de l'algorithme présenté.

Nous avons ensuite présenté les résultats, qui sont assez conformes à nos attentes. Sur les images synthétiques, nous avons souligné l'isotropie de l'algorithme proposé. Sur les images réelles, nous avons mesuré sur quelques exemples l'impact de l'attache à la contrainte locale.

Enfin, nous avons exprimé la contrainte locale à l'aide du rayon de courbure du front de propagation. Nous n'avons pas trouvé de solution analytique au rayon de courbure du front (en fonction du rapport de surface). Cette étude nous a permis de rapprocher la méthode proposée de celle, plus classique, utilisant la courbure locale du front comme facteur de régulation.

Perspectives

Modification du lissage selon la pertinence du gradient : Dans Vachier & Meyer [MV02], les auteurs modifient la taille du lissage en fonction de l'altitude du relief. Ceci est une conséquence immédiate de la remarque suivante: un gradient fort possède plus de pertinence, que la propagation se doit de respecter. Les gradients forts sont alors très peu modifiés, alors que les gradients faibles sont fortement lissés. La méthode que les auteurs utilisent pour modifier le lissage selon l'altitude est assez simple: ils emploient des fermetures de taille décroissante avec l'altitude. Seulement les biais de cette approche sont assez similaires à ceux que nous avons énoncés précédemment, les interactions spatiales du relief inondé sont assez difficiles à résoudre. Si l'on revient à notre algorithme, cette approche est assez facile à mettre en œuvre, en évitant encore une fois les biais de [MV02]. D'un

point de vue informatique, nous nourrissons l'algorithme avec une structure différente de l'élément structurant « boule » initialement utilisé. Il s'agit d'un élément structurant qui est un agrégat de boules de différentes tailles : lorsque l'algorithme centre ce nouvel élément structurant, la fonction de centrage choisit, de manière adéquate, la boule adaptée au site (fonction du gradient). Ainsi l'algorithme n'est pas modifié, puisqu'il voit un élément structurant avec les même accès. Grâce aux itérateurs, le parcours de chacune des boules de contrainte est le même. Enfin, les approches objet et méta-programmation nous permettent d'une part de cacher les détails d'implémentation de cet élément structurant, d'autre part d'utiliser le même algorithme pour des éléments structurants très divers - pourvu qu'ils proposent les mêmes méthodes d'accès.

Néanmoins, varier la taille de la boule des contraintes met à défaut la méthode de mise à jour du voisinage (lorsqu'il y a modification locale de configuration). C'est pourquoi nous ne présentons pas les résultats.

Cartes de contrainte : Pour les résultats que nous avons présentés, nous avons considéré α constant. Rappelons que ce paramètre pondère l'influence de la configuration locale par rapport au gradient. L'algorithme pourtant ne considère pas α comme constant, mais comme une donnée ponctuelle (par l'intermédiaire de « imAlpha »).

Il est possible d'ajouter de l'information sur la carte des contraintes « imAlpha », en adaptant la contrainte de manière locale. Voici quelques stratégies envisageables:

Variance locale du relief : Lorsqu'une zone du relief est soumise à de grandes variations, nous pouvons supposer que la zone en question présente les irrégularités que nous souhaitons précisément lisser. « imAlpha » pourrait être une mesure locale de la variance du gradient.

Lissage de gradient : Les zones dans lesquelles le gradient est faible doivent être soumises davantage à l'influence de la contrainte locale. C'est en effet sur ces zones (plateaux) que la propagation doit être contrôlée (zone sur laquelle le front déborde à partir d'une brèche par exemple). Il serait alors possible de préparer « imAlpha » à partir d'une convolution du gradient original par une gaussienne G:

$$\mathcal{I}_{\text{imAlpha}} = 1 - G * |\nabla \mathcal{I}|$$

Dispersion couleur: Nous avons développé dans la partie §4.1 sur la couleur (page 119) une mesure de dispersion locale que l'on a assimilée à un gradient. Il serait pertinent de voir s'il est possible d'utiliser cet outil dans la perspective d'ajustement des contraintes. Nous avons vu que de faibles contraintes sur certaines zones non-homogènes (cf. résultats sur l'image « Nicolas »: le coté droit du visage et la vitre sur la figure 5.24(d)) conduisaient à des fuites. Il serait intéressant de rehausser « imAlpha » sur les zones non homogènes,de la manière suivante:

$$\mathcal{I}_{\text{imAlpha}} = |\nabla_{chp} \mathcal{I}|$$

où ∇_{chp} est l'opérateur de gradient circulaire d'homogénéité pondéré par la saturation (cf. équation 4.18 page 121), et $\mathcal{I}$ l'image originale couleur, exprimée dans l'espace HLS.

5.5 Conclusion sur les inondations

Dans ce chapitre, un tour assez général autour de l'algorithme de *ligne de partage des eaux* a été effectué. Nous avons tout d'abord introduit le cadre théorique de la *lpe*. Nous avons également rapproché le processus d'inondation d'une équation d'évolution de contours basée sur des dérivées partielles (EDP), ce qui nous a permis par la suite d'exprimer les modifications de l'inondation par analogies.

Ce cadre établi, nous avons revu l'algorithme original de *lpe* à base de files d'attente hiérarchiques de Meyer [Mey91, Mey95]et en avons souligné les biais. Dans la section §5.2.2, nous avons proposé une correction de ces biais en proposant une *lpe* épaisse. Nous avons également rapproché ces développements du cadre informatique général introduit au chapitre 2. L'algorithme proposé fonctionne en effet sur des images de dimension quelconque, et sur des reliefs de nature quelconque également (précision réelle, vectoriel, etc.).

L'algorithme de propagation peut être facilement étendu de manière à considérer, dans la notion de *bassin versant*, d'autre mesures que celles directement liées au relief. Ceci nous a permis d'établir deux nouveaux algorithmes, l'un utilisant des informations sur les points et sur l'intérieur du front (section §5.3), l'autre utilisant une configuration géométrique locale de ce front (section §5.4). Pour ces deux algorithmes, les résultats obtenus sont appréciables: une contrainte d'homogénéité produira une partition dont les régions sont effectivement plus homogènes que pour la *lpe* classique. Une contrainte sur la configuration géométrique du front de propagation produira une partition en régions de contour lisse, et cohérente à un voisinage de taille plus grande que l'unité.

Enfin, de nombreux travaux ont été laissés en perspective. Ces travaux sont des variations directes des algorithmes proposés: par exemple de nouvelles fonctions de coûts, des influences locales des différentes contraintes, etc.. Pour toutes ces variation, la modification n'est pas au niveau de l'algorithme mais au niveau des dépendances de l'algorithme. Ces dépendances sont soit de nouvelles structures logicielles (les fonctions de coût par exemple), soit des données externes (les images d'influence des contraintes « imAlpha »), et n'impliquent pas des modifications des algorithmes. Ces perspectives montrent par ailleurs, indirectement, la flexibilité de la méthodologie proposée (variation des priorités selon des mesures), et la puissance du cadre informatique introduit au chapitre 2. Enfin, l'approche par EDP de contours donne un cadre cohérent aux modifications algorithmiques proposées.

CHAPITRE 6

Conclusion générale

Fig. 6.1.: Une illustration de Oleg Delgadcshov

« *Maude Lebowski* - Lord. You can imagine where it goes from here.
The Dude - He fixes the cable? »
The Big Lebowski, Ethan & Joel Coen

6.1 Rappel des objectifs

Dans ce manuscrit, nous avons exposé des outils nouveaux de conceptions d'algorithme de Morphologie Mathématique. Nous avons appuyé cette présentation sur la description d'une librairie de Morphologie Mathématique utilisant l'approche de programmation objet et la méta-programmation.

Dans le chapitre 2, nous avons effectué la liaison entre les outils mathématiques et algorithmiques utilisées, et les structures informatiques de données non-typés. Cette liaison a permis entre autre d'abstraire informatiquement les images, les graphes, les sous-ensembles, les voisinages, et les relations d'ordre. Nous avons également proposé une classe de file d'attente générique, structure par nature liée

au traitement algorithmique, ne nécessitant que la définition d'une relation d'ordre faible. L'ensemble de ces développements a permis d'obtenir un système algorithmique fortement indépendant de la représentation des images, en coordonnée (position des points) et en valeur (valeur des pixels).

Nous avons discuté dans le chapitre 3 des fonctions de distance, sous deux points de vue. Le premier est le calcul de distances exactes, pour des fonctions distance vérifiant un critère de convexité et d'homogénéité. Ces critères sont indépendant de la dimension et permettent une application à une variété très étendue de fonctions distances. Un algorithme utilisant les développements théoriques, et tirant profit de la généricité du système algorithmique a également été proposé, et des résultats de distance ont été montrés en dimension 4.
La deuxième partie du chapitre discute des transformations résiduelles et plus précisément de la *quasi-distance*, récemment définie par Beucher [Beu05] comme étant une extension scalaire des distances binaires morphologiques classiques. Nous avons proposé un algorithme pour le calcul rapide de cette fonction distance. Nous avons profité du formalisme initialement introduit par Beucher pour étendre les transformations résiduelles sur des images couleurs, à l'aide de relations d'ordre lexicographiques et de métriques couleur. Nous avons ensuite utilisé ce nouveau formalisme pour les quasi-distances appliquées aux images couleurs.

Le chapitre 4 a abordé le problème de la couleur en MM, selon trois aspects différents. Le premier est l'utilisation simple de métriques couleurs. Les métriques nous ont permis de définir un gradient morphologique généralisé. Cependant, dans le but de définir des gradients, une seconde approche basée sur des statistiques locales s'est avérée moins coûteuse et tout aussi efficace. Enfin, la dernière approche est algébrique et explore les relations d'ordre lexicographiques. Nous avons ensuite appliqué ces développements à deux exemples d'applications industrielles concrètes et selon des approches novatrices. La première application concerne la détection de peau dans un contexte présentant de très fortes variation d'illuminant, et sans hypothèse sur les individus. Après une analyse assez fine des lieux de peau dans différents espaces couleur et à partir de nombreuses bases de données différentes, nous avons déduit que le domaine HLS proposait une représentation adaptée à la caractérisation de peau. Nous avons ensuite analysé plus précisément l'effet de la variation d'illuminant dans cet espace, à travers la variance des estimateurs d'un modèle de distribution statistique circulaire. Nous avons également simulé le changement d'illuminant grâce à une méthode colorimétrique.

Le chapitre 5 fut consacré à la problématique de segmentation à travers l'algorithme de ligne de partage des eaux. Nous avons souligné les biais de l'algorithme de *lpe* basé sur des files d'attente hiérarchiques, et en avons proposé les corrections. Nous avons facilement appliqué la *lpe* aux images en dimension 4 et aux images couleurs, grâce à notre système algorithmique. À partir de l'algorithme existant, nous avons proposé quelques modifications ayant pour but de contrôler plus précisément les étapes de l'inondation. Ceci nous a conduit à la définition de deux nouveaux algorithmes. Le premier utilise une information sur les régions, et modifie la propagation selon une fonction. Cette fonction a permis, par exemple, d'améliorer la cohérence du contenu couleur d'une région. La seconde modification concerne la forme locale du front de propagation de l'inondation. Ceci nous a permis en outre de lisser d'une part les contours des régions, et de contrôler plus précisément l'effet de certains artefacts présents dans le relief topographique inondé.

6.2 Perspectives

> « Donner des conseils est toujours bête, mais donner de bons conseils est désastreux. »
> *Aphorismes*, O. Wilde.

Les travaux laissés en perspective restent nombreux. Nous allons reprendre chacun des chapitres et énoncer les points d'amélioration et de perspective.

Méta-programmation

Les outils développés dans cette partie rendent actuellement de nombreux services dans le développement de nouvelles méthodes en MM. Il y a cependant certains points améliorables.

Le goulot d'étranglement principal concerne la vitesse d'exécution des algorithmes utilisant des voisinages. La structure de voisinage étant générique, les calculs élémentaires restent très coûteux. Les possibilités d'amélioration sont les suivantes:

Lien avec une bibliothèque optimisée. Si nous avons déjà essayé avec succès le rapprochement de deux bibliothèques indépendantes, Morphée (dont l'approche principale a été exposée au chapitre 2) et la librairie MorphoMédia (développée dans [Bra06]), ce rapprochement induit néanmoins des coûts de développement qui ne sont pas négligeables. Par ailleurs, nous avons décidé d'utiliser uniquement des langages de haut niveau pour notre librairie, ce qui permet de s'affranchir des problèmes évidents de portabilité de plateforme. Le fait de rester *haut niveau* a bien entendu ses avantages et inconvénients, parmi lesquels nous pouvons citer le contrôle très faible des instructions générées par le compilateur. Pour reprendre, l'utilisation d'une librairie dédiée à une architecture particulière permet très efficacement de contourner ce problème, et est optimale en terme de performances finales; elle n'est par contre que difficilement généralisable à l'ensemble des architectures visées par le projet initial.

Voisinages La structure de voisinage est, comme nous l'avons mentionné, très générique. Nous pourrions même dire « *trop* » générique. Dans l'état actuel des développements, de nombreux tests sont effectués lors du centrage du voisinage. Ces tests contraignent en fait le voisinage sur le support de l'image, ce qui implique la géodésie des traitements (à l'intérieur du masque formé par le support de l'image). Or dans de nombreux cas, ces tests sont inutiles pour la majeure partie des points. Deux possibilités sont envisageables pour améliorer ce traitement:

- Communication entre l'algorithme de voisinage et le voisinage. Le voisinage pourrait être capable de communiquer le support de l'image sur lequel il est totalement inclus. Deux méthodes de centrage seraient alors disponibles: une première méthode dite « générique », qui assurerait la géodésie dans tous les cas grâce aux tests adéquats, et une méthode dite « simplifiée » qui éviterait précisément ces tests pour éviter des calculs inutiles.
- Utilisation de la redondance de voisinage. Un des points forts de la programmation que nous appellerons classique, se trouve dans le fait que nous maîtrisons totalement le contexte d'appel de certaines fonctions, ce qui nous permet de faire des hypothèses sur la manière de parcourir des images. Classiquement, les algorithmes de voisinage unitaires sont très efficaces car ils utilisent la redondance de deux voisinages successifs, afin d'éviter des relectures en mémoire de données déjà connues. C'est précisément ce qui est utilisé dans [Gra93] et à une échelle plus importante dans [Bra06]. Il serait envisageable d'inclure de telles méthodes, pour utiliser cette redondance, pour une classe de voisinage précise.

Distances

Nous distinguons dans le chapitre 3 consacré aux fonctions distances deux sous-chapitres: l'un concernant l'existence de distances exactes en n dimensions, et illustrant l'approche générique dans l'écriture des algorithmes. Nous considérons cet algorithme particulièrement complexe et quelques améliorations, à la fois algorithmiques et en termes de performances, sont facilement implémentables.

Les quasi-distances ont également été traitées et se sont avérées être efficace dans certains domaines. Selon notre approche générique, nous avons tenté d'améliorer son temps de calcul. Il serait possible d'effectuer d'autre type de développement pour améliorer davantage les temps de calcul. Par ailleurs, quelques modifications mineures pourraient contourner des comportements indésirables, comme la restriction de l'influence d'un minimum global sur une zone limitée, la pondération de la fonction de résidu par un facteur décroissant avec les itérations, etc..

Traitement multispectral

• Les travaux concernant l'espace couleur HLS nous ont semblé pertinents. Cependant, cela suppose une transformation préalable de la représentation RGB vers l'espace couleur HLS, ce qui représente un coût de calcul non négligeable. L'espace couleur HLS en norme ℓ^1 nécessite moins de transformations trigonométriques que son homologue classique (transformation géométrique), avec des résultats qui semblent totalement comparables. Partant de cette observation et dans une perspective d'industrialisation, il serait intéressant de capitaliser un certain nombre de traitements directement dans l'espace couleur RGB, de manière à améliorer les temps de traitement et éventuellement de pouvoir câbler ces opérateurs sur une architecture de type FPGA.

• Nous avons plusieurs fois souligné le fait que l'utilisation d'ordres lexicographiques est délicat, et nécessite une très bonne connaissance des données sous-jacentes. Bien que la manipulation soit facilitée par un changement dynamique de l'ordre, c'est-à-dire qu'il est possible de définir dynamiquement l'ordre utilisé par un même algorithme, le nombre de combinaisons croît rapidement avec la dimension de l'espace.

Pour cette raison, il serait opportun d'améliorer cette manipulation. Une idée serait de créer une mesure sur la qualité de l'ordre.

Inondations

Les travaux concernant les inondations ont donné le jour à de nouvelles possibilités que nous n'avons pas eu l'opportunité d'exploiter. Certains travaux ont également été omis par manque de temps.

• Nous n'avons pas évalué les temps de calcul de la *lpe* épaisse que nous avons proposé. Cette donnée est particulièrement intéressante car elle permet de choisir entre un algorithme dégradé mais rapide, si les biais ne sont pas importants, et un algorithme précis mais éventuellement plus lent. Étant donné que nous effectuons plus de tests que l'algorithme initial, il semble évident que nous avons proposé un algorithme plus lent. Mais le facteur de vitesse des deux algorithmes n'est pas du tout évident à déterminer sur les architectures à base de processeur généraliste.

L'application de l'algorithme de *lpe* aux images $4D$ peut être beaucoup plus éloquent. Dans le milieu médical par exemple, la précision des captures augmente et nous avons à notre disposition des images de type $3D$ avec dimension temporelle. Il est certain que de nouveaux problèmes apparaissent par le fait que ces dimensions sont de nature différente, et il serait intéressant d'étudier ce sujet.

Nous nous sommes restreints aux images $4D$, mais l'algorithme fonctionne en dimension quelconque. Une application intéressante aurait été la classification, mais encore une fois de nouveaux problèmes surgissent. Parmi ceux-ci, l'existence d'une grille pour la représentation des images est l'un des plus difficiles à résoudre. En effet, ce dernier consomme une mémoire extrêmement importante et est très mal adaptée aux problèmes de classification actuels. Par exemple, si nous considérons un espace de coordonnées de dimension $100, 2$ voxels sur chaque axe suffit largement à dépasser les capacités actuelles de mémoire (nous aurions alors une image de dimension 2^{100}). L'une des solution serait d'attendre sagement une vingtaine d'année pour avoir les capacités de calcul adéquates. L'autre solution serait de travailler sur une *lpe* disjointe de la représentation de grille. Les travaux envisageables par la perspectives nD sont encore très nombreux.

Concernant les algorithmes d'inondation modifiés, les perspectives sont également très nombreuses.

• Les fonctions de coût utilisées pour les illustrations de l'algorithmes avec information externe sont très simplistes. Une première perspective serait d'évaluer séparément (ie. de manière marginale) diverses fonction de coût et d'observer leur conséquence sur les partitions obtenues. Les fonctions envisageables sont des fonctions concernant la géométrie de la région, la texture, ou encore des fonctions sur la manière dont évolue le front de propagation etc.. Ensuite il serait également intéressant de voir les conséquences d'une combinaison de fonction de coût, combinaison qui pourrait être linéaire ou plus complexe.

• Enfin, les travaux sur ce que nous avons nommé les inondations visqueuses pourraient également être étendus à des espaces de dimension supérieure à 2. Nous n'avons en effet travaillé que sur des images de dimension 2. Un premier effet serait de voir le comportement de l'algorithme sur des images $3D$, et ensuite sur des images $3D + t$ comme pour la *lpe* classique. Encore une fois la dimension temporelle sera plus délicate à gérer.

Annexes

ANNEXE A

La couleur

> « La couleur n'est pas une onde, un pigment ou un colorant, elle n'est ni liquide ni solide, ni opaque, ni transparente ou réfléchissante, c'est une sensation que génère notre cerveau et que notre esprit métamorphose en art. »
>
> *Le revêtement inexistant des choses*, Yves Charnay, Journal des Arts Déco n19, 2001.

Qu'est-ce que la couleur ? Il est difficile pour la plupart d'entre nous de donner un caractère intelligible à cette notion, tant elle fait partie de l'expérience intuitive journalière. Elle a cependant trouvé une modèle parfaitement scientifique.

Nous allons introduire dans cette partie le support théorique nécessaire à la compréhension des traitements que nous allons être amené à utiliser.

Précisons enfin qu'en annexe A.3 sont exposées les principales méthodes de transformation entre espace couleur.

A.1 Cadre physique

Avant d'aller plus en avant, une introduction, selon une approche physique, des ondes lumineuses permettra une compréhension de l'origine des choix numériques sous-jacent. Capable ainsi de comprendre le cheminement des grandeurs physiques, la représentation numérique de la couleur s'inscrit progressivement dans un cadre cohérent et logique.

La lumière La lumière est une radiation contenu dans une certaine gamme de fréquence, appelée spectre du visible. Nous ne nous intéressons pas pour le moment à ce qui n'est pas visible par l'Homme. La lumière est constituée de particules appelée *photons*, de masse nulle, mais caractérisées par l'énergie qu'elles transportent. L'énergie E et la fréquence f du photon sont des notions équivalentes liées par l'équation de *Planck*:

$$E = \hbar f$$

où $\hbar = 6.626176 \cdot 10^{-34}$ est la constante de Planck: plus l'énergie d'un photon est élevée, plus sa fréquence sera importante. La notion d'énergie est importante lorsque l'on s'intéresse à la lumière émise par une certaine matière, mais elle n'interviendra pas dans nos développements.

La quantité de photons arrivant sur une unité de surface détermine la notion de *luminance*, ou d'*intensité lumineuse*. Le déplacement des photons est supposé rectiligne.

Lorsque la lumière est constituée de photons toutes de même fréquence, elle est dite *monochromatique*. De telles sources de lumière sont très rares et leur formation nécessitent un procédure précise.

Le cas général est d'avoir une lumière contenant des photons de fréquence différente. Si on s'intéresse au nombre de photons par « tranche » ou unité de fréquence, nous obtenons ce que nous appelons la densité de puissance spectrale *SPD* [(i)]. La SPD, fonction de la fréquence, décrit totalement une lumière. Enfin, il existe certaines lumières « types », c'est à dire de SPD connue, sur lesquelles nous reviendrons par la suite.

La matière Lorsqu'une lumière rencontre un objet, une réaction complexe se produit: non seulement la trajectoire des photons est modifiée, mais également leur intensité. Le résultat de la réaction est révélatrice des propriétés de l'objet en question. Bien que complexe, l'interaction est décrite par trois phénomènes, qui sont la transmission, la réflexion et l'absorption. L'absorption du photon fait que ce dernier transmet la totalité de son énergie aux atomes de la surface de l'objet. La réflexion et la transmission ne modifient pas la fréquence du photon, seulement sur un certain nombre de photons d'une même fréquence, une partie est réfléchie à l'extérieur de l'objet, l'autre partie est transmise à l'intérieur de celui-ci. Le processus de réflexion renvoi le photon, il lui est donc possible d'arriver jusqu'un observateur. Donc, même si la fréquence d'une lumière monochromatique, c'est à dire sa couleur, n'est pas modifiée, tel n'est pas le cas de son intensité lumineuse.

Lorsque l'on applique ce schéma sur une lumière de SPD définie, une partie de la SPD réfléchie à l'extérieur de l'objet est atténuée, alors qu'une autre est totalement absorbée, ce que l'on peut considérer comme une atténuation totale. La SPD est modifiée et cette modification est caractéristique des propriétés intrinsèques de l'objet. Nous pouvons résumer ce phénomène en disant qu'à chaque onde, la matière répond par un affaiblissement et cet ensemble de réponses constitue la fonction de transfert de l'objet. Cette fonction de transfert est la *réflectance spectrale*. La réflectance spectrale est exprimée comme étant une fonction de la fréquence (ou longueur d'onde); elle est utilisée comme facteur multiplicatif d'une SPD incidente.

Ce que l'on appelle communément *couleur d'un objet* est l'observation de l'interaction lumière-matière, mais la couleur est une notion qui ne possède pas de fondement physique propre à l'objet. Ainsi, une orange apparaît orange parce que sa surface a réfléchi un ensemble d'ondes qui forment la couleur orange, et a absorbé le reste des ondes. Mais cette orange n'apparaîtra pas orange sous une lumière bleue. Son aspect cognitif change, contrairement à sa réflectance spectrale. Ce simple exemple est révélateur de la difficulté que l'on a à caractériser de manière absolue un objet par une simple observation dans des conditions non maîtrisées.

Les récepteurs Les récepteurs sont les instruments de mesure de l'onde lumineuse. Ils extraient toute ou partie de l'information qu'elle contient, la traitent, et transmettent l'information traitée. L'œil est un récepteur très puissant. Il existe des récepteurs électroniques, tels que les caméras, offrant des caractéristiques très différentes, que cela soit au niveau du temps d'intégration, de la bande passante, des éventuelles corrections liées aux artefacts connu de la technologie employée.

Ici encore, il n'y a pas de récepteur parfait: il y a toujours atténuation d'une certaine gamme d'onde au détriment d'une autre. Il est tout à fait possible de considérer le récepteur comme un filtre avec une fonction de transfert caractéristique. Dans certains cas, il est possible de connaître les caractéristiques de réponse des systèmes d'acquisition, ils sont en effet souvent fourni par le constructeur, mais cela concerne la plupart du temps des systèmes précis et complexes servant à faire de réelles mesures. Nous nous placerons dans un cadre général et supposerons que nous n'avons pas accès à ces informations.

L'œil Le système visuel humain est constitué de deux parties: l'œil et le cerveau. Si les mécanismes internes à l'œil sont bien compris, la traitement de l'image transmise au cerveau reste une discipline dans laquelle les conjectures sont nombreuses.

Parmi les éléments constitutifs de l'œil, la surface sur laquelle se projette la lumière transmise au cerveau, appelée *rétine* nous intéresse particulièrement. Elle est formée de plusieurs couches, et est constituée de deux types de cellules photo-réceptives: les bâtonnets et les cônes. Comme nous le verrons par la suite, une étroite liaison existe entre la fonction de ces cellules et les bases de la colorimétrie.

[(i)] « SPD » en anglais, pour *Spectral Power Distribution*

Les bâtonnets sont associés à la perception de l'intensité de la lumière. Plus petit que les cônes, ils sont également environ dix fois plus sensibles. Ils sont disposés principalement sur le contour de l'axe de vision, appelée *fovéa*. Ils interviennent essentiellement dans la vision nocturne, lorsque l'intensité lumineuse n'est pas suffisante pour solliciter les cônes. En vision de jour par contre, il saturent rapidement et ne transmettent aucune information au cerveau, c'est alors que les cônes prennent le relais.

Les cônes sont responsable de la perception de la couleur. Ils sont très concentrés sur la fovéa. Trois types de cône, S, M et L (pour *short*, *medium* et *long*), ont été identifié dans l'œil humain, présentant une courbe de réponse en forme de cloche légèrement dissymétrique autour des fréquences respectives de $440nm$, $545nm$ et $580nm$. Même si les longueurs d'ondes associées à l'*observateur standard* ne sont pas précisément celles-ci, on est amené à penser que les couleurs, tout du moins les couleurs compréhensible par l'Homme, se projettent dans un espace à trois dimensions.

Genèse La mesure d'une onde est une opération complexe, et les interactions sont nombreuses. Les indéterminées sont au nombre de trois: la lumière illuminant la scène, la matière réfléchissante, et enfin le récepteur utilisé. Une image formée dans la mémoire d'un ordinateur passe malheureusement par toutes ces inconnues: le traitement d'image marque la fin de la chaîne d'acquisition. Généralement, les efforts du traitement portent sur la compréhension de l'objet (ie. la matière), il ne faut cependant pas perdre de vue la variabilité de l'environnement.

A.2 Représentation numérique de la couleur

La densité de puissance spectrale est une donnée qui permet d'identifier une onde lumineuse de manière exacte. L'information qu'elle porte est cependant trop riche pour être utilisable directement dans le monde de l'informatique: elle dérive d'une donnée physique qui suppose sa continuité d'une part, et d'autre part sa dimension semble infinie à première vue.

Nous allons émettre un certain nombre de remarques par rapport à tout cela. L'axe directeur de la méthodologie générale est toutefois restreinte par le caractère *humain* de la notion de couleur.

La première remarque concerne la discrétisation selon l'axe des longueurs d'onde. Il est très rare de voir dans la nature des ondes dont la SPD serait un Dirac, c'est à dire une onde extrêmement confinée en terme d'étalement selon l'axe des longueurs d'onde. Les lasers ou certains types de lampe possèdent par exemple un distribution spectrale de ce type, mais ce sont des cas particuliers que nous éviterons dans notre travail. Ainsi, il est tout à fait légitime de considérer la SPD comme un signal discrétisable.

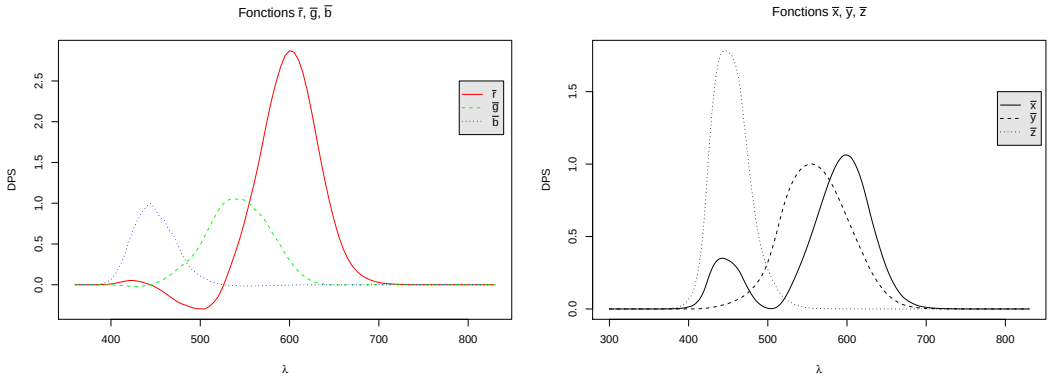
La deuxième remarque concerne la dimension de l'espace. Il est démontré qu'une représentation tri-dimensionnelle permet de synthétiser la gamme de couleurs perçue par l'œil [ST97]. L'information couleur devient alors un vecteur dont les trois composante portent les quantités de rouge, vert et de bleu. Cette déduction n'est pas le fruit d'un hasard mais celui d'une observation purement clinique: l'œil, plus particulièrement la rétine est composée de cônes correspondant approximativement à trois grands type de sensibilités - gammes d'ondes - qui sont précisément le rouge, vert et bleu. Nous n'entrerons pas ici dans les détails des réponses en intensité de ces cônes, le lecteur intéressé peut trouver une mine d'information en consultant les ouvrages [TFMB04, p. 21] et [WS82].

La manière de représenter un vecteur dépend évidement de la base dans laquelle on l'exprime. Pour les descripteurs de couleur , ou vecteurs couleurs, on parle d'« espace couleur ». Il existe évidement un nombre infini d'espaces de représentation, cependant d'après un formalisme et en respect à certaines propriétés, seul un sous-ensemble restreint mais tout de même riche de ces espaces est utilisé.

Nous allons voir dans la suite un certain nombre d'espace couleurs, cette liste est loin d'être exhaustive. Le lecteur trouvera en annexe A.3 les formules de transformation d'espace.

Notons toutefois que d'autres approches que la projection sur une base couleur existent pour le traitement de la couleur. Dans [Geu00], l'auteur travaille directement dans l'espace spectral et utilise des noyaux spatio-fréquentiel gaussiens pour la description locale d'une couleur.

Espaces informatiques et vidéo L'existence de ces espaces est intimement liée aux outils informatiques et vidéo, monde dans lequel ils sont encore largement utilisés. Ils permettent un représentation facile de la couleur au moyen de trois canaux, corrélés ou non. Parmi ces espaces, notons RGB, YUV,



(a) Fondamentales $\overline{r}$, $\overline{g}$, $\overline{b}$ du CIE (b) Fondamentales $\overline{x}$, $\overline{y}$, $\overline{z}$ du CIE

Fig. A.1.: Fondamentales de projection CIE

YIQ, YES... Les espaces RGB sont généralement employés dans les technologies moniteurs, alors que les espaces YUV, YIQ, et YES sont employés en télévision ou en traitement vidéo.

L'existence de l'espace RGB est lié aux premières expériences sur les cônes de visions de l'œil et est largement utilisé en informatique, principalement dans l'affichage et le stockage des images couleurs. Les espaces types Yxx permettent une séparation de l'information couleur en paire luminance-chrominance. Ils sont définis sur des standards différents de luminance et de chrominance. Dans le domaine informatique, ils sont principalement utilisés dans la vidéo et ce pour deux raisons: d'une part ils sont extrêmement simple à calculer, d'autre part, l'œil est moins sensible aux variations de chrominance que de luminance, ce qui permet de stocker une information de chrominance dégradée.

Espaces colorimétriques Il est simple d'imaginer qu'un objet n'aura pas la même couleur sous une lampe à incandescence ou sous la lumière du jour. Si l'on sépare la démarche cognitive du processus de vision, la lumière éclairant une scène conditionne les couleurs perçues par l'œil ou par le capteur. La figure A.2 est un exemple de la variation de température d'illuminant. Les espaces informatiques et vidéo que nous avons décris ne permettent pas de faire une telle distinction, les couleurs qu'ils décrivent sont conditionnés à une lumière précise, et le fait de changer de lumière s'accompagne nécessairement d'une ré-évaluation des couleurs dans ces espaces.

Lorsque l'on utilise uniquement des composantes RGB positives, il est impossible de synthétiser toutes les couleurs. Cette propriété n'est pas immédiate et vient de la démarche expérimentale qui a été mise en œuvre pour appareiller les composantes RGB au système de vision humain: on s'est aperçu que pour certaines couleurs d'une composante, il fallait ajouter de l'intensité aux autres composantes pour obtenir une appariement exacte, et ce indépendamment de la luminosité. Il est alors nécessaire d'utiliser des composantes négatives comme l'illustre le diagramme des fondamentales A.1(a). Cela pose évidement des problèmes de représentation numérique. Cela signifie également qu'il existe une infinité de représentation RGB se projetant dans l'espace XYZ. L'exemple des fondamentales de la figure A.1(a) en est une, l'espace RGB CIE, mais d'autres définitions d'espace existent, comme le sRGB beaucoup utilisé en informatique également, le Adobe RGB... Le tableau A.3.2.1 donné en annexe en reprend quelques uns.

Les espaces dits *colorimétriques*, plus particulièrement l'espace XYZ, apportent une réponse à ces limitations, et proposent une représentation universelle des couleurs. Toutes les couleurs « possibles » y sont représentables. Une nouvelle notion apparaît dans la définition de ces espaces: le blanc de référence, c'est à dire le point qui est considéré comme étant du blanc, et qui peut selon l'environnement contenir une teinte (par exemple rouge: lampe à tungstène, ou bleue: coucher de soleil). L'espace xyY (voir A.3.2.2) est construit à partir de XYZ et permet de projeter la composante chromatique d'une couleur

(a) $2175K$ (b) $2325K$ (c) $2475K$

(d) $2750K$ (e) $2975K$ (f) $3075K$

Fig. A.2.: Exemples de variation de température d'illuminant, issus de la base « The Amsterdam Library of Object Images (ALOI) » [GBS05]. La température de l'illuminant varie, alors que l'objet, la calibration de la caméra et l'angle de vue sont identiques. L'apparence, bien qu'à peine perceptible entre deux variations successives, est assez claire entre les températures extrémales.

sur le plan xy. Cet espace sera utilisé pour des raisons de visualisation essentiellement.

Plusieurs coordonnées de référence de blanc, dits « *standards* », sont données dans le tableau A.2 en annexe. Parmi ceux-ci,par exemple, l'illuminant A correspondrait à la lumière émise par une lampe à incandescence, l'illuminant B se rapprocherait de la lumière du soleil directe, le C à un jour moyen sans UV, le D_{65}, très utilisé, à un jour moyen avec UV. De manière plus générale, on parle souvent de température d'un illuminant: il s'agit de la température du corps noir produisant un rayonnement équivalent à l'illuminant. Le rayonnement du corps noir est donné par:

$$l_T(\lambda) = \frac{2\pi \cdot \hbar \cdot c^2}{\lambda^5 \cdot (e^{\frac{\hbar \cdot c}{k \cdot T \cdot \lambda}} - 1)} \tag{A.1}$$

avec T la température du corps en kelvin, $c = 2.99792458 \cdot 10^8$ la célérité, $k = 1.380662 \cdot 10^{-23}$ la constante de Boltzmann et $\hbar$ la constante de Planck.

Étant donné que le traitement d'image se trouve en fin de chaîne (cf. A.1), effectuer le processus d'estimation du blanc à partir des acquisitions est un problème inverse difficile. C'est la raison « triviale » pour laquelle le blanc est omis dans les traitements et les algorithmes; il est pourtant nécessaire à la validité des résultats de la transformation d'espace et conditionne, comme nous le verrons dans nos applications, le résultat des outils de traitement.

L'espace XYZ fournit un espace « universel » de représentation des informations couleur, mais ne fournit pas de métrique en soi. La métrique serait la capacité perceptuelle humaine à faire la différence entre deux couleurs différentes. Or, *Wright* et *McAdam* ont mis en évidence dans l'espace XYZ des zones (respectivement par hauteur rectilignes et ellipsoïdiques) dans lesquelles les couleurs sont impossibles à différencier. Les segments et les droites n'ont ni la même orientation, ni la même élongation sur l'espace XYZ; ce phénomène implique que les zones de teintes homogènes dans XYZ, c'est à dire des zones dans lesquelles la distance est très proche de 0, sont réparties de manière non-uniforme.

Ces remarques ont conduit la construction d'espaces plus évolués : Lab et Luv. Ces espaces possèdent par construction une métrique euclidienne. Il nous est alors permit de comparer deux couleurs, mais

cette comparaison n'est pas aussi métrique qu'elle ni paraît: le bleu est-il plus proche du rouge que du vert ? Cette question est bien-sûr une vulgarisation, cependant elle met en relief la précaution qu'il faut prendre lors de l'utilisation de cette métrique, qui est plus une métrique de proximité que d'éloignement, dans le sens où il est valide de dire que deux couleurs sont proches lorsque leur distance est petite, mais que le rouge n'est pas plus différent du bleu que du vert (distance grande). Ceci amène à la définition d'un paramètre, le *JND* [(ii)]: un JND inférieur à 1 implique deux couleurs de différence imperceptible, un JND supérieur à 3 deux couleurs très différentes.

Les développements récents de la colorimétrie ([ST97]) s'orientent vers la prise en compte d'un contexte dans la perception de la couleur. En effet, on l'avait déjà mis en relief avec les observateurs standards dits 2° (1931) et 10° (1964), ou ne serait-ce qu'avec la notion de blanc de référence, la perception d'une couleur est conditionnée par l'environnement. Un certain nombre de recherches sont menées à la fois dans l'affinement des métriques (nouvelles distances créée en 1994 pour l'espace Lab, utilisées dans l'industrie textile par exemple), soit dans la conception d'espaces couleurs plus évolués. Mais cela dépasse largement le cadre de notre travail ici.

Espaces interfaces Les espaces dit *interfaces* sont les espaces qui n'ont pas de réalité physique. Ils ont été conçu dans l'esprit de faciliter certain traitements ou pour une manipulation sémantique plus aisée que des espaces purement informatiques. Dans ces espaces nous voyons apparaître des termes liés à une approche subjective humaine.

Luminosité: correspond à l'aspect clair, foncé ou terne d'une couleur. C'est l'attribut de sensation visuelle selon lequel une surface éclairée par une source lumineuse déterminée parait émettre plus ou moins de lumière.

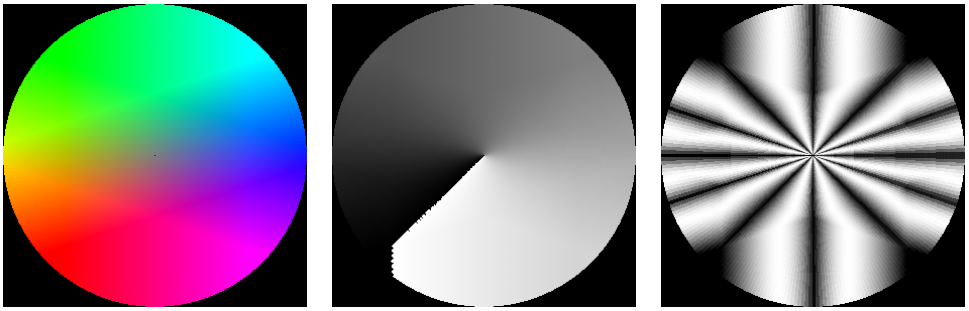
Teinte: ou tonalité chromatique, correspond à la longueur d'onde prédominante présente dans la couleur. Cet attribut de sensation visuelle est à la base des dénominations de couleur.

Saturation: caractérise le coté plus ou moins délavé, pâle ou vif. Cela correspond au degré de mélange de la longueur d'onde prédominante avec le blanc. On peut considérer cela également comme l'attribut de sensation visuelle permettant d'estimer la proportion de couleur chromatiquement pure contenue dans la sensation totale.

La définition de la teinte, saturation et même luminance n'est pas unique. Les espaces les plus connus sont HSV, HLS, HSI, TSL. Le lecteur intéressé trouvera en annexe A.3.3 les opérations de transformation d'espace.

Le modèle HSV est relativement ancien par rapport aux autres, il fut rapidement remplacé par l'espace HLS; et nous ne l'utiliserons pas dans le cadre du manuscrit. L'espace HLS se décline en plusieurs versions: la définition classique A.16 en fait un espace totalement cylindrique malgré le fait que sa structure soit présentée comme celle d'un double cône. La structure de double cône est intéressante dans la mesure où, plus l'on va dans les extrêmes de la luminance (proche de 0 ou de 1), moins la saturation a de l'importance ou de sens. Ce détail prend de l'importance lorsqu'il s'agit d'utiliser les valeurs numérique de la saturation dans un calcul: en effet, pour des valeurs extrêmes de la luminance, la saturation devient de moins en moins précise; le fait de réduire son support (par la structure double conique) limite l'imprécision introduite dans le calcul. La rectification de la définition originale fût donnée par Hanbury [Han02a]. Plus tard, Angulo [AL03] introduit un formalisme sur la métrique utilisée pour la transformation de RGB vers HLS. Plus précisément, cette métrique intervient sur la manière de mesurer la distance d'une couleur par rapport à l'axe achromatique (axe des blancs), et conditionne ainsi la formule utilisée pour le calcul de la saturation. Les propriétés de l'espace en sont également affectées. La figure A.3 montre la différence numériques obtenue à partir des deux définitions de teinte de GHLS et HLS 1 à l'aide de notre logiciel. La différence maximale est proche d'un degré, et sera négligée par la suite.

[(ii)] pour « Just Noticeable Difference »



(a) Image originale (b) Canal de teinte (c) Différence entre GHLS et HLS 1

Fig. A.3.: Différence de teinte entre GHLS et HLS 1 sur une image synthétique. La différence numérique est en partie dûe à l'approximation des fonctions trigonométriques intervenant dans le calcul de GHLS. Elle est de l'ordre de 0.019 sur un espace défini sur $[0, 2\pi[$, ce qui représente une différence d'angle de $1.10°$.

L'utilisation de cet espace est cependant sujet à certaines critiques souvent passées sous silence. Par opposition aux espace colorimétriques, la notion de blanc de référence est totalement absente. Le blanc de référence sert à intégrer l'illuminant de départ, *conjectural*, dans la transformation. Bien que par construction, ce paramètre n'intervient pas dans les transformations de type interface à proprement parler, il serait intéressant de voir quel est l'impact de la modification de l'illuminant sur la transformation. Nous verrons en détail ce point sur l'exemple concret de la modélisation de peau. Ensuite, la construction classique de ces espaces n'est jamais soumise à des correction γ, correction qui, rappelons-le, fait partie de la normalisation de l'espace RGB de départ. Cette remarque a d'ailleurs amené Finlayson [FS01] à construire un nouvel espace, dont la transformation est invariante à ce paramètre. Nous omettrons l'étude exclusive de ce paramètre dans nos travaux.

A.3 Formules de transformation d'espace couleur

Cette partie se veut être un addendum mathématique aux transformations couleurs mentionnées dans le manuscrit. Nous reprenons ici un grand nombre de formules utilisées lors de la transformation entre espaces couleur. Pour les espaces informatiques, les sources sont essentiellement des sources Internet. Pour les transformations colorimétriques, l'ouvrage de référence est [WS82], mais on préférera l'excellent site [Lin] pour la clarté et la disponibilité de l'information. Enfin, pour les espaces géométriques, les sources sont diverses.

A.3.1. Espaces informatiques et vidéos

Ces transformations sont le plus souvent linéaires, et s'obtiennent par opération matricielle sur l'espace RGB de départ. Les transformations étant toutes bijectives, les transformations inverses existent et s'obtiennent par simple inversion matricielle. Voici quelques une de ces matrices:

$$\mathcal{M}_{RGB\to YUV} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.1 \end{bmatrix} \tag{A.2}$$

$$\mathcal{M}_{RGB\to YES} = \begin{bmatrix} 0.253 & 0.684 & 0.063 \\ 0.5 & -0.5 & 0.0 \\ 0.25 & 0.25 & -0.5 \end{bmatrix} \tag{A.3}$$

$$\mathcal{M}_{RGB\to YIQ} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & 0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \tag{A.4}$$

Une formulation généralisée pour le calcul d'un espace couleur RGB vers $YC_{bleu}C_{rouge}$ est donnée par :

$$\begin{cases} Y & = & c_1 \cdot R + c_2 \cdot G + c_3 \cdot B \\ C_{bleu} & = & \frac{B-Y}{2-2c_3} \\ C_{rouge} & = & \frac{R-Y}{2-2c_1} \end{cases} \tag{A.5}$$

dont quelques coefficients standards sont donnés dans la table A.3.1.

A.3.2. Espaces colorimétriques

Comme nous l'avons mentionné dans le chapitre A.2, les espaces RGB ne permettent pas la synthèse de toutes les couleurs possibles. Ceci justifie l'existence d'autre espaces que l'on nomme colorimétriques, dans lesquels l'information couleur est consistante, dans la mesure où la description des couleurs est faite de manière absolue, et qu'elle ne dépend pas d'un choix particulier du contexte d'illumination.

Afin de faire intervenir une notion colorimétrique à une couleur, il est nécessaire de replacer cette information couleur dans un contexte d'illumination, c'est à dire de connaître la couleur de la lumière. Le calcul pour obtenir la matrice de passage est décrit de manière exhaustive dans un grand nombre de manuscrits, dont [WS82]. Il est repris ici dans l'unique but de faciliter le lecteur dans le parcours de ce manuscrit.

A.3.2.1. Conversion vers le domaine XYZ

Puisqu'on ne peut pas synthétiser toutes les couleurs dans l'espace RGB, alors que par définition ceci est possible dans l'espace XYZ, La transformation d'un cube RGB dans l'espace XYZ, bien que transformation linéaire, est injective dans la totalité du domaine de couleurs de l'espace XYZ. Elle devient bien sûr bijective dans le gamut RGB de départ.

Pour effectuer cette conversion, il est nécessaire de connaître l'espace RGB de départ ou à défaut de faire une supposition. Comme nous l'avons mentionné dans A.2, il n'existe pas d'espace RGB en tant que tel, il est plus correct d'y associer le système de coordonnées utilisées. Quelques valeurs de ce système sont mentionnées dans le tableau A.3.2.1.

Ces espaces RGB définissent à la fois les primaires *r,g,b* utilisées pour les coordonnées du gamut dans xyY, ainsi que le blanc de référence de l'espace XYZ d'arrivé. Quelques coordonnées de ces primaires dans des espaces RGB communément utilisés sont repris dans le tableau A.3.2.1.

Notons $\mathcal{M}_{RGB\to XYZ}$ la matrice générique de passage d'un espace RGB vers l'espace XYZ, $(x_i, y_i), i \in R, G, B$ les coordonnées des primaires dans xyY de l'espace RGB concerné[(iii)], $(X_w, Y_w, Z_w)_{XYZ}$ le blanc de référence dans XYZ.

$$\mathcal{M}_{RGB\to XYZ} = \begin{bmatrix} s_r X_r & s_g X_g & s_b X_b \\ s_r Y_r & s_g Y_g & s_b Y_b \\ s_r Z_r & s_g Z_g & s_b Z_b \end{bmatrix} \tag{A.6}$$

avec:

$$\begin{cases} X_i = x_i / y_i \\ Y_i = 1 \\ Z_i = (1 - x_i - y_i)/y_i \end{cases} \quad i \text{ étant r,g ou b} \tag{A.7}$$

et:

$$\begin{bmatrix} s_r \\ s_g \\ s_b \end{bmatrix} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \cdot \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix}^{-1} \tag{A.8}$$

[(iii)] une des dimension n'intervient pas, et seule les coordonnées chromatiques nous intéressent

Nous avons alors:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathcal{M}_{RGB \to XYZ} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{A.9}$$

Les coordonnées (r, g, b) ne sont pas exactement les valeurs numériques comprises dans $\mathbb{N}_{255}$ [(iv)]. Il faut en effet, avant de pouvoir appliquer la transformation A.9, ramener toutes ces valeurs dans l'intervalle $[0, 1]$ et ensuite appliquer une correction gamma sur chacune des composantes[(v)]. Les valeurs de γ dépendent des espaces RGB et sont également données en tableau A.3.2.1. L'espace sRGB est quelque peu différent des autres espaces RGB, la valeur $\gamma \approx 2.2$ n'est qu'une approximation donnant des résultats approchés. Les résultats exacts sont donnés par:

$$c_i = \begin{cases} C_i/12.92 & C_i \leq 0.04045 \\ ((C_i + 0.055)/1.055)^{2.4} & C_i > 0.04045 \end{cases} \tag{A.10}$$

où i indiquant soit le canal rouge, vert ou bleu, c_i le canal après normalisation et C_i le canal avant normalisation dans l'intervalle $[0, 1]$. Le transformation inverse est donnée par:

$$C_i = \begin{cases} 12.92 \cdot c_i & c_i \leq 0.0031308 \\ 1.055 \cdot (c_i)^{1/2.4} - 0.055 & c_i > 0.0031308 \end{cases} \tag{A.11}$$

A.3.2.2. Conversion vers le domaine xyY

Cet espace couleur donne une normalisation des composantes chromatiques tout en conservant l'information d'intensité lumineuse initiale. Il est plus pratique d'utiliser cet espace que l'espace xyz[(vi)], bien que celui-ci fut imposé comme un standard et utilisé dans les ouvrages de référence ([WS82]). Nous avons dans cet espace $Y = 1$ pour tous les blancs de référence.

$$\begin{aligned} x &= \frac{X}{X+Y+Z} \\ y &= \frac{Y}{X+Y+Z} \\ Y_{xyY} &= Y_{XYZ} \end{aligned} \tag{A.12}$$

On prendra les valeurs chromatique (x et y) du blanc de référence dans le cas du « noir » ($X = Y = Z = 0$), avec évidement $Y_{xyY} = 0$. A priori nous pouvons prendre n'importe quelle valeur pour x et y, cela n'a pas d'impact puisque la luminance est nulle. Ce cas doit néanmoins pas se produire puisque le diagramme chromatique n'est pas défini sur la droite $X = Y = 0$.

La transformation inverse se fait naturellement par:

$$\begin{aligned} X &= \frac{xY}{y} \\ Y_{XYZ} &= Y_{xyY} \\ Z &= \frac{(1-x-y)Y}{y} \end{aligned}$$

Dans le cas où $y = 0$, nous prendrons par convention pour valeurs $X = Y = Z = 0$. Encore une fois ce qui compte est d'avoir une luminance nulle $Y_{XYZ} = 0$.

(iv) c'est le cas le plus courant lorsque les composantes couleurs sont codées chacune sur 8 bits
(v) $r = R^{\gamma}$ est la correction gamma sur le canal rouge
(vi) Rappel: les minuscules indiques une normalisation.

A.3.2.3. Conversion vers le domaine Lab

Cette transformation est définie à partir d'un espace de couleur XYZ. Notons $(X_w, Y_w, Z_w)_{XYZ}$ respectivement les coordonnées en X, Y, Z du blanc de référence[vii]. La conversion est donnée par:

$$\begin{aligned} L &= 116 f(y_w) - 16 \\ a &= 500\,(f(x_w) - f(y_w)) \\ b &= 200\,(f(y_w) - f(z_w)) \end{aligned} \tag{A.13}$$

avec:

$$\begin{aligned} x_w &= X/X_w \\ y_w &= Y/Y_w \\ z_w &= Z/Z_w \end{aligned}$$

et :

$$f : a \mapsto \begin{cases} \sqrt[3]{a} & a > \epsilon \\ \frac{\kappa a + 16}{116} & a \leq \epsilon \end{cases}$$

Les constantes κ et ϵ ont pour valeur:

$$\begin{aligned} \epsilon &= 216/24389 \\ \kappa &= 24389/27 \end{aligned}$$

Cette transformation est inversible depuis l'introduction de valeurs rationnelles dans les constantes κ et ϵ[viii]. La transformation inverse est donnée par [ix]:

$$\begin{aligned} x_w &= \begin{cases} f_a^3 & f_a^3 > \epsilon \\ \frac{1}{\kappa}(116 f_a - 16) & f_a^3 \leq \epsilon \end{cases} \\ y_w &= \begin{cases} ((L+16)/116)^3 & L > \kappa \cdot \epsilon \\ L/\kappa & L \leq \kappa \cdot \epsilon \end{cases} \\ z_w &= \begin{cases} f_b^3 & f_b^3 > \epsilon \\ \frac{1}{\kappa}(116 f_b - 16) & f_b^3 \leq \epsilon \end{cases} \end{aligned} \tag{A.14}$$

avec:

$$\begin{aligned} f_a &= \frac{a}{500} - f_L \\ f_b &= f_L - \frac{b}{200} \\ f_L &= \begin{cases} (L+16)/116 & y_w > \epsilon \\ (\kappa y_w + 16)/116 & y_w \leq \epsilon \end{cases} \end{aligned}$$

et enfin:

$$\begin{aligned} X &= x_w \cdot X_w \\ Y &= y_w \cdot Y_w \\ Z &= z_w \cdot Z_w \end{aligned}$$

[vii] Ces coordonnées sont obtenues soit en calculant les valeurs de la transformée dans l'espace XYZ du vecteur $(1,1,1)_{RGB}$, soit en appliquant la transformation xyY vers XYZ des coordonnées chromatiques des blancs de référence donnée dans le tableau A.2

[viii] L'utilisation des valeurs décimales conduisaient à une discontinuité dans un voisinage de ϵ

[ix] Les valeurs de ϵ et κ sont les mêmes que dans l'équation A.13.

A.3.3. Espaces interface

Ces espaces n'ont pas de réelle signification physique. Ils ont été crées dans l'unique but de faciliter les manipulations de couleurs avec un vocabulaire sémantique et intuitif. Il existe principalement deux espaces couleurs de plus en plus appréciés pour leurs propriétés: HLS et HSI. La principale caractéristique de ces espaces est leur géométrie cylindrique, qui impose un certain nombre de difficultés dont nous avons déjà discuté.

A.3.3.1. Espace HSV

Nous n'avons que très peu utilisé cet espace, auquel nous avons préféré HLS. Nous donnons néanmoins les équations de la transformation correspondante:

$$\begin{array}{rcl} v_{hsv} & = & \max(r,g,b) \\ s_{hsv} & = & 1-\min(r,g,b)/v_{hls} \\ h_{hsv} & = & \dfrac{1}{s_{hsv}v_{hsv}} \begin{cases} g-b & \textit{si } r=v_{hsv} \\ b-r+2s_{hsv}v_{hsv} & \textit{si } g=v_{hsv} \\ r-g+4s_{hsv}v_{hsv} & \textit{si } b=v_{hsv} \end{cases} \end{array} \tag{A.15}$$

A.3.3.2. Espace HLS

Il n'existe malheureusement pas de définition universelle de la transformation vers l'espace HLS. La définition classique de cet espace est la suivante:

$$\begin{cases} l_{hls} & = & \frac{1}{2}\left(\max(r,g,b)+\min(r,g,b)\right) \\ s_{hls} & = & \frac{1}{2}(\max(r,g,b)-\min(r,g,b)) * \begin{cases} \frac{1}{l_{hls}} & \textit{si } l_{hls} \leq 0.5 \\ \frac{1}{1-l_{hls}} & \textit{si } l_{hls} > 0.5 \end{cases} \\ h_{hls} & = & \begin{cases} \frac{g-b}{\max(r,g,b)-\min(r,g,b)} & \textit{si } r=\max(r,g,b) \\ \frac{b-r}{\max(r,g,b)-\min(r,g,b)}+2 & \textit{si } g=\max(r,g,b) \\ \frac{r-g}{\max(r,g,b)-\min(r,g,b)}+4 & \textit{si } b=\max(r,g,b) \end{cases} \end{cases} \tag{A.16}$$

Or cet espace, malgré les représentations graphique en forme de cône que l'on peut trouver sur l'espace public, n'est pas conique. Par ailleurs, la division par l'opérateur $\max-\min$ est source d'instabilités numérique lorsque les deux tendent vers 0. Cette instabilité entraîne des variations très importante pour des petites variations des coordonnées initiales. Enfin, un certain nombre de contraintes sont définies dans [AL03] de manière d'un part à assurer la cohérence des calculs que l'on fait sur l'espace HLS d'arrivée, d'autre part la correcte indépendance de l'axe achromatique et du plan chromatique. La définition de l'équation A.16 ne vérifie pas ces contraintes, ce qui a produit un certain nombre de travaux sur une normalisation plus correcte de cet espace.

Concernant la forme cylindrique du cône pour HSV et du double cône pour HLS, Hanbury [Han02b] apporté des modifications aux axes de saturation des ces espaces, de manière à leur rendre leur propriété coniques:

$$\begin{array}{l} s^{con}_{hsv} = s_{hsv}v_{hsv} \\ s^{con}_{hls} = s_{hls}\left[1-2\left|\frac{1}{2}-l_{hls}\right|\right] \end{array} \tag{A.17}$$

Des développements ont été réalisé par Levkowitz [LH93] sur un espace couleur HLS général, appelé *GHLS*, proposant une définition de la teinte de manière trigonométrique. Voici les équations concernant cette transformation:

$$\begin{array}{rcl} l_{ghls} & = & 0.2125r+0.7154g+0.0721b \\ c_1 & = & r-\dfrac{g+b}{2} \\ c_2 & = & \dfrac{\sqrt{3}}{2}(b-g) \end{array}$$

La chroma, donnée par $c = \sqrt{c_1^2 + c_2^2}$, permet ensuite le calcul de la saturation et de la teinte comme suit:

$$h_{ghls} = \begin{cases} \text{non-définie} & c = 0 \\ \arccos \frac{c_1}{c} & c \neq 0 \text{ et } c_2 \leq 0 \\ 2\pi - \arccos \frac{c_1}{c} & c \neq 0 \text{ et } c_2 > 0 \end{cases} \tag{A.18}$$

et enfin la saturation:

$$s_{ghls} = \frac{2}{\sqrt{3}} \cdot c \sin\left(\frac{2\pi}{3} - h'\right) \tag{A.19}$$

avec $h' \equiv h[\frac{\pi}{3}]$.

Angulo et Hanbury se sont ensuite intéressés à la transformation géométrique sous-jacente du cube RGB vers l'espace HLS, et plus particulièrement aux normes utilisées pour la projection du vecteur couleur sur le plan chromatique.

A.3.3.3. Norme ℓ^1

Les travaux d'Angulo [AL03] utilisent cet espace pour mettre en évidence des zones de reflet dans les images. Ceci est par ailleurs significatif de la différence de réponse que présentent ces espaces aux traitements.

La norme ℓ^1 est utilisée pour ses propriétés d'inversibilité. En utilisant le même type de projection sur le plan chromatique, les relations de teinte et de saturation sont :

$$\begin{aligned} l_1 &= \tfrac{1}{3}(max + med + min) \\ s_1 &= \tfrac{3}{2} \cdot \begin{cases} max - l_1 & \textit{si } max + min \geq 2med \\ l_1 - min & \textit{si } max + min < 2med \end{cases} \\ h_1 &= \tfrac{\pi}{3}\left[\lambda + \tfrac{1}{2} - (-1)^\lambda \tfrac{3}{2} \tfrac{l_1 - med}{s_1}\right] \end{aligned} \tag{A.20}$$

avec :

$$\lambda = \begin{cases} 0 & \textit{si } r > g \geq b \\ 1 & \textit{si } g \geq r > b \\ 2 & \textit{si } g > b \geq r \\ 3 & \textit{si } b \geq g > r \\ 4 & \textit{si } b > r \geq g \\ 5 & \textit{si } r \geq b > g \end{cases}$$

λ détermine le quartier du cercle unité sur lequel une couleur se trouve, et on le considère souvent à juste titre comme étant la teinte dominante. Nous noterons $\varphi = \frac{1}{2} - (-1)^\lambda \frac{3}{2} \frac{l_1 - med}{s_1}, \varphi \in [0, 1]$ et avons donc $h_1 = \frac{\pi}{3}[\lambda + \varphi]$.

L'inversion de cette espace se fait d'abord par la détermination de λ et de φ à partir de h_1 et de la fonction $x \mapsto \lfloor x \rfloor$. Il vient:

$$med = l_1 + (-1)^\lambda \frac{2s_1}{3}\left(\varphi - \frac{1}{2}\right)$$

Suivant la valeur de l_1 et de med, nous avons ensuite, par la condition sur s_1 de l'équation A.20 :

$$\begin{aligned} l_1 \geq med \Rightarrow & \begin{cases} max &= l_1 + \frac{2}{3}s_1 \\ min &= 3l_1 - max - med \end{cases} \\ l_1 < med \Rightarrow & \begin{cases} min &= l_1 - \frac{2}{3}s_1 \\ max &= 3l_1 - min - med \end{cases} \end{aligned} \tag{A.21}$$

Nous retrouvons ensuite le même choix concernant le secteur λ et la correspondance entre $\begin{bmatrix} max \\ med \\ min \end{bmatrix}$ et (r, g, b).

Standard	c_1 (rouge)	c_2 (vert)	c_3 (bleu)
Rec.601	0.2989	0.5866	0.1145
Rec.709	0.2126	0.7152	0.0722

Tab. A.1.: Coefficients $YC_{bleu}C_{rouge}$

Illuminant	x	y
A	0.447592	0.407539
B	0.348483	0.351747
C	0.310115	0.316312
D_{50}	0.345669	0.358496
D_{55}	0.332405	0.347552
D_{65}	0.312712	0.329008
D_{75}	0.299091	0.315025
E	$1/3$	$1/3$

Tab. A.2.: Coordonnées chromatiques des blancs de références dans l'espace xyY ($Y = 1$)

Espace rgb (blanc de réference, γ)	$\begin{bmatrix} x_r & y_r & Y_r \\ x_g & y_g & Y_g \\ x_b & y_b & Y_b \end{bmatrix}$						
Adobe rgb (D_{65}, $\gamma = 2.2$)	0.6400	0.3300	0.297361	Ntsc rgb (C, $\gamma = 2.2$)	0.6700	0.3300	0.298839
	0.2100	0.7100	0.627355		0.2100	0.7100	0.586811
	0.1500	0.0600	0.075285		0.1400	0.0800	0.114350
Apple rgb (D_{65}, $\gamma = 1.8$)	0.6250	0.3400	0.244634	Pal/secam rgb (D_{65}, $\gamma = 2.2$)	0.6400	0.3300	0.222021
	0.2800	0.5950	0.672034		0.2900	0.6000	0.706645
	0.1550	0.0700	0.083332		0.1500	0.0600	0.071334
CIE rgb (E, $\gamma = 2.2$)	0.7350	0.2650	0.176204	sRGB (D_{65}, $\gamma \approx 2.2$)	0.6400	0.3300	0.212656
	0.2740	0.7170	0.812985		0.3000	0.6000	0.715158
	0.1670	0.0090	0.010811		0.1500	0.0600	0.072186

Tab. A.3.: Coordonnées de primaires rgb dans l'espace xyY

Bibliographie

Bibliographie

[AL03] Jesús Angulo López: *Morphologie mathématique et indexation couleur. Application à la microscopie en biomédecine*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2003. 107, 108, 111, 119, 177, 206, 254, 259, 260

[Ale01] Andrei Alexandrescu: *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley Professional, première édition, 2001. 12, 13

[ALS03] Jesús Angulo López et Jean Serra: *Color segmentation by ordered mergings*. Dans *Proceedings of the IEEE International Conference on Image Processing*, tome 2, pages 1–4, Barcelone, Espagne, septembre 2003. 111

[AS95] Helmut Alt et Otfried Schwarzkopf: *The Voronoï diagram of curved objects*. Dans *Proceedings of the 11th annual Symposium on Computational Geometry*, pages 89–97. ACM Press, 1995. 50

[Aur91] Franz Aurenhammer: *Voronoï diagrams : a survey of a fundamental geometric data structure*. ACM Computing Surveys, 23(3):345–405, 1991. 50

[BDB06] Jonathan Betser, Sébastien Delest et Romuald Boné: *Segmentation 3D hiérarchique par ligne de partage des eaux sans biais*. Dans *RFIA'06 : Actes du 15^{e} congrès Francophone de Reconnaissances de Formes et Intelligence Artificielle.*, 2006. 190, 200

[BDCK03] Ulrich Bodenhofer, Martine De Cock et Etienne E. Kerre: *Openings and closures of fuzzy preorderings: Theoretical basics and applications to fuzzy rule-based systems*. International Journal of General Systems, 32(4):343–360, 2003. 175

[Bea06] Richard Beare: *A locally constrained watershed transform*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(7):1063–1074, juillet 2006. 181, 183

[Beu91] Serge Beucher: *Segmentation d'images et morphologie mathématique*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, juin 1991. 110, 177, 189

[Beu03] Serge Beucher: *Transformations résiduelles en morphologie numérique*. Rapport technique, Centre de Morphologie Mathématique, École des Mines de Paris, décembre 2003. xi, 47, 67, 70, 73

[Beu04] Serge Beucher: *Algorithmes sans biais de Ligne de Partage des Eaux*. Rapport technique, Centre de Morphologie Mathématique, École des Mines de Paris, avril 2004. 184, 190

[Beu05] Serge Beucher: *Numerical residues*. Dans *Proceedings of the 7th International Symposium on Mathematical Morphology*, pages 23–32, avril 2005. 8, 47, 67, 100, 242

[Bev02] Alessandro Bevilacqua: *A novel background initialization method in visual surveillance*. Dans *Proceedings of IAPR Workshop on Machine Vision Applications (MVA 2002)*, pages 614–617, décembre 2002. 165

[BL79] Serge Beucher et Christian Lantuéjoul: *Use of watersheds in contour detection*. Dans *International Workshop on image processing, real-time edge and motion detection/estimation*. Rennes, France, septembre 1979. 177, 184

[Bra06] Jaromír Brambor: *Algorithmes de la Morphologie Mathématique pour les architectures orientées flux*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2006. 30, 243

[CSS+97] José Crespo, Ronald W. Schafer, Jean Serra, Christophe Gratin et Fernand Meyer: *The flat zone approach: A general low-level region merging segmentation method*. Signal Processing, 62(1):37–60, 1997. 42

[Cui97] Olivier Cuisenaire: *Region growing euclidean distance transforms*. Dans *Proceedings of the 9th International Conference on Image Analysis and Processing*, tome 1, pages 263–270, septembre 1997. 52

[Cui99] Olivier Cuisenaire: *Distance transformations: Fast algorithms and applications to medical image processing*. Thèse de doctorat, Université Catholique de Louvain, Louvain-La-Neuve, Belgique, 1999. 51, 52, 100

[Dan80] Per-Erik Danielsson: *Euclidean distance mapping*. Computer Graphics and Image Processing, 14:227–248, 1980. 51, 52

[DED04] Petr Dokladal, Raffi Enficiaud et Eva Dejnozkova: *Contour-based object tracking with gradient-based contour attraction field*. Dans *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing.*, tome 3, pages 17–20, 2004. 208

[Dej04] Eva Dejnozkova: *Architecture dédiée au traitement d'image basé sur les équations aux dérivées partielle*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2004. 184

[DG98] David F. Dinges et Richard Grace: *"PERCLOS": A valid psychophysiological of alertness as assessed by psychomotor vigilance*. Rapport technique, Federal Highway Administration, 1998. 5

[DH02] Ting-Quan Deng et Henk Heijmans: *Grey-scale morphology based on fuzzy logic*. Journal of Mathematical Imaging and Vision, 16(2):155–171, 2002. 175

[DP02] B.A. Davey et H.A. Priestley: *Introduction to Lattices and Order*. Cambridge University Press, seconde édition, 2002. 28

[FHB03] Ronaldo Fumio Hashimoto et Junior Barrera: *A greedy algorithm for decomposing convex structuring elements*. Journal of Mathematical Imaging and Vision, 18(3):269–289, 2003. 33

[Fri04] Gabriel Fricout: *Propriétés morphologiques et optiques des surfaces rugueuses*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2004. 4

[FS01] Graham Finlayson et Gerald Schaefer: *Hue that is invariant to brightness and gamma*. Dans *British Machine Vision Conference*, 2001. 255

[GBS05] Jan Mark Geusebroek, Gertjan J. Burghouts et Arnold W.M. Smeulders: *The Amsterdam library of object images*. International Journal of Computer Vision, 61(1):103–112, janvier 2005. 253

[Geu00] Jan Mark Geusebroek: *Color and Geometrical Structure in Images*. Thèse de doctorat, Université d'Amsterdam, Pays-Bas, 2000. 251

[GHC04] Nicolas Gourier, Daniela Hall et James L. Crowley: *Estimating face orientation from robust detection of salient facial features*. Dans *Proceedings of POINTING'04: Visual Observation of Deictic Gestures (in association with ICPR 04), Cambridge, Royaume- Uni*, 2004. 142

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides: *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley Professional, première édition, 1994. 13

[GKRR01] Roman Goldenberg, Ron Kimmel, Ehud Rivlin et Michael Rudzsky: *Fast geodesic active contours*. IEEE Transactions on Image Processing, 10(10):1467–1475, 2001. 206

[Gom01] Christina Gomila: *Mise en correspondance de partition en vue de suivi d'objets.* Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2001. 206

[Gra93] Christophe Gratin: *De la représentation des images au traitement morphologique d'images tridimensionnelles*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 1993. 243

[Gri91] Michel Grimaud: *La géodésie numérique en morphologie mathématique. Application à la détection automatique des microcalcifications.* Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 1991. 177

[Han02a] Allan Hanbury: *Morphologie mathématique sur le cercle unité, avec applications aux teintes et aux textures orientées.* Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2002. 105, 111, 119, 174, 254

[Han02b] Allan Hanbury: *The taming of the hue, saturation and brightness colour space. Mathematical morphology in the HLS colour space*. Dans *Proceedings of the 7th Computer Vision Winter Workshop*. Bad Aussee, Autriche, février 2002. 259

[HM06] Allan Hanbury et Beatriz Marcotegui: *Waterfall segmentation of complex scenes*. Dans *7th Asian Conference on Computer Vision*, janvier 2006. 72

[HS01a] Allan Hanbury et Jean Serra: *Mathematical Morphology in the L*a*b* colour space*. Rapport technique, Centre de Morphologie Mathématique, École des Mines de Paris, 2001. 111

[HS01b] Allan Hanbury et Jean Serra: *Morphological operators on the unit circle*. IEEE Transactions on Image Processing, 10(12):1842–1850, 2001. 111

[HS02] Allan Hanbury et Jean Serra: *a 3d-polar coordinate colour representation suitable for image analysis*. Rapport technique, Pattern Recognition and Image Processing group, 2002. 111

[IS99] Marcin Iwanowski et Jean Serra: *Morphological interpolation and color images*. Dans *Proceedings of the 10th IAPR International Conference on Image Analysis and Processing*, pages 50–55, Venise, Italie, 1999. 111

[KLGP95] Jean Claude Klein, Fabrice Lemonnier, M. Gauthier et R. Peyrard: *Hardware implementation of the watershed zone algorithm based on a hierarchical queue structure*. Dans *IEEE Workshop on Non linear Signal and Image Processing*, pages 859–862. Neos Marmaras Malkidiki, Grèce, juin 1995. 178, 184

[Lav05] Nicolas Laveau: *Mouvement et vidéo : estimation, compression, et filtrage morphologique.* Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2005. 23

[LB81] Christian Lantuéjoul et Serge Beucher: *On the use of geodesic metric in image analysis*. Journal of Microscopy, 121:39–49, janvier 1981. 69, 179

[LB05] Nicolas Laveau et Christophe Bernard: *Structuring elements following the optical flow*. Dans *Proceedings of the 7th International Symposium on Mathematical Morphology*, pages 43–52, avril 2005. 23

[LDM05] Romain Lerallut, Étienne Decencière et Fernand Meyer: *Image filtering using morphological amoebas*. Dans *Proceedings of the 7th International Symposium on Mathematical Morphology*, pages 13–22, avril 2005. 24

[Ler06] Romain Lerallut: *Modélisation et Interprétation d'Images à l'aide de Graphes*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2006. 24, 174, 206

[LH93] Haim Levkowitz et Gabor T. Herman: *GLHS: A generalized lightness, hue, and saturation color model*. CVGIP: Graphical Model and Image Processing, 55(4):271–285, 1993. 259

[Lin] Bruce Lindbloom: *http://www.brucelindbloom.com*. 151, 255

[LKP96] Choong Hwan Lee, Jun Sung Kim et Kyu Ho Park: *Automatic human face location in a complex background using motion and color information*. Pattern Recognition, 11(29):1877–1889, 1996. 140

[LM84] Christian Lantuéjoul et Francis Maisonneuve: *Geodesic methods in quantitative image analysis*. Pattern Recognition, 17(2):177–187, 1984. 130, 179

[Mar72] Kanti V. Mardia: *Statistics of directional data*. Academic Press, New York, 1972. 119, 120, 149

[Mar05] Petros Maragos: *PDEs for Morphological Scale-Spaces and Eikonal Applications*, chapitre 4.16. Academic Press, seconde édition, 2005. 182

[Mat69] Georges Matheron: *Théorie des ensembles aléatoires*. Les cahiers du centre de morphologie mathématique de Fontainebleau. Fascicule 4. École des Mines de Paris., 1969. 3, 53, 54, 137

[MBLS01] Jitendra Malik, Serge Belongie, Thomas Leung et Jianbo Shi: *Contour and texture analysis for image segmentation*. International Journal of Computer Vision, 43(1):7–27, 2001. 72

[Mey91] Fernand Meyer: *Un algorithme optimal de ligne de partage des eaux*. Dans *Actes du 8^e Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle*, pages 847–857, 1991. xi, 8, 45, 177, 184, 185, 186, 187, 204, 240

[Mey94] Fernand Meyer: *Topographic distance and watershed lines*. Signal Processing, 38(1):113–125, 1994. 177, 179, 180

[Mey95] Fernand Meyer: *Method for the processing of images by hierarchically organized queues*. US Patent 5463698, 1995. 45, 131, 177, 240

[Mey01] Scott Meyers: *Effective STL: 50 Specific Ways to Improve Your Use of the Standard Template Library*. Addison-Wesley Professional, première édition, 2001. 9, 11, 12

[MI96] Beatriz Marcotegui Iturmendi: *Segmentation de séquences d'images en vue du codage*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 1996. 177, 206

[MJ02] J.Jones Michael et M. Rehg James: *Statistical colour models with application to skin detection*. International Journal of Computer Vision, 46(1):81–96, 2002, ISSN 0920-5691. 141

[Moi97] Lionel Moisan: *Traitement numérique d'images et de films : équations aux dérivées partielles préservant forme et relief*. Thèse de doctorat, Université de Paris-IX - Dauphine, 1997. 206

[MS98] Georges Matheron et Jean Serra: *The birth of mathematical morphology*. Invited conference, ismm, palo alto, june 2000, Centre de Morphologie Mathématique / ENSMP, 1998. CF D-16/02/MM. 3

[MSL01] J. Birgitta Martinkauppi, Maricor N. Soriano et Mika V. Laaksonen: *Behavior of skin color under varying illumination seen by different cameras at different color spaces*. Dans M. A. Hunt (rédacteur): *SPIE Proceedings on Machine Vision Applications in Industrial Inspection IX*, tome 4301, avril 2001. 140, 174

[MV02] Fernand Meyer et Corrine Vachier: *Image segmentation based on viscous flooding simulation*. Dans *International Symposium on Mathematical Morphology*. Sydney, Australia, 2002. 219, 238

[Nac02] Mike Nachtegael: *Vaagmorfologische en vaaglogische filtertechnieken in beeldverwerking*. Thèse de doctorat, Université de Ghent, Belgique, 2002. 175

[Naj94] Laurent Najman: *Morphologie Mathématique: de la Segmentation d'Images à l'Analyse Multivoque*. Thèse de doctorat, Université Paris-Dauphine - U.F.R. Mathématiques de la décision, 1994. 177, 179, 180

[NWvdB03] Hieu Tat Nguyen, Marcel Worring et Rein van den Boomgaard: *Watersnakes: Energy-driven watershed segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(3):330–343, 2003. 177, 179, 181, 182, 183, 219

[PD98] Nikos Paragios et Rachid Deriche: *Geodesic active regions for texture segmentation*. Rapport technique, INRIA Sophia Antipolis, 1998. 206

[Ris01] Valéry Risson: *Application de la morphologie mathématique à l'analyse des conditions d'éclairage des images couleur*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2001. 128

[RM00] J.B.T.M. Roerdink et Arnold Meijster: *The watershed transform: Definitions, algorithms and parallelization strategies*. Fundamenta Informaticae, 41(1-2):187–228, 2000. 178, 179, 181

[RPM01] J. Rozé, T. Pebayle et A. Muzet: *Variations of the level of vigilance and of behavioural activities during simulated automobile driving*. Pergamon, Accident analysis and prevention(33):181–186, 2001. 5

[SAG99] Moritz Störring, Hans J. Andersen et Erik Granum: *Skin colour under changing lighting conditions*. Dans *Proceedings of the 7th Symposium on Intelligent Robotics System*, 1999. 141

[Sch89] Michel Schmitt: *Des algorithmes morphologiques à l'intelligence artificielle*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 1989. 41

[Ser88] Jean Serra: *Image Analysis and Mathematical Morphology - Part 2: theoretical advances*. Accademic Press, London, première édition, 1988. 111

[Set96] JA. Sethian: *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, tome 3 de *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, première édition, 1996. 206, 236

[SM03] Anastasia Sofou et Petros Maragos: *PDE-based modeling of image segmentation using volumic flooding*. Dans *Proceedings of International Conference on Image Processing*, pages 431–434, septembre 2003. 182

[Soi99] Pierre Soille: *Morphological Image Analysis : principles and applications*. Springer, seconde édition, 1999. 6

[SP96] Karin Sobottka et Ioannis Pitas: *Segmentation and tracking of faces in colour images*. Dans *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*, pages 236–241, 1996. 140, 153

[SS95] Philippe Salembier et Jean Serra: *Flat zones filtering, connected operators, and filters by reconstruction*. IEEE Transactions on Image Processing, 4(8):1153–1160, 1995. 42

[ST97] G. Sharma et H. J. Trussell: *Digital color imaging*. IEEE Transactions on Image Processing, 6(7):901–932, 1997. 251, 254

[Stö04] Moritz Störring: *Computer vision and Human skin colour*. Thèse de doctorat, Computer Vision & Media Technology Laboratory, Université d'Aalborg, Danemark, 2004. 141

[Str97] Bjarne Stroustrup: *The C++ Programming Language*. Addison-Wesley Professional, troisième édition, 1997. 9, 12

[TFMB04] Alain Trémeau, Christine Fernandez-Maloigne et Pierre Bonton: *Image Numérique Couleur*. Dunod, première édition, 2004. 105, 251

[TO03] Richard Tsai et Stanley Osher: *Level set methods in image science*. Rapport technique, Computational and Applied Mathematics, janvier 2003. 206

[Vin90] Luc Vincent: *Algorithmes morphologiques à base de files d'attente et de lacets. Extension aux graphes*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 1990. 34

[Vin91] Luc Vincent: *Exact euclidean distance function by chain propagations*. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 520–525, 1991. 52, 53

[Vin93] Luc Vincent: *Morphological grayscale reconstruction in image analysis : applications and efficient algorithms*. IEEE Transactions on Image Processing, 2(2):176–201, 1993. 130, 131

[VS91] Luc Vincent et Pierre Soille: *Watersheds in digital spaces: An efficient algorithm based on immersion simulations*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(6):583–598, 1991. 177, 184

[Wal03] Thomas Walter: *Application de la Morphologie Mathématique au diagnostic de la Rétinopathie Diabétique à partir d'images couleur*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2003. 4

[WS82] Wysecki et Stiles: *Color Science : Concepts and Quantitative Data and Formulae*. John Wiley & Sons, New-York, seconde édition, 1982. 140, 251, 255, 256, 257

[WS02] Power P. Wayne et Johann Schoonees: *Understanding background mixture models for foreground segmentation*. Dans *Image and Vision Computing New Zealand (IVCNZ 2002)*, pages 267–271, novembre 2002. 165

[XP98] Chenyang Xu et J.L. Prince: *Snakes, shapes, and gradient vector flow*. IEEE Transactions on Image Processing, 7(3):359–369, mars 1998. 206

[YKA02] Ming Hsuan Yang, David J. Kriegman et N. Ahuja: *Detecting faces in images: A survey*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(1):34–58, 2002. 140

[Zim91] Hans-Jürgen Zimmermann: *Fuzzy set theory and its application*. Kluwer Academic Publishers, seconde édition, 1991. 175

[ZT01] Maria Francisca Zanoguera-Tous: *Segmentation interactive d'images fixes et de séquences vidéos basée sur des hiérarchies de partitions*. Thèse de doctorat, Centre de Morphologie Mathématique, École des Mines de Paris, 2001. 206