Segmentation interactive d'images fixes et de séquences vidéo basée sur des hiérarchies de partitions

Maria Fransisca Zanoguera Tous

ECOLE DES MINES
DE PARIS

THÈSE

pour obtenir le grade de

Docteur de l'École des Mines de Paris

spécialité « Morphologie Mathématique »

présentée et soutenue publiquement
par

María Francisca ZANOGUERA TOUS

le 13 décembre 2001

Segmentation interactive d'images fixes et de séquences vidéo basée sur des hiérarchies de partitions

---

Directeur de thèse : Fernand MEYER

Jury

Jean SERRA	Président
Touradj EBRAHIMI	Rapporteur
Ferran MARQUÉS	Rapporteur
Bruno BACHIMONT	Examinateur
Sherif MAKRAM-EBEID	Examinateur
Fernand MEYER	Examinateur
Laurent NAJMAN	Examinateur

I John
Diolch am ddisgwyl amdana i

Remerciements

Au moment où je rédige ces lignes, je me sens très heureuse d'avoir fini cette thèse mais, surtout, je me réjouis d'avoir décidé de l'entreprendre. Car, en regardant en arrière, je prends conscience d'avoir parcouru un long chemin, qui a été un chemin de croissance, autant professionnelle que personnelle. Ceci a été possible en grand partie grâce aux gens que j'ai rencontrés. Je remercie tous ceux que j'ai croisé sur mon chemin et qui, d'une manière ou d'une autre, ont contribué au bon déroulement de cette thèse. Il y a cependant des gens que j'aimerais remercier tout particulièrement :

- Mon directeur de thèse, Fernand Meyer, dont la sagesse et le savoir vivre m'inspireront à vie.
- Jean Serra, qui a bien voulu m'accueillir dans son laboratoire.
- Ferran Marqués, qui ne m'a jamais refusé son appui et son aide.
- Cristina Gomila, qui a très généreusement écouté et partagé mes soucis et mes angoisses.
- Beatriz Marcotegui, de qui j'ai énormément appris, et qui a toujours été disponible pour m'aider et m'orienter.
- Etienne Decencière et Beatriz Marcotegui, dont j'ai appris ce que c'est que l'hospitalité.
- Michel et Nicole Bilodeau, Michel pour avoir subi mes problèmes informatiques avec patience et bonne humeur, et tous les deux pour être si sympathiques et accueillants.
- Lilianne Pipault, qui a guidé mes premiers pas à Fontainebleau et m'a tellement simplifié la vie.
- Cathérine Moysan qui avec sa gaieté a rendu mes journées plus agréables, et qui a bien voulu relire mon français encore assez défectueux.
- Marc Waroquier, qui a toujours été prêt à m'aider résoudre toute sorte de problèmes.
- Michel Gauthier, pour sa bonne humeur.
- Lothar Bergen, qui a guidé mes premiers pas avec Xlim.
- Mes collègues du CMM, avec qui j'ai partagé de bons moments et parfois des sorties : Claire-Hélène Demarty, Valéry Risson, Thomas Walter, Jesús Angulo, Allan Hanbury, Christophe Bernard ...
- J'aimerais également remercier tous les gens que j'ai rencontrés au cours du projet MoMuSys, avec qui j'ai passé de très bons moments.

Je remercie très fort ma famille, qui est toujours là pour me soutenir. Finalement, je veux très spécialement rendre hommage à John pour sa patience et son amour inconditionnel.

Résumé

La grande variété des images et séquences vidéo rencontrées dans le domaine multimédia rendent tout projet de segmentation automatique extrêmement complexe. Notre approche cherche à obtenir une segmentation efficace au prix d'un minimum d'interaction.

Pour permettre une grande flexibilité et des temps de réponse rapides, le contenu de la séquence est représenté en forme de partitions emboîtées. Tous les contours possibles dans l'image sont détectés, chacun avec un indice indiquant sa force. L'étape de segmentation proprement dite offrira à l'utilisateur divers mécanismes de sélection finale des contours qui réellement l'intéressent. Ainsi de multiples segmentations sont possibles sur cette représentation hiérarchique, sans nécessiter de nouveaux calculs.

Dans un premier temps, différentes hiérarchies associées aux inondations morphologiques sont étudiées, ainsi que plusieurs mécanismes permettant l'introduction de connaissances a priori quand elles sont disponibles. Dans un deuxième temps, les notions présentées pour les images fixes sont étendues aux séquences vidéo en utilisant une approche 3D-récursive. Ainsi, une unique hiérarchie associée à une séquence vidéo complète est calculée. Des outils d'interaction sont proposés permettant à l'utilisateur de manipuler la hiérarchie de manière intuitive. Grâce aux représentations en forme d'arbre utilisées, la manipulation de la hiérarchie se fait avec un très faible coût de calcul et les résultats de l'interaction sont perçus par l'utilisateur comme étant immédiats.

Abstract

In multimedia applications, the diversity of treated images makes automatic segmentation very difficult. Our approach aims at obtaining good quality segmentations by allowing a minimum amount of user interaction.

In order to achieve high flexibility and fast response times, a video sequence is represented as a set of nested partitions. All possible contours in the image are detected, together with an estimate of their importance. The interactive step allows the user to choose the contours that correspond to the objects of interest. In this way, multiple segmentations are possible without significant recalculations.

Initially, various hierarchies associated to morphological floodings are studied, together with different mechanisms allowing the introduction of a priori knowledge when this knowledge is available. Later on, the notions presented for still images are generalized to the case of video sequences. A single hierarchy representing a video sequence shot is calculated. Several interaction tools are also proposed, that allow the user to manipulate the hierarchy in an intuitive manner. The tree structure chosen to represent the hierarchy allows for an efficient manipulation of the information, and the results of the interaction are perceived by the user as being immediate.

Table des matières

Chapitre 1

Introduction

1.1 Motivation

La segmentation consiste à diviser une image en régions satisfaisant certains critères, la finalité de cette démarche étant différente selon l'application. Par exemple, dans des systèmes de vision par ordinateur la segmentation a pour but de réduire la complexité du problème d'analyse, tandis que dans des applications de codage, elle est utilisée pour réduire le débit.

Par le passé, la segmentation a trouvé beaucoup d'applications dans les milieux industriels et médicaux. Dans ce type d'applications, les images traitées partagent des caractéristiques qui peuvent être exploitées pour la conception d'algorithmes automatiques performants. Autrement dit, pour chaque application on a des connaissances *a priori* sur le contenu des images – par exemple des radiographies d'une partie du corps, des images d'une pièce industrielle, des images d'un type particulier de cellule –, ainsi qu'une définition précise de ce que l'on cherche à segmenter.

Plus récemment, avec le développement des nouvelles technologies de l'information et la disponibilité d'ordinateurs personnels puissants, ainsi que l'apparition d'appareils d'acquisition d'image et de vidéo numériques, de nouveaux besoins se sont développés. Les nouvelles applications de la segmentation s'orientent plutôt vers l'édition multimédia. Citons quelques exemples :

- La disponibilité d'appareils de photographie et de vidéo numériques a créé une demande pour des applications liées à l'édition de ce type de données. La segmentation y joue un rôle très important car elle permet la composition de nouvelles scènes à partir de plusieurs images. La plupart des logiciels d'infographie disponibles sur le marché n'offrent que des outils de segmentation extrêmement élémentaires, l'utilisateur réalisant très souvent les tâches de segmentation de manière totalement manuelle.
- La norme de codage MPEG-4 [36], [77] introduit de nouvelles fonctionnalités basées sur la décomposition de la scène en *objets audiovisuels*. Un objet audiovisuel est défini comme chacune des entités sémantiques constituant une scène. Mais la norme MPEG-4 ne spécifie pas la technique à utiliser pour réaliser cette décomposition. L'utilité pratique de ces fonctionnalités dépendra donc étroitement de la disponibilité d'outils de segmentation performants.
- Le stockage numérique de grandes quantités d'information et le besoin d'indexation n'est plus limité à des documents de texte. La norme MPEG-7 [68] standardise la description de contenu multimédia, ce qui facilitera les recherches sur ce type de données. La norme

définit *comment* le contenu d'une scène est décrit, mais elle ne spécifie pas la manière dont ces informations sont extraites ni les algorithmes de recherche. Ainsi, des nouvelles applications d'indexation et de recherche de données multimédia sont en train d'être développées. Beaucoup de ces techniques nécessitent une décomposition préalable des données multimédia en objets sémantiques, tâche qui doit être réalisée par des algorithmes de segmentation.

Les applications que nous venons de citer partagent un certain nombre de caractéristiques qui les différencient des applications plus traditionnelles telles que les applications industrielles :

- Il n'est pas possible de faire des suppositions sur le contenu des images, et les images traitées dans une même application sont très différentes les unes des autres.
- Les critères de segmentation sont strictement sémantiques. Il n'est pas possible de définir à l'avance les caractéristiques de l'objet à segmenter, comme c'est le cas de la plupart des applications médicales ou industrielles.
- Il ne s'agit plus de segmenter uniquement des images fixes, mais également des séquences vidéo.

L'énorme variété des images traitées, ainsi que la limitation dans le choix de critères non sémantiques de segmentation font de la segmentation automatique un problème extrêmement difficile qui a amené dans les dernières années à des efforts de recherche vers la conception d'algorithmes interactifs. Un algorithme interactif cherche à automatiser au maximum la segmentation tout en profitant des connaissances sémantiques de l'utilisateur. Un bon algorithme interactif devra requérir peu d'interaction et permettre d'obtenir la partition souhaitée en un minimum de temps. Idéalement, et surtout pour les applications destinées au grand public, les mécanismes d'interaction proposés doivent être intuitifs et ne doivent pas nécessiter de connaissances sur le fonctionnement interne de l'algorithme.

D'ailleurs, le traitement de séquences vidéo nécessite le développement de stratégies de mise en correspondance des informations afin de minimiser la quantité d'interaction nécessaire pour segmenter la séquence complète. Il n'est pas en effet envisageable de demander à l'utilisateur de segmenter une séquence vidéo image par image. Ainsi, c'est pour la segmentation de séquences que les gains de temps par rapport à la segmentation manuelle sont les plus importants.

Dans cette thèse, nous présentons un système de segmentation interactive avec les caractéristiques suivantes :

- possibilité de segmenter interactivement des images fixes et des séquences vidéo en réduisant de manière très importante la quantité d'interaction nécessaire par rapport à la segmentation manuelle ;
- le système ne fait pas de supposition sur les caractéristiques des images à segmenter, mais il est possible d'en améliorer les performances si certaines caractéristiques sont connues ;
- les mécanismes d'interaction proposés correspondent à des notions intuitives, rendant le système particulièrement adapté aux applications de type grand public.

1.2 L'approche multiéchelle en segmentation interactive

Nous avons choisi d'utiliser une stratégie multiéchelle pour la segmentation. C'est-à-dire qu'au lieu de trouver une unique partition associée à une image, nous calculons toute une famille de partitions emboîtées qui représentent l'image à différents niveaux de résolution.

Nous verrons que le calcul d'une segmentation multiéchelle peut être réalisé avec un coût additionnel négligeable par rapport au calcul d'une partition unique.

L'approche multiéchelle est particulièrement bien adaptée aux applications interactives multimédia, car on ne connaît pas d'avance les caractéristiques de l'objet d'intérêt. La représentation de l'image à différents niveaux de résolution permet une plus grande flexibilité dans la définition de l'objet. Une fois la famille de partitions calculée, les possibilités de génération de partitions en combinant des régions provenant de différents niveaux sont très nombreuses, par opposition avec les approches basées sur le calcul d'une partition unique.

Les avantages de l'approche multiéchelle se voient multipliés dans le cas des séquences vidéo. Le calcul d'une segmentation multiéchelle associée à une séquence vidéo complète permet à l'utilisateur d'interagir sur une image de la séquence pour obtenir l'objet sur toute la séquence, ce qui réduit de manière très importante la quantité de travail requise de la part de l'utilisateur. Par rapport aux approches traditionnelles de suivi, l'avantage principal de l'approche multiéchelle est la réalisation d'une partie importante des calculs *avant* la définition de l'objet. Ainsi, et en contraste avec les méthodes de suivi, de multiples possibilités de définition des objets sont possibles sans refaire les calculs.

Dans cette thèse, nous présentons une série d'outils permettant à l'utilisateur de manipuler des hiérarchies de partitions emboîtées associées à des images fixes ou à des séquences vidéo pour définir efficacement les objets d'intérêt. Or les outils puissants d'interaction ne garantissent pas à eux seuls la qualité du système obtenu. En effet, si la qualité de la hiérarchie de partitions sous-jacente n'est pas suffisamment bonne, la quantité d'interaction nécessaire pour obtenir l'objet d'intérêt augmente considérablement. Ainsi, une bonne hiérarchie est celle dont les différents niveaux, et en particulier ceux contenant peu de régions, sont visuellement bien adaptés au contenu des images. Nous avons développé des méthodes permettant de construire des hiérarchies adaptées au contenu des images et des séquences vidéo sans pour autant faire de supposition sur leur contenu. Dans des applications particulières où les images partageraient des caractéristiques communes, nous proposons des mécanismes permettant de tenir compte de ces informations lors de la construction de la hiérarchie. La représentation choisie, en forme d'arbre, nous permet un accès très rapide aux informations, les avantages de cette représentation étant d'autant plus remarquables dans le cas des séquences vidéo.

1.3 Plan de la thèse

Nous commençons par faire, dans le chapitre 2, une révision de l'état de l'art en segmentation interactive. Ensuite, le chapitre 3 introduit des concepts mathématiques concernant la construction de hiérarchies, en particulier les notions d'arbre et de distance ultramétrique. Le chapitre 4 présente une batterie d'outils d'édition interactive qui peuvent être associés à une hiérarchie de partitions emboîtées. Nous ne présentons pas dans ce chapitre des détails de mise en œuvre, que nous verrons par la suite dans les chapitres 6 et 7.

Dans les chapitres 5 à 7 nous présentons en détail notre système : le chapitre 5 présente l'étape de pré-traitement, où nous proposons des outils de filtrage et de détection de zones plates sur les images couleur très utiles pour la segmentation. Dans le chapitre 6, nous nous concentrons sur la construction de hiérarchies de partitions emboîtées associées aux images fixes. Nous montrerons que l'utilisation de distances ultramétriques basées sur des processus d'inondation permet de construire des hiérarchies de partitions présentant des contours très satisfaisants pour la vision humaine, même dans les partitions qui comportent peu de régions.

Ces hiérarchies sont représentées sous forme d'arbre aux arêtes valuées, ce qui permet un accès très rapide aux différents niveaux de résolution. Finalement, le chapitre 7 montre comment les concepts présentés pour les images fixes peuvent être étendus au calcul d'une famille de partitions emboîtées associée à une séquence vidéo complète. La représentation résultante élimine la redondance de manière extrêmement efficace, ce qui permet à l'utilisateur de segmenter une séquence vidéo complète avec très peu d'effort. Le chapitre 8 suggère des perspectives industrielles pour les techniques présentées dans cette thèse, et le chapitre 9 donne des conclusions et propose quelques lignes pour de futures recherches.

Les lecteurs n'étant pas familiarisés avec la morphologie mathématique pourront consulter les ouvrages de référence de cette discipline [88], [89], [96].

Chapitre 2

État de l'art en segmentation interactive

2.1 Introduction

Le besoin d'interaction a été au début considéré comme un défaut des algorithmes. Cependant, dans certains contextes d'application tels que le multimédia, la variété des images traitées est telle qu'il est pratiquement impossible avec la technologie actuelle de développer des algorithmes adaptés à toutes les applications et à tous les types d'images. Dans ce type d'applications il existe des ambiguïtés sur ce qui constitue l'objet d'intérêt. Regardons par exemple la scène de la figure 2.1. Selon le contexte, l'objet d'intérêt peut être l'un des locuteurs, un des danseurs au fond ou encore le texte dans le coin inférieur gauche. Pour cette raison, dans des applications génériques l'introduction d'interaction est maintenant largement acceptée comme étant un bon compromis entre la rigidité des algorithmes automatiques et l'énorme quantité de travail requise par la segmentation manuelle. Le développement de nouvelles applications multimédia a de ce fait motivé une concentration des efforts sur la conception d'algorithmes de segmentation interactive.

La qualité d'un système de segmentation interactive est déterminée par deux aspects principaux. Le premier est la qualité de l'algorithme de base. Idéalement, on voudrait un algorithme complètement automatique, mais nous avons vu que ceci est très difficile pour des images tout venant. Le deuxième aspect est le type d'interaction prévue. Cet aspect est souvent négligé mais cependant il détermine en très grande partie l'utilité pratique des systèmes interactifs. Des mécanismes conviviaux, flexibles et rapides sont une condition nécessaire pour le succès d'un système de segmentation interactive.

Les systèmes de segmentation interactive peuvent ainsi être analysés sous deux perspectives différentes :

- l'algorithme de base ;
- la manière dont l'interaction s'intègre dans le processus de segmentation.

Ces deux aspects ne sont en général pas complètement dissociés et, très souvent, l'algorithme de base détermine également les mécanismes d'interaction.

Dans la section 2.2 nous présentons quelques-unes des méthodes de segmentation que l'on trouve en tant qu'algorithmes de base dans des systèmes interactifs. Dans la section 2.3 nous regardons les systèmes interactifs existants du point de vue des mécanismes d'interaction. Dans la section 2.4 nous faisons une révision des systèmes interactifs d'extraction d'objets

vidéo qui ont été proposés. Nous terminons avec une brève discussion dans la section 2.5.

FIG. 2.1 – Scène contenant plusieurs objets à signification sémantique.

2.2 Les algorithmes de segmentation

De très nombreuses méthodes de segmentation ont été proposées dans la littérature. Nous n'allons pas tenter ici une description exhaustive de toutes les techniques existantes, mais nous nous concentrerons sur celles qui sont les plus utilisées dans le domaine de la segmentation d'objets vidéo multimédia, sujet d'étude de cette thèse. Les lecteurs intéressés par des études plus détaillées pourront consulter [17], [27], [73], [81].

Les techniques de segmentation peuvent être classées en deux grands groupes : les techniques basées sur les régions et les techniques basées sur les contours. Chacune de ces méthodes peut être déterministe ou statistique.

Dans les méthodes basées sur les régions, les techniques déterministes cherchent à diviser l'image de façon à minimiser un critère d'homogénéité à l'intérieur des régions et/ou maximiser un critère de dissimilarité entre régions. Les régions ainsi obtenues peuvent être connexes ou non, selon les algorithmes. Dans ce groupe de méthodes on trouve la croissance de régions [60], [52], [14], [43], où des marqueurs choisis au préalable vont croître en attirant les pixels voisins par ordre croissant de différence de niveau de gris ou de couleur. Proche de la croissance de régions est la ligne de partage des eaux (LPE) [7], où l'image est interprétée comme un relief topographique qui est progressivement inondé (cf. annexe A). D'autres techniques déterministes sont le « k-means » [29], [87] ou le « split and merge » (séparer et fusionner) [27], [30], [104]. Le k-means cherche à diviser l'image en un nombre prédéfini de régions et opère par itérations successives, attribuant les pixels aux régions selon leur distance au centre de la région et en mettant à jour le centre pour chaque itération. L'algorithme s'arrête quand il y a peu de changement entre une itération et la suivante. L'algorithme « split and merge » est constitué d'une étape de division en segments réguliers des régions ne satisfaisant pas un critère d'homogénéité, suivie d'une étape de fusion des segments par similarité. Plus récemment, des méthodes basées sur le calcul de coupes minimales ont trouvé leur application en segmentation d'image. Dans ces méthodes, une fonction capacité est définie entre des couples de pixels voisins. La coupe minimale est celle dont la capacité totale est la plus petite [105]. Cependant la coupe minimale est biaisée vers la détection de petites structures. Pour cette raison, des généralisations ont été faites afin d'éliminer ce problème, qui cherchent à minimiser les coupes normalisées [93], [94]. D'autre part, les méthodes statistiques [11] maximisent la probabilité d'appartenance de chaque pixel à la région à laquelle il est attribué, mais pour cela il faut d'abord estimer la fonction de densité de probabilité de chacune des régions.

Les méthodes basées sur les contours cherchent à détecter les zones de transition de l'image. Dans ce type d'algorithmes, les contours actifs ou « snakes » [34] définissent le contour de l'objet comme celui minimisant une certaine fonction d'énergie globale du contour. Cette fonction est composée d'un terme d'énergie interne qui favorise la régularité et fermeture du contour et d'un terme qui représente une force externe définie par le contenu de l'image. Cette méthode a l'avantage de produire des contours très réguliers, mais elle a l'inconvénient de dépendre de plusieurs paramètres ainsi que d'une initialisation correcte de l'algorithme. Les algorithmes de chemin minimum [16] détectent les contours par minimisation d'une fonction de coût qui dépend en général de la valeur du gradient à chaque point. À la différence des contours actifs, les algorithmes de chemin minimum ne considèrent pas les caractéristiques globales du contour. Ceci les rend plus rapides, mais les contours résultants sont moins réguliers. D'autres techniques sont encore basées sur une première détection des zones de transition [9], pour appliquer ensuite un algorithme de fermeture des contours. Bien sûr les techniques par régions peuvent être combinées avec les méthodes de détection de contours afin d'améliorer leurs performances respectives [76], [42].

2.3 Les mécanismes d'interaction

Les systèmes de segmentation interactive proposés sont souvent basés sur des techniques de segmentation automatique existantes sur lesquelles on ajoute une étape d'initialisation et/ou correction des résultats. Ceci n'est qu'une suite logique de la grande quantité de recherche qui a été faite sur la segmentation automatique, mais c'est une approche qui ne permet qu'une interactivité très restreinte et n'exploite pas toutes les possibilités d'un algorithme conçu pour être interactif.

Différentes façons d'interagir ont été proposées dans la littérature, que nous classons ici selon le moment où elles sont appliquées.

Avant exécution : ce type d'interaction peut consister à introduire des paramètres [97], à marquer grossièrement les objets d'intérêt [72], [111], [25] où encore à marquer des régions de l'image ayant des caractéristiques statistiquement semblables à celles des régions composant l'objet [11]. L'interaction avant exécution est peu flexible, puisque l'utilisateur ne peut pas prévoir la réponse de l'algorithme à ses actions. Si cette réponse n'est pas satisfaisante, il est nécessaire de recommencer dès le début.

Pendant exécution : l'utilisateur guide activement l'évolution de l'algorithme. Dans les approches basées sur les contours, l'interaction peut consister à diriger le contour à l'aide de la souris à partir d'un point initial [63], [16] . Le contour est alors calculé entre le point de départ et la position de la souris. En cliquant, l'utilisateur valide un segment de contour et donne le point d'ancrage pour le segment suivant. Dans les approches basées sur les régions, l'utilisateur peut guider directement ou indirectement le niveau de détail requis sur les différentes régions de l'image [78], [108], [109], [44]. Une version plus simple consiste à pointer avec la souris sur les régions homogènes, calculées au préalable par l'algorithme, qui constituent l'objet d'intérêt [10]. L'interaction pendant l'exécution est la plus flexible parce que l'utilisateur a une maîtrise constante des résultats, mais elle requiert des algorithmes spécialement conçus pour l'interactivité.

Après exécution : ici des outils sont mis à la disposition de l'utilisateur, qui peut modifier manuellement les résultats, ou lancer des algorithmes semi-automatiques de correction [15]. Ici, l'utilisateur n'a pas autant de maîtrise sur l'évolution du résultat qu'avec l'interaction

pendant exécution.

2.4 Les prototypes d'extraction d'objets vidéo

L'une des principales motivations pour la segmentation interactive dans les études récentes est l'apparition des normes MPEG-4 [65], [64] et MPEG-7 [67], [66]. Dans ce contexte, une série de prototypes ont été développés qui ont pour but l'extraction, interactive ou automatique, d'objets vidéo à partir de séquences dont le contenu peut être très varié. Ce manque d'informations sur le contenu des images fait que les approches interactives sont particulièrement bien adaptées.

Nous décrivons ici quelques-unes des approches interactives existantes, car elles sont en rapport très étroit avec le contenu de cette thèse. Des exemples d'approches automatiques sont décrits dans [49] et [102].

Dans la plupart de systèmes interactifs, l'interaction a lieu sur la première image de la séquence, afin de définir l'objet d'intérêt. Ensuite, cet objet est automatiquement suivi tout au long de la séquence. En général, les algorithmes permettent également à l'utilisateur d'interrompre l'exécution pour faire des modifications quand les résultats obtenus ne sont pas satisfaisants.

Dans [11], un vecteur de caractéristiques est utilisé pour décrire chaque pixel de l'image. L'utilisateur doit introduire des points d'apprentissage sur la première image de la séquence permettant d'estimer la fonction de densité de probabilité de chacune des régions constituant l'objet. Les pixels sont alors attribués à la région pour laquelle la probabilité d'appartenance au modèle est la plus grande. Les régions ainsi obtenues ne sont pas nécessairement connexes. Bien que cette méthode ait été une des premières proposées pour la segmentation interactive, l'interaction requise de la part de l'utilisateur n'est pas très intuitive, car l'utilisateur doit être capable de choisir les points des objets les plus représentatifs, et la méthode ne prévoit pas de corrections si les résultats ne sont pas satisfaisants. Un système similaire mais avec de meilleures possibilités d'interaction et des considérations de connexité des régions a été proposé dans [72].

Dans [10], un vecteur de caractéristiques est également utilisé pour décrire les pixels de l'image. Un algorithme de type « fuzzy c-means » avec des contraintes de connexité classe ensuite les pixels en régions. Pour le traitement d'une séquence, l'algorithme est exécuté image par image, l'image au temps $t-1$ servant d'initialisation à l'algorithme pour l'image au temps t. L'interaction avec l'utilisateur n'a lieu qu'à la fin, quand l'utilisateur choisit, parmi les régions trouvées, celles constituant l'objet.

L'approche présentée dans [25] demande à l'utilisateur de dessiner un contour approximatif de l'objet sur la première image de la séquence. Les versions érodées des régions à l'intérieur et à l'extérieur de ce contour sont alors utilisées comme marqueurs d'une LPE afin de trouver les contours exacts de l'objet sur la première image de la séquence. Ensuite, l'objet est suivi automatiquement en utilisant une approche de compensation des contours en mouvement suivie d'une ligne de partage des eaux en prenant comme marqueurs – de la même manière que sur la première image – des versions érodées des régions à l'intérieur et à l'extérieur du contour compensé. L'utilisation des régions intérieur et extérieur à l'objet comme marqueurs pour une LPE a également été proposée pour la segmentation automatique de scènes routières [107], [106]. La technique proposée dans [111] correspond au même type d'interaction. Le contour approximatif introduit par l'utilisateur est alors utilisé comme initialisation d'un contour actif

afin de trouver le contour exact de l'objet. L'intérieur de ce contour est ensuite resegmenté en régions homogènes moyennant une technique de croissance de régions. Dans l'étape de suivi, ces régions sont compensées en mouvement et utilisées comme marqueurs pour une nouvelle croissance de régions sur l'image suivante. La méthode dans [26] est similaire, mais l'utilisateur a la possibilité d'indiquer des caractéristiques sémantiques de l'objet telles que la rigidité. La mesure de distance utilisée pour la croissance de régions est calculée par analyse discriminante sur l'objet défini par l'utilisateur sur la première image. Si des corrections doivent être faites au long du processus de suivi, la mesure de distance est recalculée pour inclure les nouvelles informations.

2.5 Discussion

Beaucoup des méthodes interactives ont été développées à partir d'algorithmes automatiques sur lesquels une étape d'interaction a été ajoutée, comme l'introduction de paramètres d'initialisation, le marquage des objets ou la correction des résultats. Ces mécanismes d'interaction présentent souvent des problèmes :

- besoin de connaître le fonctionnement de l'algorithme de segmentation sous-jacent (en particulier pour l'introduction de paramètres) ;
- imprévisibilité des résultats ;
- manque de flexibilité, qui oblige à refaire les calculs si les résultats ne sont pas totalement satisfaisants.

Pour le cas des séquences vidéo, nous avons vu que l'approche par suivi est la plus fréquente. Un inconvénient d'ordre pratique mais très important lié à cette méthode est le fait que l'utilisateur est obligé de superviser l'algorithme pendant l'exécution (qui peut être très longue). Ceci est nécessaire afin de pouvoir l'arrêter en cas d'erreur pour faire des corrections. Autrement, les possibles erreurs commises par l'algorithme se propagent et il est alors nécessaire de refaire tous les calculs à partir de la première image ayant eu besoin de correction.

Les algorithmes et mécanismes d'interaction que nous présentons dans cette thèse cherchent à apporter une solution à ces problèmes.

Chapitre 3

Les hiérarchies de partitions emboîtées

3.1 Introduction

Ce chapitre présente les bases mathématiques en relation avec les hiérarchies de parties d'un ensemble. Les hiérarchies de partitions emboîtées d'une image – sur lesquelles notre système de segmentation interactive est basé – en sont un cas particulier. Nous proposerons plus tard, dans les chapitres 6 et 7, des méthodes spécifiques permettant de construire des hiérarchies adaptées au contenu des images et des séquences vidéo.

Nous considérons que nous disposons d'un ensemble I dont on veut hiérarchiser les éléments $i \in I$. Nous commençons par introduire, dans la section 3.2, la notion d'arbre. Ensuite, la section 3.3 présente la notion de hiérarchie de parties d'un ensemble. La section 3.4 montre l'équivalence entre hiérarchie et distance ultramétrique, ainsi que plusieurs façons de transformer une mesure de dissimilarité en distance ultramétrique. Nous présentons également, dans la section 3.5, deux manières de représenter une hiérarchie à l'aide de graphes : le dendrogramme et l'arbre aux arêtes valuées. Dans la section 3.6 nous concrétisons les notions présentées pour le cas des hiérarchies de partitions emboîtées d'une image. Finalement, nous décrivons dans la section 3.7 quelques-unes des techniques de construction de hiérarchies utilisées en segmentation.

3.2 La structure d'arbre

Nous reprenons ici le formalisme utilisé par Benzécri [3] pour définir la structure d'arbre et les hiérarchies. Nous noterons la conjonction logique par l'opérateur $\wedge$ et l'union logique par l'opérateur $\vee$.

On notera une relation d'ordre entre deux éléments avec le symbole $\preceq$ placé entre l'élément inférieur et l'élément supérieur. On lira donc $a \preceq b$ comme *b précède a* ou *a est sous b*. La relation $\preceq$ n'exclut pas l'égalité. Ainsi, nous noterons par $\prec$ une relation binaire excluant l'égalité, et nous lirons $a \prec b$ comme *a est strictement sous b*.

Définition 3.1 *Soit A un ensemble. Une relation binaire $\preceq$ sur A est une* **relation de préordre** *si elle est transitive et réflexive :*

$$\forall a, b, c \in A \qquad a \preceq b \ \wedge \ b \preceq c \ \Rightarrow \ a \preceq c \tag{3.1}$$

$$\forall a \in A \qquad a \preceq a \tag{3.2}$$

Définition 3.2 *Soit A un ensemble. Une relation binaire $\preceq$ sur A est une* **relation d'ordre** *si elle satisfait :*

$$\forall a \in A \qquad a \preceq a \tag{3.3}$$

$$\forall a, b, c \in A \qquad a \preceq b \ \wedge \ b \preceq c \ \Rightarrow \ a \preceq c \tag{3.4}$$

$$\forall a, b \in A \qquad a \preceq b \ \wedge \ b \preceq a \ \Leftrightarrow \ a = b \tag{3.5}$$

Si en plus elle satisfait

$$\forall a, b \in A \qquad a \preceq b \ \vee \ b \preceq a \tag{3.6}$$

on dit que l'ordre est **total**.

C'est-à-dire, dans un ensemble muni d'un ordre total, toute paire d'éléments de l'ensemble peut être comparée.

Définition 3.3 *Un ensemble A muni d'une relation d'ordre $\preceq$ a une structure d'*__arbre__ *si*

$$\forall\, a, u, v \in A \qquad a \preceq u \ \wedge \ a \preceq v \ \Rightarrow \ u \preceq v \ \vee \ v \preceq u \tag{3.7}$$

C'est-à-dire qu'un ensemble muni d'une relation d'ordre est un arbre si pour chacun de ses éléments, l'ensemble de ses prédécesseurs est totalement ordonné.

Lemme 3.1 *Sur un arbre, chaque élément (à l'exception du sommet, défini dans le paragraphe suivant) admet un prédécesseur immédiat unique.*

Démonstration En effet, considérons deux prédécesseurs u et v d'un élément $a \in A$. La définition 3.3 implique que soit $u \preceq v$, soit $v \preceq u$, et dans ce cas l'un précède forcément l'autre. Dans le cas où les deux relations seraient vérifiées simultanément, on aurait (axiome 3.5 de la relation d'ordre) $u = v$ et donc le prédécesseur est forcément unique.

Définition 3.4 *Étant donné un ensemble A avec structure d'arbre pour une relation d'ordre $\preceq$, on peut définir les* **sommets** *de l'arbre comme l'ensemble des éléments maximaux de A :*
$Som(A) = \{a \,|\, (a \in A) \ \wedge \ (\forall b \in A : \ a \preceq b \ \Rightarrow \ a = b)\}$

Définition 3.5 *L'ensemble des* **éléments minimaux**, *ou terminaux, de A se défini comme :*
$Ter(A) = \{a \,|\, (a \in A) \ \wedge \ (\forall b \in A : \ b \preceq a \ \Rightarrow \ a = b)\}$

Définition 3.6 *Un arbre avec un sommet unique est appelé* **connexe**.

3.3 La hiérarchie de parties d'un ensemble

Définition 3.7 *Un ensemble $A \subset P(I)$ de parties non vides de I, I étant un ensemble fini, est une* **hiérarchie de parties d'un ensemble** *s'il satisfait :*

Axiome d'intersection : *deux éléments de A non comparables par relation d'inclusion ont une intersection vide.*

$$\forall\, a, b \in A, \qquad a \cap b \in \{a, b, \varnothing\} \tag{3.8}$$

Axiome de réunion : *tout élément de A qui n'est pas minimal est la réunion des éléments distincts de lui et inclus dans lui.*

$$\forall\, a \in A, \qquad \cup \{b \,|\, (b \in A) \ \wedge \ (b \neq a) \ \wedge \ (b \subset a)\} \in \{a, \varnothing\} \tag{3.9}$$

Définition 3.8 *On appelle* **support** *d'une hiérarchie A l'ensemble formé par la réunion de ses parties :*

$$support(A) = \cup\{a \mid a \in A\}$$

Si le support de A est I tout entier, on dit que A est une **hiérarchie de parties sur** I.

Une hiérarchie de parties est dite **fine** si chacun de ses éléments minimaux est un ensemble à un élément. Pour une hiérarchie fine on a donc que $\{i\} \in A$. Remarquons que si $\{i\} \in A$, alors l'axiome de réunion (axiome 3.9) devient superflu, car tout élément de A peut être obtenu par union des éléments isolés $\{i\}$ inclus dans lui.

Dans ce qui suit nous nous intéresserons aux hiérarchies fines de sommet I (on a donc $I \in A$). Une hiérarchie fine de sommet I est une hiérarchie **totale**. Pour une telle hiérarchie, on a $I \in A$ et $\forall i \in I : \{i\} \in A$.

Lemme 3.2 *Une hiérarchie de parties d'un ensemble A muni de la relation d'ordre d'inclusion $\subseteq$ est un arbre.*

Démonstration En effet, si l'on considère deux prédécesseurs u et v quelconques d'un élément $a \in A$, ils ont une intersection non vide puisqu'ils contiennent a ($a \subseteq u$, $a \subseteq v$). Alors, par l'axiome 3.8 on a $u \cap v \in \{u, v\}$ (leur intersection est égale au plus petit des deux) et ils sont alors comparables par inclusion, c'est-à-dire, on a $u \subseteq v$ ou $v \subseteq u$, ce qui correspond à la définition 3.3.

Définition 3.9 *Une relation binaire $\trianglelefteq$ est une* **stratification** *sur l'arbre A, munie de l'ordre hiérarchique $\preceq$, si c'est un préordre total satisfaisant :*

$$\forall a, b \in A \qquad a \prec b \Rightarrow a \lhd b \tag{3.10}$$

C'est-à-dire, si a est strictement en dessous de b pour l'ordre hiérarchique, a est aussi strictement en dessous de b pour l'ordre de stratification. L'ensemble A muni des relations $\preceq$ et $\trianglelefteq$ est appelé un **arbre stratifié**.

L'ordre de stratification peut être défini par une application de A dans $\mathbb{R}$, appelée **indice de stratification**, telle que

$$\forall a, b \in A \qquad a \prec b \Rightarrow f(a) < f(b) \tag{3.11}$$

L'indice de stratification associe une valeur réelle croissante à chaque niveau de la hiérarchie.

Quand A est une hiérarchie totale de parties de I, on peut imposer des conditions supplémentaires à la fonction f :

$$f(I) \leq 1 \qquad \text{et} \qquad f(\{\{i\} \mid i \in I) = 0 \tag{3.12}$$

Dans ce cas, on dit que f est un **indice de diamètre**. Un indice de diamètre prend donc des valeurs croissantes entre 0 et 1, la valeur 0 correspondant aux éléments isolés.

3.4 Hiérarchies et distance ultramétrique

Définition 3.10 *Une fonction d sur un ensemble I est une* **distance ultramétrique** *si elle satisfait aux axiomes :*

$$\forall i, i' \in I \qquad i \neq i' \Rightarrow d(i,i') > 0 \tag{3.13}$$
$$\forall i \in I \qquad d(i,i) = 0 \tag{3.14}$$
$$\forall i, i' \in I \qquad d(i,i') = d(i',i) \tag{3.15}$$
$$\forall i, i', i'' \in I \qquad d(i,i'') \leq sup(d(i,i'), d(i',i'')) \tag{3.16}$$

Remarquons que l'axiome 3.16 est une condition plus forte que l'inégalité triangulaire.

À partir de la distance ultramétrique, nous pouvons définir les boules B_o (boule ouverte) et B_f (boule fermée) de centre c et rayon ρ :

$$B_o(c,\rho) = \{i \,|\, (i \in I) \wedge (d(c,i) < \rho)\} \tag{3.17}$$
$$B_f(c,\rho) = \{i \,|\, (i \in I) \wedge (d(c,i) \leq \rho)\} \tag{3.18}$$

3.4.1 Propriétés

Lemme 3.3 *Le diamètre d'une boule fermée (resp. ouverte) est inférieur ou égal à son rayon.*

Démonstration En effet, considérons deux points i, i' quelconques appartenant à la boule fermée (resp. ouverte) de rayon ρ et centre c. Alors, selon l'axiome 3.16, et sachant que les deux points sont à l'intérieur de la boule, on a :

$$d(i,i') \leq sup\left(d(i,c), d(c,i')\right) \leq \rho$$

ce qui signifie que deux points quelconques appartenant à la boule peuvent être, au plus, éloignés entre eux d'une distance égale au rayon. Ainsi, les deux points les plus éloignés (définition de diamètre) sont au maximum à distance égale au rayon.

Lemme 3.4 *Tout point d'une boule (ouverte ou fermée) est centre de la boule.*

Démonstration Soit $B(c,\rho)$ une boule fermée et i un point quelconque de cette boule. Montrons que i est également centre de $B(c,\rho)$. En vertu du lemme 3.3 on sait que tous les points appartenant à $B(c,\rho)$ sont à distance inférieur ou égale à ρ de i. Ainsi, on a $B(c,\rho) \subseteq B(i,\rho)$.

Montrons à présent l'inclusion inverse $B(i,\rho) \subseteq B(c,\rho)$. Soit $i' \in B(i,\rho)$ et montrons que $i' \in B(c,\rho)$. À partir de l'inégalité ultramétrique, on a :

$$d(i',c) \leq sup\{d(i',i), d(i,c)\} \leq \rho \quad \Leftrightarrow \quad i' \in B(c,\rho)$$

Lemme 3.5 *Deux boules sont soit disjointes soit incluses l'une dans l'autre.*

Démonstration Si les deux boules ont une intersection non vide, forcément tous les points de l'intersection sont centres des deux boules, et en conséquence elles sont concentriques.

3.4.2 Ultramétrique et chaînes de points

Soient un espace muni d'une distance ultramétrique d et une chaîne de points $i^0, i^1, \ldots, i^n$. La relation suivante est vérifiée :

$$d(i^0,\, i^n) \;\leq\; sup\,\{\, d(i^{h-1},\, i^h) \,|\, h \,=\, 1, \ldots,\, n \,\} \tag{3.19}$$

C'est-à-dire que la distance entre deux extrémités d'une chaîne est au maximum égale au plus long des maillons de la chaîne. Cette inégalité est plus forte que l'inégalité triangulaire, et se démontre par récurrence sur n ($n = 2$ correspond à l'axiome 3.16).

3.4.3 Ultramétrique et hiérarchie

Considérons maintenant une hiérarchie totale stratifiée A de parties d'un ensemble I munie d'un indice de diamètre f. On peut définir la distance d entre deux éléments i, i' de I comme le plus petit indice f de la hiérarchie pour lequel les deux éléments i et i' appartiennent au même élément de A, $a(i, i')$. Nous appellerons les éléments $a \in A$ *classes* de la hiérarchie. La classe la plus petite contenant i et i' est

$$a(i,\, i') \;=\; inf\{a \,|\, (a \in A) \;\wedge\; (i \in a) \;\wedge\; (i' \in a)\} \tag{3.20}$$

et la distance correspondante est l'indice de stratification de cet élément dans la hiérarchie :

$$d(i, i') \;=\; f(a(i, i')) \tag{3.21}$$

L'existence de l'élément $a(i, i')$ est garantie si $I \in A$ (l'ensemble complet est réuni dans une seule classe correspondant au sommet de l'arbre). L'élément $a(i, i')$ est obtenu par intersection de toutes les classes contenant i et i'. Si l'élément $a(i, i')$ n'existe pas ($i \notin A$ et i, i' appartiennent à des sommets différents), alors la distance entre i et i' est définie comme étant infinie.

La distance ainsi définie est une distance ultramétrique. Pour le vérifier, il suffit de vérifier que

$$\forall\, i, i', i'' \in A \qquad d(i', i'') \;\leq\; sup\,\{\, d(i', i),\, d(i, i'') \,\} \tag{3.22}$$

En effet, $a(i, i')$ et $a(i, i'')$ contiennent tous les deux l'élément i, ce qui implique que l'un des deux ensembles contient l'autre (axiome 3.8 de la hiérarchie) et en conséquence contient les trois éléments i, i' et i''. La relation ultramétrique est alors vérifiée, et peut être énoncée : *l'indice de la hiérarchie pour lequel deux éléments i' et i'' sont réunis pour la première fois est plus petit ou égal à l'indice de la hiérarchie pour lequel trois éléments i, i' et i'' sont réunis pour la première fois.*

Réciproquement, étant donné un ensemble fini I muni d'une distance ultramétrique d, on peut définir une hiérarchie totale indicée de parties A en prenant comme éléments de la hiérarchie les boules de la distance ultramétrique et en associant à chaque élément le rayon de la boule la plus petite qui le contient. En effet, l'ensemble des boules d'une distance ultramétrique vérifie les axiomes 3.8 et 3.9. L'axiome 3.8 est satisfait puisque deux boules de l'ultramétrique sont soit disjointes soit incluses l'une dans l'autre, et en conséquence leur intersection donne la plus petite des deux ou l'ensemble vide. L'axiome 3.9 est aussi satisfait : si l'on prend une boule quelconque de rayon r, B_r, et on considère l'ensemble des boules ayant pour centre l'un

des points de cette boule et un rayon strictement inférieur à r, on sait, par une des propriétés de l'ultramétrique vues précédemment, que ces boules sont totalement incluses dans la boule de départ. En conséquence, et parce qu'il y a au moins une boule centrée sur chaque point de l'espace (la boule triviale formée par un seul point avec $d = 0$), l'union de ces boules donne la boule de départ B_r.

Il est donc équivalent de se définir sur un ensemble I une hiérarchie fine stratifiée de parties ou une distance ultramétrique. Pour définir l'ultramétrique à partir de la hiérarchie il suffit de prendre comme distance ultramétrique entre deux éléments l'indice de stratification de la classe la plus petite contenant les deux éléments, et pour définir la hiérarchie stratifiée à partir de la distance ultramétrique il suffit de prendre les boules de cette distance comme classes de la hiérarchie et leur rayon comme indice de stratification. La hiérarchie associée à d est forcément fine, car l'axiome 3.13 de l'ultramétrique ainsi l'impose ($i \neq i' \Rightarrow d(i,i') > 0$, ce qui implique que les boules de rayon nul contiennent les éléments isolés). En assouplissant cet axiome pour inclure l'égalité ($i \neq i' \Rightarrow d(i,i') \geq 0$), on inclut la possibilité d'avoir des boules de rayon nul (niveau plus fin de la hiérarchie) contenant plus d'un élément de I, ce qui permet de travailler avec des hiérarchies A qui ne sont pas fines.

3.4.4 Algèbre d'ultramétriques

Différentes ultramétriques d_k peuvent être définies sur un même ensemble de données I. On peut alors s'interroger sur la façon de les combiner pour obtenir de nouvelles ultramétriques, qui donneront lieu à de nouvelles hiérarchies.

Sup d'ultramétriques Il est facile de vérifier que la distance définie comme

$$d(i,i') = \sup_k \{d_k(i,i')\} \qquad \forall\, i,i' \in I \tag{3.23}$$

est une distance ultramétrique, ce qui permet de combiner plusieurs ultramétriques.

Interprétation : Le sup d'ultramétriques considère deux éléments comme différents si *une* des distances de départ les a vus comme étant différents. Pareillement, deux éléments seront considérés comme étant proches seulement si toutes les distances sont d'accord.

Inf d'ultramétriques Malheureusement, la métrique

$$d(i,i') = \inf_k \{d_k(i,i')\} \qquad \forall\, i,i' \in I \tag{3.24}$$

n'est pas une ultramétrique et ses boules ne définissent donc pas une hiérarchie.

À titre d'exemple, considérons un ensemble A formé par trois points a, b, c, sur lequel deux distances ultramétriques d_1 et d_2 ont été définies :

	d_1	d_2	d_{inf}
a, b	1	2	1
a, c	3	2	2
b, c	3	1	1

Bien que les distances d_1 et d_2 vérifient la condition ultramétrique, ce n'est pas le cas pour d_{inf}, puisque $d_{inf}(a,c) \not\leq sup\{d_{inf}(a,b), d_{inf}(b,c)\}$.

Cependant, nous verrons dans la section 3.4.5 comment construire des ultramétriques à partir d'un indice de distance. Ainsi, même si la métrique définie par inf de distances

ultramétriques n'est pas une ultramétrique en elle-même, elle peut être transformée en une ultramétrique en passant par la construction d'une hiérarchie stratifiée.

On pourra considérer par exemple l'ultramétrique inférieure maxima – appelée également ultramétrique sous-dominante –, c'est-à-dire la plus grande ultramétrique plus petite que toutes les d_k. En effet, la famille de toutes les ultramétriques inférieures aux d_k n'est pas vide, puisqu'elle contient au moins l'ultramétrique nulle. Le sup d'ultramétriques est une ultramétrique, ce qui permet de choisir, parmi toutes les ultramétriques inférieures aux d_k, la plus grande.

Lexicographies d'ultramétriques Un ordre lexicographique est établi dans un ensemble d'éléments en associant à chaque élément d^m un vecteur $(d_1^m, d_2^m, \ldots, d_n^m)$. La relation d'ordre se définit de la manière suivante :

$$d^j \leq d^k \quad \Leftrightarrow \quad \begin{cases} \left(d_1^j < d_1^k\right) & \text{ou} \\ \left(d_1^j = d_1^k \quad \text{et} \quad d_2^j \leq d_2^k\right) & \text{ou} \\ \vdots & \\ \left(d_1^j = d_1^k \quad \text{et} \quad d_2^j = d_2^k \quad \ldots \quad \text{et} \quad d_n^j \leq d_n^k\right) & \end{cases}$$

Bien entendu, pour pouvoir définir un ordre lexicographique il est nécessaire qu'une relation d'ordre ait été définie entre les d_k. Notre cas ne pose pas de problème, puisque les d_k sont des distances ultramétriques, et donc des nombres réels. L'ordre lexicographique correspond à l'ordre du dictionnaire : pour comparer deux éléments on regarde le premier chiffre, s'il y a égalité on regarde le deuxième, etc.

La lexicographie d'ultramétriques n'est pas une ultramétrique, comme le montre l'exemple suivant :

	d_1	d_2	d_{lex}
a, b	3	3	(3, 3)
a, c	3	2	(3, 2)
b, c	2	3	(2, 3)

où l'on voit que $d_{lex}(a,b) \not\leq sup\{d_{lex}(a,c), d_{lex}(c,b)\}$. De la même manière que pour le cas de l'inf d'ultramétriques, et comme nous le montrerons dans la section 3.4.5 il est possible de considérer la lexicographie obtenue comme un indice de distance sur lequel une hiérarchie, et donc une ultramétrique, peut être construite.

3.4.5 Construction d'ultramétriques à partir d'un indice de distance

Souvent, étant donné un ensemble I dont on veut hiérarchiser les éléments, on ne dispose pas d'une distance ultramétrique, mais on peut estimer un indice de distance (ou dissimilarité) s entre les éléments.

Définition 3.11 *Étant donné un ensemble I, un indice de distance s entre les éléments de I est une fonction sur $I \times I$ satisfaisant :*

$$\forall i, i' \in I \qquad i \neq i' \Rightarrow s(i, i') > 0 \tag{3.25}$$

$$\forall i \in I \qquad s(i, i) = 0 \tag{3.26}$$

$$\forall i, i' \in I \qquad s(i, i') = s(i', i) \tag{3.27}$$

On peut alors s'interroger sur la façon de construire des distances ultramétriques d à partir de l'indice de distance s.

Étant donné un indice de distance s, il est possible de construire une distance ultramétrique d à travers l'équivalence entre ultramétrique et hiérarchie stratifiée. **La distance ultramétrique associée sera l'indice de stratification minimum pour lequel deux éléments sont regroupés dans la même classe** : $d(i,i') = f(a(i,i'))$, où $f(a(i,i'))$ est l'indice de stratification de la classe $a(i,i')$ dans la hiérarchie, et $a(i,i')$ est la plus petite classe contenant i et i'.

Il existe deux techniques principales permettant de construire des hiérarchies à partir d'un indice de distance s : la méthode ascendante (« bottom-up » en anglais) et la méthode descendante (« top-down »). La méthode ascendante produit la hiérarchie en regroupant successivement des éléments dans la même classe selon l'indice de distance s. La méthode descendante adopte l'approche contraire, où la hiérarchie s'obtient par des divisions successives des classes jusqu'à obtenir en dernière phase les éléments isolés.

Mais pour construire des hiérarchies il faut être capable de calculer la dissimilarité entre deux classes $a, a' \in A$ à partir de l'indice de dissimilarité $s(i,i')$ entre les éléments isolés $i, i' \in I$. Ainsi, l'indice de dissimilarité entre classes, que nous appelons $r(a,a')$ est une fonction de l'indice de dissimilarité entre éléments $s(i,i')$. Différentes définitions de cette fonction amèneront à des hiérarchies différentes et donc à des ultramétriques différentes. Quand la fonction $r(a,a')$ est croissante ($a,a' \subset b \Rightarrow r(a,a') \leq r(b,b')$) alors cette fonction peut être utilisée comme indice de stratification f donnant la distance ultramétrique (rappelons que l'indice de stratification doit être une fonction croissante avec les classes de la hiérarchie). Quand $r(a,a')$ n'est pas une fonction croissante, un indice de stratification doit être défini. Dans le cas de la construction de bas en haut, il suffit de prendre l'ordre de fusion comme indice de stratification, ou n'importe quelle fonction croissante de cet ordre, tandis que pour la construction de haut en bas on prendra une fonction décroissante de l'ordre de division.

Étant donnés un indice de distance entre les éléments isolés $s(i,i')$ et un indice de distance entre classes $r(a,a')$, on peut se demander si la hiérarchie obtenue par construction de haut en bas est la même que celle obtenue par construction de bas en haut. La réponse à cette question est non, comme nous verrons par la suite.

3.4.5.1 L'approche ascendante

L'approche ascendante est décrite par l'algorithme 3.1. Un indice de distance entre classes est défini, et à chaque itération les deux classes les plus proches sont regroupées.

Nous donnons ensuite quelques exemples de fonctions utilisées pour définir la dissimilarité entre classes, $r(a,a')$, à partir de l'indice de dissimilarité entre les éléments de I.

Ultramétrique sous-dominante Une possibilité pour définir l'indice de distance entre classes à partir de l'indice de distance entre éléments isolés consiste à prendre comme indice de distance entre classes l'indice de distance entre leurs éléments les plus proches. C'est-à-dire :

$$r(a,a') = inf\{s(i,i') \mid (i \in a) \wedge (i' \in a')\} \qquad a,a' \in A \quad i,i' \in I$$

Pour cette définition, la distance entre classes est une fonction croissante :

$$\forall\, a,b,c \in A \qquad (a \subseteq c) \wedge (b \nsubseteq c) \Rightarrow r(a,c) < r(b,c)$$

Algorithme 3.1 Algorithme ascendant de construction d'une hiérarchie

$n \leftarrow$ Nombre initial d'éléments
$TableDissimilarités \leftarrow$ Dissimilarité entre toute paire d'éléments
while $n > 1$ **do**
  $a_i, a_j \leftarrow$ TrouverMinimum($TableDissimilarités$)
  Regrouper(a_i, a_j) {Regrouper les deux classes les plus proches}
  Actualiser($TableDissimilarités$) {Actualiser dissimilarités entre les classes}
  $n \leftarrow n - 1$
end while

Ainsi, on peut prendre comme indice de stratification d'une classe b de la hiérarchie l'indice de distance pour lequel cette classe est constituée :

$$\forall\, a, a', b \in A \qquad f(b) = sup\{r(a, a') \mid a, a' \in b\}$$

et comme distance ultramétrique le plus petit indice de stratification pour lequel deux éléments sont regroupés dans la même classe :

$$d(i, i') = f(b(i, i')) \quad \text{avec} \quad b(i, i') = inf\{b \,|\, (b \in A) \wedge (i \in b) \wedge (i' \in b)\} \tag{3.28}$$

L'ultramétrique ainsi définie est l'ultramétrique inférieure maxima (la démonstration peut être trouvée dans [3]). Par inférieure, on signifie une distance en tout point inférieure à s, c'est-à-dire :

$$d(i, i') \leq s(i, i') \quad \forall i, i' \in I \tag{3.29}$$

Par maxima, on signifie qu'elle est plus grande que toutes les ultramétriques inférieures à l'indice s. Cette ultramétrique est celle qu'on obtiendrait en considérant la famille d_k des ultramétriques inférieures ou égales à s, et en calculant le sup. En effet, comme nous avons vu dans la section 3.4.4, le sup d'ultramétriques est encore une ultramétrique.

L'ultramétrique inférieure maxima correspond à la construction de la hiérarchie par sauts minimaux. À chaque itération, les deux classes a et a' contenant les éléments i et i' les plus proches sont fusionnées. Ainsi, deux classes sont fusionnées si elles ont deux éléments proches l'un de l'autre, indépendamment des autres éléments de la classe. Ce type de hiérarchie correspond aux méthodes de classification par chemins minimaux ou « single linkage » (chaînage simple) [32], [33], [22]. Dans ce type de hiérarchies, les classes peuvent se rassembler par des voisinages étroits, ce qui peut donner lieu à des classes aux formes allongées.

Ultramétrique supérieure Au lieu de définir l'indice de distance entre classes comme l'indice de distance entre leurs éléments les plus proches, on peut le définir comme l'indice de distance entre leurs éléments les plus éloignés :

$$r(a, a') = sup\{s(i, i') \mid (i \in a) \wedge (i' \in a')\} \qquad a, a' \in A \quad i, i' \in I \tag{3.30}$$

La hiérarchie est alors construite par fusion des classes a et a' dont les deux éléments $i \in a$ et $i' \in a'$ les plus éloignés sont plus proches. L'indice de stratification plus petit pour lequel deux

éléments sont regroupés dans la même classe donne la distance ultramétrique, tel qu'exprimé par la relation 3.28. La fonction $r(a, a')$ étant croissante, on peut prendre la valeur de cette fonction comme indice de stratification, ou n'importe quelle fonction croissante de l'ordre de fusion.

L'indice r que nous venons de définir donne une ultramétrique en tout point supérieure à s, mais remarquons qu'elle n'est pas l'ultramétrique supérieure minima. En effet, si l'on considère l'ensemble des ultramétriques d_k en tout point supérieures à s, pour trouver la plus petite il faut calculer l'inf. Or l'inf d'ultramétriques n'est pas, comme nous l'avons vu dans la section 3.4.4, une ultramétrique. Ainsi, l'inf de la famille peut ne pas exister.

Ce type de hiérarchie correspond aux méthodes de classification connues comme « complete linkage » (chaînage complet) [33], [32] et donne lieu à des classes aux formes arrondies, puisque deux classes ne sont rassemblées que si tous leurs éléments sont proches.

Ultramétrique obtenue par distances moyennes Ici, l'indice de distance entre classes est défini comme une moyenne des dissimilarités entre les éléments de la classe :

$$r(a, a') = \frac{\sum\{s(i, i') \mid (i \in a) \wedge (i' \in a')\}}{|a| \times |a'|} \tag{3.31}$$

où $|a|$ est le nombre d'éléments de l'ensemble a.

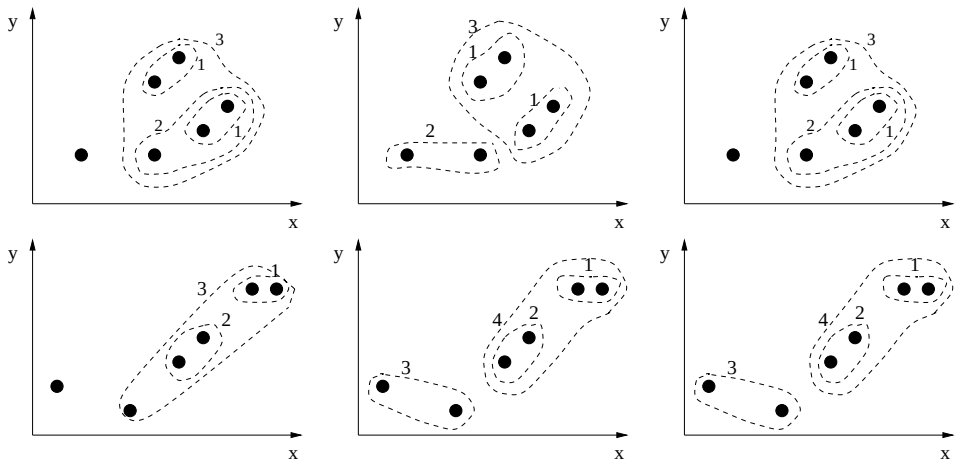
Il est possible que la mesure de dissimilarité entre classes qui résulte ne soit pas croissante ($r(a, a') \not\leq r(b, b')$ quand $a, a' \subset b$), mais dans ce cas on peut encore définir un indice de stratification croissant, donné par l'ordre de fusion des classes. L'ultramétrique associée est l'indice de stratification plus petit pour lequel deux éléments se trouvent dans la même classe.

Quand la distance entre classes $r(a, a')$ est une fonction croissante, l'ultramétrique résultante a la propriété d'être la plus proche de la métrique s au sens des moindres carrés [28].

La figure 3.1 présente deux exemples des ultramétriques obtenues pour les constructions que nous venons de présenter. Les boules des ultramétriques sont montrées ainsi que leur rayon. Tandis que le chaînage simple rejoint les points par le chemin le plus court, le chaînage complet fusionne deux ensembles quand tous leurs éléments sont proches. Enfin, le chaînage moyen (ultramétrique obtenue par distances moyennes) est une solution intermédiaire. Dans le premier exemple, le résultat coïncide avec celui donné par le chaînage simple, tandis que pour le deuxième exemple l'ultramétrique obtenue coïncide avec celle obtenue avec le chaînage complet. Ces coïncidences sont dues à la simplicité des ensembles considérés, mais en général les résultats obtenus par chaînage moyen seront différents de ceux obtenus pour les autres constructions.

3.4.5.2 L'approche descendante

L'approche descendante à la construction de hiérarchies est synthétisée par l'algorithme 3.2. À chaque itération l'algorithme doit chercher la meilleure partition entre toutes les partitions possibles. Pour cette raison, cette approche est en général plus coûteuse en temps de calcul que la méthode ascendante, en particulier quand le nombre d'éléments est élevé.



(a) Chaînage simple (b) Chaînage complet (c) Chaînage moyen

FIG. 3.1 – Hiérarchies obtenues à partir d'un indice de distance.

Algorithme 3.2 L'approche descendante de calcul d'une hiérarchie

$N \leftarrow$ Nombre total d'éléments
$n \leftarrow 1$ {Nombre de classes}
while $n < N$ **do**
  $a_i, a_j \leftarrow$ Trouver la division entre classes avec dissimilarité maximale
  Diviser(a_i, a_j)
  $n \leftarrow n + 1$
end while

Ultramétrique obtenue par coupes maximales L'indice de dissimilarité entre classes est défini comme la somme de dissimilarités entre les éléments des classes :

$$r(a,a') \;=\; \sum\{s(i,i') \mid (i \in a) \wedge (i' \in a')\} \tag{3.32}$$

La hiérarchie est construite en calculant les partitions telles que la dissimilarité entre classes est maximale. Cette mesure de dissimilarité entre classes étant une fonction croissante avec les classes de la hiérarchie, on peut prendre comme indice de stratification d'une classe, de la même manière que nous l'avons fait pour les hiérarchies ascendantes, la plus grande dissimilarité à l'intérieur de la classe : $f(b) = sup\{r(a,a') \,|\, a,a' \in b\}$. La distance ultramétrique entre deux éléments, $d(i,i')$ est alors définie comme l'indice de stratification de la plus petite classe contenant i et i'.

Il est également possible de normaliser par le nombre d'éléments, et dans ce cas on obtient l'indice de dissimilarité entre classes utilisé pour la construction ascendante basée sur les distances moyennes, donné par l'équation 3.31. Les ultramétriques obtenues par construction ascendante et descendante à partir d'un même indice de distance entre classes $r(a,a')$ ne sont pas les mêmes. En effet, l'approche de construction de haut en bas cherche à maximiser la dissimilarité entre classes, tandis que l'approche de construction de bas en haut cherche à minimiser la dissimilarité à l'intérieur des classes. Supposant que les deux approches trouvent la solution optimale – ce qui n'est pas nécessairement le cas, car dans certains cas trouver une solution optimale pourrait impliquer remettre en question les niveaux de la hiérarchie déjà calculés – il s'agit de deux problèmes différents dont la solution n'est pas la même. Même dans certains cas particuliers où la maximisation de la dissimilarité entre classes serait équivalente à la minimisation de dissimilarité à l'intérieur des classes, on n'obtiendrait la même solution que si elle est unique et que les deux constructions donnent la solution optimale.

Un problème similaire est celui des coupes minimales. Dans ce cas, la fonction $s(i,i')$ ne représente pas une dissimilarité, mais une similarité, et l'algorithme 3.2 doit trouver la division entre classes avec similarité minimale [4].

3.4.6 Le treillis des hiérarchies

Considérons l'ensemble de toutes les ultramétriques d_j définies sur un ensemble I. Il est possible de définir une relation d'ordre entre les ultramétriques de la manière suivante :

$$d_j \leq d_k \quad \Leftrightarrow \quad d_j(i,i') \leq d_k(i,i') \quad \forall\, i,i' \in I \tag{3.33}$$

Sur cet ensemble, le sup est défini comme :

$$sup\{d_j,d_k\}(i,i') \;=\; sup\{d_j(i,i'),d_k(i,i')\} \quad \forall\, i,i' \in I \tag{3.34}$$

ce qui est une ultramétrique.

L'inf de deux ultramétriques d_j, d_k est donné par la plus grande ultramétrique plus petite que d_j et d_k, c'est-à-dire par l'ultramétrique sous-dominante, qui s'exprime :

$$inf\{d_j,d_k\}(i,i') \;=\; \inf_{C \in C(i,i')} \{\sup_{C}\{inf(d_j(i,i'),d_k(i,i'))\}\} \quad \forall\, i,i' \in I \tag{3.35}$$

où $C(i,i')$ est l'ensemble des chemins C possibles entre les éléments i et i' : $C(i,i') = \{i^0,i^1,\ldots,i^n \,|\, i^0 = i, i^n = i'\}$.

Afin de compacter la notation, nous noterons le sup de deux ultramétriques d_j et d_k comme $d_j \vee d_k$, et l'inf comme $d_j \overline{\wedge} d_k$. Nous utilisons le symbole $\overline{\wedge}$ au lieu de $\wedge$ pour l'inf afin de le distinguer de l'inf élément à élément qui, comme nous l'avons vu, n'est pas une ultramétrique.

Pour $d_j \overline{\wedge} d_k$, deux points i, i' ne sont à distance plus grande que ρ entre eux que s'ils le sont pour d_j et pour d_k sur tous les chemins possibles entre i et i'. De cette manière, deux points qui sont éloignés pour l'une des ultramétriques mais proches pour l'autre seront considérés comme étant proches. On voit bien que les boules obtenues pour $d_j \overline{\wedge} d_k$ sont plus grandes que celles associées à d_j et d_k pour une même valeur du rayon. Les boules de $d_j \overline{\wedge} d_k$ de rayon plus petit ou égal à ρ contiennent tous les points qui sont à distance inférieur ou égale à ρ entre eux *pour l'une des ultramétriques*.

L'ensemble des ultramétriques définies sur un ensemble I, muni de la relation $\leq$, du sup $\vee$ et de l'inf $\overline{\wedge}$, forme donc un **treillis d'ultramétriques**.

Nous avons vu que les notions d'ultramétrique et de hiérarchie stratifiée de parties d'un ensemble sont équivalentes. Ainsi, le treillis des ultramétriques sur un ensemble I équivaut au **treillis des hiérarchies** stratifiées de parties de l'ensemble I. Le treillis des hiérarchies est muni de la relation d'ordre :

$$A_j \leq A_k \quad \Leftrightarrow \quad a_j(i,i') \subseteq a_k(i,i') \quad \forall\, i, i' \in I \tag{3.36}$$

où $a_j \in A_j$, $a_k \in A_k$, et $a(i,i')$ est la plus petite classe de la hiérarchie contenant i et i'. Ainsi, une hiérarchie A_j est plus petite qu'une hiérarchie A_k si elle est plus fine. Pour la hiérarchie A_j, toute paire d'éléments se retrouve dans une même classe pour un indice de stratification plus petit que pour la hiérarchie A_k. C'est-à-dire que la distance ultramétrique entre toute paire d'éléments i, i' – donnée par l'indice de stratification – est plus grande pour la hiérarchie A_j que pour la hiérarchie A_k.

À $d_j \vee d_k$ correspond l'inf des hiérarchies $A_j \wedge A_k$. En effet, $d_j \vee d_k$ prend comme distance entre deux éléments i, i' la plus grande distance de d_j, d_k. Considérons une distance ρ donnée et étudions les boules de $d_j \vee d_k$ de rayon ρ. Pour deux points quelconques i, i' appartenant à une de ces boules, on a soit $d_j(i,i') = \rho$ et $d_k(i,i') \leq \rho$, soit $d_j(i,i') \leq \rho$ et $d_k(i,i') = \rho$. Ainsi, les boules de rayon ρ de $d_j \vee d_k$ sont formées des points qui sont à distance inférieur ou égale à ρ entre eux *à la fois sur d_j et d_k*. Les boules (ou classes de la hiérarchie) associées au sup d'ultramétriques sont donc données par l'intersection – pour chaque valeur de ρ – des boules associées à chacune des distances :

$$inf\{A_j, A_k\} = A_j \wedge A_k = \{a_j \cap a_k \,|\, f(a_j) = f(a_k)\} \tag{3.37}$$

où f(a) est l'indice de stratification f de la classe a dans la hiérarchie, et $a_m \in A_m$.

À $d_j \overline{\wedge} d_k$ correspond le sup de hiérarchies $A_j \vee A_k$. Pour $d_j \overline{\wedge} d_k$, deux points ne sont séparés dans des boules différentes pour une valeur de distance ρ que si d_j et d_k ont une valeur supérieure ou égale à ρ pour tous les chemins possibles. Ainsi, deux points i, i' appartiennent à la même boule de rayon ρ de $d_j \overline{\wedge} d_k$ s'il existe un chemin pour lequel toute paire de points intermédiaires appartiennent à la même boule soit pour d_k, soit pour d_j. Seulement deux points à distance plus grande que ρ à la fois pour d_k et d_j, et cela pour tous les chemins possibles, appartiendront à des boules différentes sur $d_j \overline{\wedge} d_k$. Ainsi, pour chaque niveau ρ des hiérarchies, deux points qui appartiennent à la même classe pour l'une des hiérarchies A_k ou A_j appartiennent aussi à la même classe pour $A_j \vee A_k$. La figure 3.2 montre un exemple. Les points représentent des éléments $i \in I$, et en pointillé nous avons représenté les classes de la hiérarchie pour un niveau ρ, ce qui équivaut à prendre les boules de rayon ρ de l'ultramétrique.

Les classes de $A_j \vee A_k$ sont données par les plus petites classes qui contiennent les classes de d_j et d_k avec lesquelles elles intersectent.

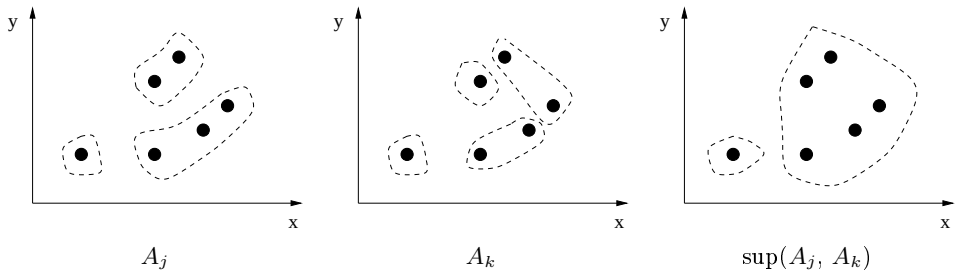


FIG. 3.2 – Exemple de sup de hiérarchies (niveau ρ de la hiérarchie).

Nous verrons l'interprétation du treillis des hiérarchies pour les hiérarchies de partitions d'une image dans le chapitre 6, section 6.5.2.

Le treillis des partitions Remarquons d'abord que les classes de la hiérarchie d'indice de stratification plus petit ou égal à r forment une partition de l'ensemble I si $\cup\{a \,|\, f(a) \leq r\} = I$. Pour une telle hiérarchie, le treillis de hiérarchies est un **treillis de hiérarchies de partitions**.

Considérons maintenant le cas particulier d'une hiérarchie de partitions à un seul niveau. Pour une telle hiérarchie, les éléments i, i' soit appartiennent à la même classe $a(i, i')$, soit appartiennent à des classes différentes. L'ensemble des classes de cette hiérarchie constitue une partition. On peut associer à cette hiérarchie une ultramétrique binaire, pour laquelle $d(i, i') = 0$ s'il existe un $a(i, i')$ qui les contient tous les deux, et $d(i, i') = 1$ autrement. Pour ce cas particulier de hiérarchie à un niveau, le treillis des hiérarchies de partitions correspond au **treillis de partitions** tel qu'il est décrit dans [90].

3.5 Représentation de la hiérarchie

Nous discutons ici de différentes façons de représenter des hiérarchies à l'aide de graphes. À ce propos, nous introduisons d'abord quelques définitions utilisées en théorie des graphes.

3.5.1 Quelques définitions

Définition 3.12 *Un* **graphe** $G = (X, U)$ *est formé par un ensemble de* nœuds X *et un ensemble d'arcs* U*, les arcs étant des couples* $e_{ij} = (v_i, v_j)$*,* $v_i, v_j \in X$*.*

Définition 3.13 *Un* **chemin** *est une séquence d'arcs telle que l'extrémité terminale de chaque arc coïncide avec l'extrémité initiale de l'arc suivant.*

Souvent, nous nous intéressons aux graphes **non orientés**, où la notion d'arc est remplacée par celle d'**arête**, consistant en un couple non-orienté de nœuds : $e_{ij} = e_{ji}$.

Définition 3.14 *Une* **chaîne** *est une séquence d'arcs telle que chacun de ses arcs intermédiaires est rattaché à l'arc précédent par une de ses extrémités et à l'arc suivant par l'autre.*

Un chemin est donc une chaîne, mais le contraire n'est pas vrai.

Définition 3.15 *Un graphe est* **connexe** *si pour tout couple de nœuds distincts, il existe une chaîne allant de l'un à l'autre.*

Définition 3.16 *Dans un graphe, on appelle* **cycle** *une chaîne n'utilisant pas deux fois le même arc et où les nœuds initial et terminal coïncident.*

Définition 3.17 *Étant donné un graphe $G = (X, V)$ et un sous-ensemble de nœuds $A \subset X$, le* **cocycle** *de A est l'ensemble d'arcs avec une extrémité en A et l'autre en $\overline{A} = X \backslash A$.*

Définition 3.18 *Un* **arbre** *est un graphe connexe et sans cycle. Une* **forêt** *est un graphe dont chaque composante connexe est un arbre.*

Cette définition d'arbre correspond à celle utilisée en théorie des graphes, et ne doit pas être confondue avec celle donnée dans la section 3.2, qui correspond à celle utilisée en taxinomie[1]. Nous regardons ensuite le rapport entre ces deux concepts.

Théorème 3.1 *[4] Soit $T = (X, U)$ un graphe avec ordre (nombre de nœuds) $|X| = n \geq 2$. Les propriétés suivantes sont équivalentes pour caractériser un arbre :*

- *T est connexe et sans cycle.*
- *T est sans cycle et admet $n - 1$ arcs.*
- *T est connexe et admet $n - 1$ arcs.*
- *T est sans cycle et en ajoutant un arc, on crée un seul cycle.*
- *T est connexe, et si on supprime un arc quelconque, il n'est plus connexe.*
- *Tout couple de nœuds est relié par une chaîne et une seule.*

Définition 3.19 *On appelle* **centre** *d'un graphe un nœud v_i tel que tout autre nœud du graphe puisse être atteint par un chemin issu de v_i.*

Notons qu'un graphe non-orienté ne peut pas avoir de centre, puisque la notion de chemin ne s'applique qu'aux graphes orientés. Cependant, ont peut également interpréter un graphe non-orienté comme un graphe où chaque arête correspond à un couple d'arcs $e_{ij} = e_{ji}$ orientés dans les deux sens, et pour cette interprétation on peut considérer que tout nœud est centre du graphe.

Ensuite, notons aussi qu'un graphe orienté peut ne pas avoir de centre.

Définition 3.20 *On appelle* **arborescence** *un arbre muni d'un centre.*

3.5.2 Le dendrogramme

Le dendrogramme représente d'une façon graphique très intuitive la structure d'arbre définie dans la section 3.2 (sens taxinomique). Chaque élément A de l'arbre est représenté par un nœud du dendrogramme. Le dendrogramme est généralement représenté verticalement, les nœuds en haut représentant les sommets de l'arbre (l'arbre a un sommet unique s'il est – au sens taxinomique – connexe), les nœuds les plus bas représentant les nœuds terminaux. Les branches du dendrogramme représentent des relations d'inclusion entre les nœuds. Noter que sur un dendrogramme les arcs sont orientés, puisqu'ils représentent des relations d'inclusion. On peut assigner à chacun des noeuds de l'arbre une valeur réelle $f(a)$ telle que $f(a) < f(b)$ si $a \subset b$. Dans ce cas, le dendrogramme représente un arbre stratifié.

Du point de vue de la théorie des graphes, le dendrogramme est une arborescence : il s'agit bien d'un arbre (au sens de la théorie des graphes) puisqu'il est connexe et sans cycle, et il admet un centre, correspondant au sommet du dendrogramme.

[1] Science de la classification

Nous pouvons maintenant établir le rapport entre les définitions d'arbre données par la taxinomie et par la théorie des graphes. Un arbre au sens taxinomique correspond à une arborescence au sens de la théorie des graphes, et donc la définition d'arbre donnée par la théorie des graphes a un sens plus large que celle donnée par la taxinomie.

La figure 3.3 montre un dendrogramme correspondant à un arbre stratifié.

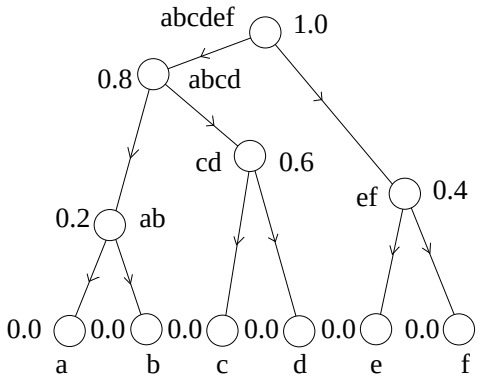


FIG. 3.3 – Dendrogramme représentant un arbre stratifié.

3.5.3 Arbre aux arêtes valuées et arbre de poids minimum

Afin d'éviter des confusions terminologiques, dans la discussion qui suit nous utiliserons le terme *arbre* au sens de la théorie des graphes. Nous ferons référence à la notion d'arbre en taxinomie en utilisant le terme *hiérarchie*, et nous appellerons sa représentation *dendrogramme*.

Le dendrogramme donne une représentation directe et intuitive de la hiérarchie des parties d'un ensemble A. Cependant, il existe d'autres manières de représenter les hiérarchies. En particulier, nous considérons ici celles basées sur les arbres non orientés, où chaque arête e_{ij} est valuée avec une valeur $w_{ij} > 0$.

Sur un arbre aux arêtes valuées, les nœuds représentent les éléments minimaux de la hiérarchie $Ter(A)$. À la différence du dendrogramme, les éléments non minimaux de la hiérarchie ne sont pas représentés par des nœuds, mais par les forêts qui s'obtiennent en supprimant les arêtes de l'arbre *par ordre décroissant de valeur*. Nous avons vu par le théorème 3.1 que la suppression d'une arête dans un arbre produit un graphe non connexe sans cycle, et donc une forêt, contenant dans ce cas deux arbres. De la même manière, la suppression de $k-1$ arêtes ($k \leq n$, où n est le nombre de nœuds de l'arbre) donne une forêt avec k arbres. Chaque arbre correspond à une classe a de la hiérarchie. Les classes de la hiérarchie ainsi définie satisfont aux axiomes 3.8 et 3.9 : les arbres sont soit disjoints soit inclus l'un dans l'autre (à chaque fois qu'on supprime une arête on repartitionne l'un des arbres), et chacun des arbres peut être obtenu par union des arbres plus petits que lui et inclus dans lui.

La figure 3.4 montre la même hiérarchie que la figure 3.3, mais représentée sous forme d'arbre aux arêtes valuées. Les nœuds de l'arbre correspondent aux nœuds terminaux du dendrogramme. Si l'on commence par enlever toutes les arêtes de l'arbre, on obtient les éléments minimaux $Ter(A)$ de la hiérarchie. Quand on ajoute successivement les arêtes par ordre croissant de valeurs, on obtient des composantes connexes (des sous-arbres) de plus en plus larges, qui correspondent aux nœuds intermédiaires du dendrogramme.

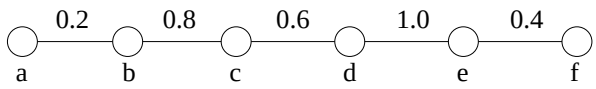


FIG. 3.4 – Arbre aux arêtes valuées représentant une hiérarchie.

Arbre et ultramétrique Sur un arbre T, toute paire de nœuds est reliée par une chaîne unique (théorème 3.1). Considérons la chaîne d'arêtes reliant les nœuds v_i et v_j, $C_{ij} = (e^1, e^2, \ldots, e^k)$. L'arbre considéré ayant les arêtes valuées, à chaque arête e^l de la chaîne correspond une valeur (ou poids) w_l. On peut définir une ultramétrique sur l'arbre T comme :

$$d(v_i, v_j) = \underset{l \in C_{ij}}{sup} \{w^l\} \tag{3.38}$$

C'est-à-dire, la distance entre deux nœuds de l'arbre est définie comme la valeur maximale sur la chaîne unique reliant les deux nœuds.

Les boules maximales de cette ultramétrique de rayon plus petit que r s'obtiennent en supprimant les arêtes de l'arbre de valeur supérieure ou égale à r. En effet, les boules de rayon plus petit que r correspondent aux ensembles de nœuds à distance plus petite que r du centre. En supprimant les arêtes de valeur supérieure ou égale à r sur l'arbre on obtient une forêt formée de plusieurs arbres (composantes connexes). Les nœuds appartenant à un même arbre sont à distance inférieure à r entre eux, car toutes les arêtes reliant les nœuds ont une valeur inférieure à r. Ainsi, chaque arbre (composante connexe) de la forêt résultante correspond à une boule de rayon inférieur à r. Ces boules sont maximales, car on ne peut pas en obtenir de plus grandes et de rayon inférieur à r.

Relation entre le dendrogramme et l'arbre aux arêtes valuées Le dendrogramme peut être déduit à partir de l'arbre selon l'algorithme 3.3.

Algorithme 3.3 Construction du dendrogramme à partir de l'arbre aux arêtes valuées

T : Arbre aux arêtes valuées
D : Dendrogramme
pile ← Arêtes de l'arbre, ordonnées par ordre croissant de valeur
while NoVide(*pile*) **do**
  e_{ij} ← ExtraireArête(*pile*)
  v_i, v_j ← TrouverExtremitésArête(T, e_{ij})
  a_1, a_2 ← TrouverSommets(D, v_i, v_j)
  D ← CréerNouveauSommet(D, a_1, a_2)
end while

Afin d'illustrer le fonctionnement de l'algorithme 3.3, regardons comment le dendrogramme de la figure 3.3 est déduit à partir de l'arbre de la figure 3.4. Les étapes à suivre sont présentées dans la figure 3.5. Initialement, le dendrogramme ne contient que les nœuds correspondant aux éléments isolés. Au début, donc, le dendrogramme contient autant de sommets que d'éléments. Les arêtes de l'arbre sont traitées par ordre croissant de valeur. La première arête à être traitée

est celle reliant les nœuds a et b. Les sommets du dendrogramme contenant chacun de ces nœuds sont les nœuds eux-mêmes, et ils seront donc fusionnés sur le dendrogramme, pour donner lieu à un nouveau sommet, ab. Les prochaines arêtes sont celles reliant les nœuds e-f et ensuite c-d, pour lesquels de nouveaux sommets sont également produits sur le dendrogramme. Ensuite, c'est l'arête b-c qui doit être traitée. Les sommets correspondants sur le dendrogramme sont ab et cd, qui seront fusionnés dans un nouveau nœud $abcd$. Finalement, la dernière arête à être traitée est l'arête d-e, et les sommets contenant ces nœuds, $abcd$ et ef, sont fusionnés dans le sommet unique du dendrogramme, $abcdef$.

L'arbre aux arêtes valuées définit de façon unique le dendrogramme. Cependant, le contraire n'est pas vrai. Afin de le démontrer, essayons de passer du dendrogramme de la figure 3.3 à l'arbre de la figure 3.4. Les deux premiers nœuds à fusionner sur le dendrogramme sont les nœuds a et b. Sur l'arbre, il faudra donc ajouter une arête entre ces mêmes nœuds, valuée avec la valeur de l'indice de stratification, 0.2. Les deux prochains éléments à fusionner sur le dendrogramme sont e et f, avec un indice de 0.4, ce qui se traduit sur l'arbre par une arête entre ces mêmes nœuds, avec une valeur de 0.4. Ensuite, ce sont les nœuds c et d qui fusionnent de la même façon. Le problème apparaît quand les composantes connexes – ou classes – formées par les éléments ab et cd fusionnent. On doit ajouter une arête sur l'arbre afin d'enregistrer cette fusion, mais il y a *quatre possibilités*, toutes équivalentes du point de vue du dendrogramme. En effet, doit-on ajouter l'arête entre les nœuds a-c, a-d, b-c ou b-d ? Dans la figure 3.4 nous avons choisi une des possibilités, mais elle n'est pas la seule. Le dendrogramme ne définit donc pas de manière unique l'arbre aux arêtes valuées. **La représentation d'une hiérarchie moyennant un arbre aux arêtes valuées est *plus riche* que le dendrogramme**, car l'arbre aux arêtes valuées indique les chemins reliant deux éléments alors que le dendrogramme indique uniquement leur distances.

L'arbre de poids minimum L'arbre valué tel que nous venons de le présenter peut représenter différentes hiérarchies selon les chemins reliant les éléments et les valeurs des arêtes. À chaque distribution des valeurs des arêtes correspond une ultramétrique différente ayant un rapport différent avec l'indice de distance s. L'arbre de poids minimum est un cas particulier d'arbre valué, pour lequel les valeurs des arêtes sont données par l'indice de distance s. Ainsi, l'arbre de poids minimum représente une hiérarchie particulière dont le rapport avec l'indice de distance s est bien connu. Nous l'étudions ici par son rapport très étroit avec les processus d'inondation, que nous verrons dans le chapitre 6.

Définition 3.21 *Considérons un ensemble d'éléments $\{i\} \in I$ et un indice de distance entre les éléments $s(i,i')$. Considérons également l'ensemble d'arbres $T = (X,V)$ ayant pour nœuds les éléments $X = \{\{i\} \in I\}$ et dont les arêtes $e_{ij} \in V$ sont valuées avec l'indice de distance entre les extrémités de l'arête, $w_{ij} = s(i,j)$.* **L'arbre de poids minimum** *est un arbre dont le poids total, donné par :*

$$\sum_{e_{ij} \in V} w_{ij} = \sum_{e_{ij} \in V} s(i,j) \tag{3.39}$$

est minimum

Si toutes les $s(i,j)$ ont des valeurs distinctes, alors l'arbre de poids minimum est unique.

L'arbre de poids minimum définit une ultramétrique, donnée par l'équation 3.38. Cette ultramétrique est l'ultramétrique sous-dominante [22] de l'indice s, ce qui correspond aux techniques de construction de hiérarchies par chaînage simple.

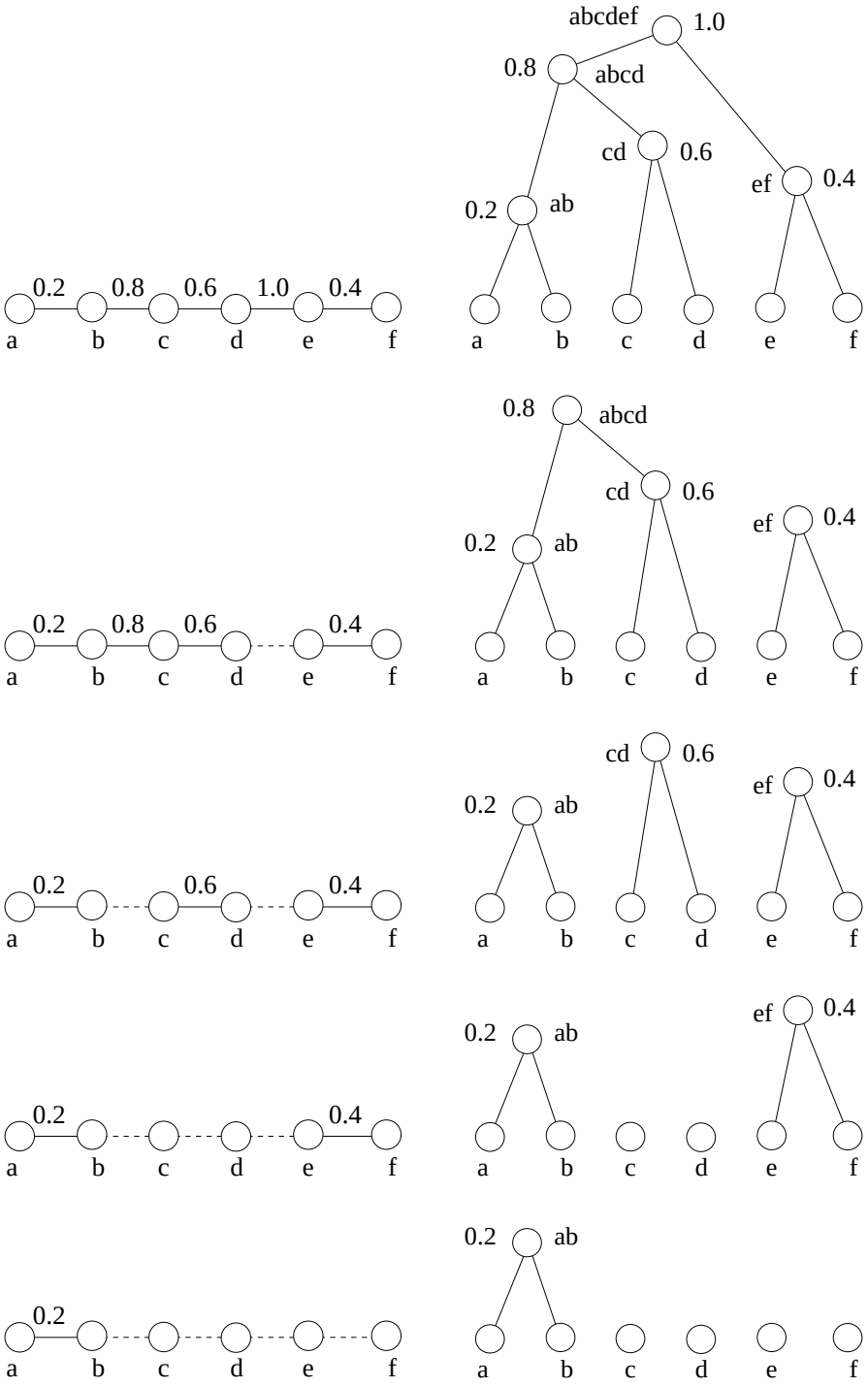


FIG. 3.5 – Construction du dendrogramme à partir de l'arbre aux arêtes valuées.

Le problème de l'arbre de poids minimum est souvent formulé dans un contexte de graphes. Ainsi, nous considérons le graphe non-orienté $G = (X, U)$ dont l'ensemble de nœuds X est donné par les éléments $\{i\} \in I$ et où les valeurs w_{ij} des arêtes $e_{ij} \in U$ donnent les indices de distance $s(i, j)$ entre les nœuds. Nous exprimerons également la valeur d'une arête u comme $w(u)$.

Les deux théorèmes suivants caractérisent complètement l'arbre de poids minimum :

Première caractérisation Considérons l'arbre de poids minimum $T = (X, V)$ d'un graphe $G = (X, U)$. En éliminant une arête $u \in V$ de l'arbre T on obtient deux composantes connexes de nœuds $A \subset X$ et $\overline{A} = X \backslash A$. Le cocycle θ^u sur G est donné par l'ensemble des arêtes $v \in U$ ayant une extrémité en A et l'autre en $\overline{A}$. Le théorème suivant exprime que l'arête u appartenant à l'arbre est celle de plus petit poids du cocycle.

Théorème 3.2 *$T = (X, V)$ est un arbre de poids minimum de $G = (X, U)$ si et seulement si pour toute arête $u \in U$, on a $w(u) \leq w(v)$ $\forall v \in \theta^u$ $(u \neq v)$.*

Deuxième caractérisation Considérons maintenant les arêtes de G n'appartenant pas à l'arbre de poids minimum, $u \in U \backslash V$. Chacune de ces arêtes relie deux nœuds $a, b \in X$. Sur l'arbre de poids minimum, ces deux nœuds sont reliés par un chemin unique, et ajouter l'arête u introduira nécessairement un cycle μ^u. Le théorème suivant exprime que l'arête u est celle de plus grande valeur appartenant au cycle.

Théorème 3.3 *$T = (X, V)$ est un arbre de poids minimum de G si et seulement si pour toute arête $u \in U \backslash V$, le cycle μ^u formé en ajoutant cette arête à l'arbre T satisfait $w(u) \geq w(v)$ $\forall v \in \mu^u$.*

Considérons la suppression une à une, par ordre croissant de valeurs, des arêtes d'un graphe $G = (X, V)$. Cette suppression produira des composantes connexes sur le graphe G. Au fur et à mesure que plus d'arêtes sont supprimées, les composantes connexes seront partitionnées en composantes connexes plus petites : on obtient une hiérarchie de parties d'un ensemble (l'ensemble des nœuds de l'arbre). Cette hiérarchie est la même que celle qu'on obtient moyennant suppression par ordre croissant de valeurs des arêtes de l'arbre de poids minimum de G.

Un arbre valué quelconque T représente une hiérarchie. Entre deux nœuds quelconques v_1 et v_2 de T il existe un chemin unique de valeur maximale λ. Ajouter une arête entre v_1 et v_2 de valeur plus grande ou égale à λ ne change pas à la hiérarchie induite par T (hiérarchie associée à la suppression d'arêtes par ordre croissant de valeurs). De cette manière, bien que T ne contient pas toutes les distances entre les points v_i, il est possible de construire des graphes G à partir de T dont la hiérarchie obtenue par suppression d'arêtes est la même que celle représentée par T : on construit un graphe G tel que T est sont arbre de poids minimum. Remarquons que donné un arbre T, il n'y a pas un unique G compatible avec T.

3.6 Hiérarchie de partitions emboîtées d'une image

Une *image* I est formée par un ensemble de pixels : $I = \{p_1, p_2, \ldots, p_n\}$. Une *région* R est un sous-ensemble connexe de pixels de l'image. Une *partition* P est un ensemble de régions $P = \{R_1, R_2, \ldots, R_k\}$ tel que :

– l'union des régions de la partition donne l'ensemble de départ : $I = \bigcup_{i=1,\ldots,k} R_i$;

– les régions ont une intersection nulle : $\forall i, j, i \neq j \; R_i \bigcap R_j = \varnothing$.

Une *famille de partitions emboîtées* d'une image est un ensemble de partitions $A = \{P_0, P_1, \ldots, P_l\}$ tel que les régions de la partition $P_i = \{R_1^i, R_2^i, \ldots, R_k^i\}$ sont incluses dans les régions de la partition $P_j = \{R_1^j, R_2^j, \ldots, R_k^j\}$, $j > i$: $R_m^i \subseteq R_p^j$ ou $R_m^i \cap R_p^j = \varnothing$. C'est-à-dire, deux régions quelconques appartenant à des partitions différentes de la famille sont soit disjointes, soit incluses l'une dans l'autre. On appelle la partition P_i le *niveau* i de la famille. P_0 est la partition la plus fine, et constitue le niveau inférieur de la famille. P_l est la partition la plus grossière (niveau supérieur). Les régions du niveau i sont toujours incluses dans les régions des niveaux supérieurs $j > i$. Ainsi, chaque partition de la famille est obtenue par fusion de deux ou plusieurs régions du niveau inférieur (plus fin). La figure 3.6 illustre ce concept. En parcourant la famille de bas en haut, des régions sont fusionnées pour former des partitions plus grossières, emboîtées les unes dans les autres.

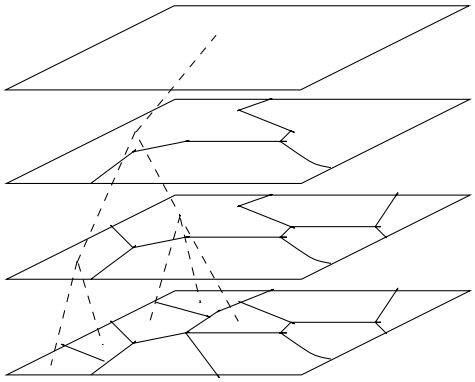


FIG. 3.6 – Famille de partitions emboîtées.

L'ensemble $A = \{P_0, P_1, \ldots, P_n\}$ des partitions emboîtées d'une image constitue une hiérarchie de parties de l'ensemble I, puisqu'il satisfait aux axiomes 3.8 et 3.9. L'axiome d'intersection est vérifié parce que les régions à un niveau de la hiérarchie sont obtenues par fusion de régions d'un niveau inférieur, ce qui implique que deux régions quelconques sont soit disjointes soit incluses l'une dans l'autre. L'axiome de réunion est également vérifié : chacune des régions de la hiérarchie est repartitionée à des niveaux inférieurs, et elle peut donc être obtenue par union des régions plus petites qu'elle et incluses dans elle.

À partir d'une famille de partitions d'une image, on peut définir la distance ultramétrique entre toute paire de pixels p_m et p_l de l'image I comme l'indice i correspondant au niveau de la famille pour lequel les deux pixels ont été fusionnés dans une même région pour la première fois. Si $I \notin A$ (le niveau supérieur a plus d'une région), il est possible que ces pixels n'appartiennent jamais à la même région. Dans ce cas, la distance entre les deux pixels est infinie. La partition P_i de la famille est formée des boules maximales de l'ultramétrique de rayon plus petit ou égal à i.

La hiérarchie de partitions emboîtées peut être représentée par un dendrogramme, où chaque nœud représente une région R_m^i de la hiérarchie, et les arcs représentent les relations d'inclusion entre les régions. L'indice de stratification de chaque nœud correspond à l'indice i de la partition P_i la plus fine contenant cette région. De la même manière, il est possible de représenter la hiérarchie moyennant un arbre aux arêtes valuées, mais la construction de l'arbre à partir de la famille de partitions ne donne pas de solution unique, comme nous l'avons

montré dans la section 3.5.3.

3.7 Les hiérarchies de partitions en segmentation d'image

Nous présentons ici quelques méthodes de construction de hiérarchies qui ont été utilisées en segmentation d'image. Les techniques ascendantes sont beaucoup plus utilisées que les méthodes descendantes, qui sont en général plus coûteuses en temps de calcul. En segmentation d'image, et quelle que soit la méthode de construction de la hiérarchie, le choix de l'indice de dissimilarité entre les régions – ou classes – détermine la pertinence des partitions obtenues par rapport au contenu de l'image. Nous décrivons ici quelques mesures de dissimilarité qui ont été utilisées pour la construction de hiérarchies de partitions emboîtées d'une image. Souvent, la mesure choisie pour exprimer la dissimilarité entre les régions d'un niveau de la hiérarchie ne s'exprime pas facilement en fonction de la dissimilarité entre pixels.

3.7.1 Les approches descendantes

Ces approches obtiennent la hiérarchie par divisions successives des régions. La méthode décrite dans [105] est basée sur le calcul de la coupe minimale. Ici, ce sont des mesures de similarité – et non pas de dissimilarité – qui sont définies entre régions adjacentes. La valeur – ou capacité – d'une coupe est définie comme la somme de similarités entre régions adjacentes des deux côtés de la coupe. Le problème consiste à trouver la partition, ou coupe, de valeur minimale. Le calcul de la coupe minimale, pour lequel il existe des algorithmes efficaces [70], est cependant biaisé vers la détection de petites régions. Pour cette raison, le calcul de coupes normalisées a été proposé [93], [103] où la taille des régions est prise en considération.

Différentes mesures de similarité ont été proposées dans le cadre des hiérarchies obtenues par coupes minimales :

- des différences de niveau de gris ou de couleur [105], [93] ;
- réponse à des bancs de filtres (détection de texture) [93], [95] ;
- mesures basées sur l'orientation des contours [42] ;
- mesures basées sur le mouvement [94].

L'approche descendante proposée dans [84] et [79] est basée sur la morphologie mathématique et est très différente des techniques de segmentation par coupes minimales. Des filtres connexes sont utilisés pour élargir les zones plates de l'image. Ensuite, les zones plates les plus significatives selon des critères de contraste et de taille sont utilisées comme marqueurs pour une LPE sur le gradient d'une version filtrée de l'image originale. Les régions ainsi obtenues sont modelées et le résidu par rapport à l'image originale est calculé. Les régions ayant un résidu important sont mal représentées, et doivent donc être resegmentées dans l'étape suivante. Ainsi, dans l'étape suivante l'image originale est filtrée avec un filtre de plus petite taille que dans l'étape précédente afin de préserver plus de détails, et les régions ayant un grand résidu sont resegmentées. Le processus est itéré jusqu'à ce qu'un suffisamment faible résidu soit obtenu, ce qui indique une bonne représentation de toutes les régions de l'image. Puisque c'est la LPE sur l'image gradient qui est utilisée pour trouver la partition à partir des marqueurs, la mesure de dissimilarité entre régions est donnée par la valeur plus faible du gradient au long de la coupe. En effet, la LPE situera la frontière entre les deux marqueurs sur la coupe au long de laquelle la valeur minimale est plus grande.

3.7.2 Les approches ascendantes

Les hiérarchies ascendantes sont construites en fusionnant à chaque itération des régions selon un certain critère. Une mesure de dissimilarité entre régions est implicitement ou explicitement définie. À chaque itération, le couple de régions avec dissimilarité minimale est fusionné, comme nous l'avons décrit dans l'algorithme 3.1.

Différentes mesures de dissimilarité ont été proposées. Souvent, l'indice de dissimilarité est basé sur des mesures déterministes des caractéristiques des régions. Les plus simples consistent en une différence des moyennes des niveaux de gris [61], [62]. Ce critère est intéressant quand les régions sont très homogènes, mais il crée des frontières artificielles en cas de transitions graduelles du niveau de gris. D'ailleurs, ce critère privilégie les régions très contrastées indépendamment de leur taille, qui peuvent correspondre à du bruit. D'autres mesures de dissimilarité considèrent également la variance et la surface des régions [69], [2], [60]. Dans [43], seule une bande autour de la frontière entre régions adjacentes est considérée pour le calcul quand les régions sont suffisamment grandes, ce qui évite les erreurs qui se produisent en cas de transitions graduelles du niveau de gris. D'autres critères sont basés sur la similarité de mouvement [43], [82] ou de texture [43] des régions. Il est également possible de combiner des critères de fusion différents selon le niveau de la hiérarchie. Il est aussi raisonnable d'utiliser des critères plus complexes pour les niveaux les plus élevés, puisqu'on dispose de plus d'informations [60]. Dans d'autres approches, un modèle est adapté à chacune des régions. Celles ayant des modèles similaires sont fusionnées en priorité. Dans [92] la fonction de coût est le résidu quand le modèle d'une des régions est adapté à l'autre région.

Dans le cadre de la morphologie mathématique, les premières approches hiérarchiques ont été conçues pour éliminer la sursegmentation qui apparaît quand on applique la ligne de partage des eaux sans marqueurs. Une stratégie consiste à inonder chaque bassin versant au niveau de son col le plus bas afin de réduire le nombre de bassins versants – et donc le nombre de régions – de l'image [5], [6]. En itérant ce processus plusieurs fois on obtient des partitions de plus en plus grossières. Des hiérarchies de partitions ont été également produites par classification des minima de l'image. La première proposition de hiérarchisation des minima de l'image a été faite par Grimaud [23], qui a défini la dynamique. Ainsi, à chaque minimum de l'image gradient peut être associée sa dynamique (profondeur du bassin versant associé), ce qui introduit une hiérarchie. Ensuite, des segmentations à différents niveaux de résolution peuvent être obtenues en sélectionnant les minima de l'image avec la dynamique la plus forte et en les utilisant comme marqueurs pour la LPE. Un nombre plus élevé de marqueurs donnera lieu à des segmentations plus détaillées. Les partitions obtenues sont emboîtées puisque les contours obtenus sont un sous-ensemble de ceux obtenus par calcul de LPE à partir de tous les minima. La hiérarchisation des minima a été généralisée dans [98] pour donner les valeurs d'extinction surfaciques et volumiques. Dans [71] (commentaires sur cet article dans [41] et [86]), au lieu de hiérarchiser les minima, ce sont les contours de la LPE qui sont valués avec une mesure de dynamique. Le principal avantage de cette approche est que les partitions correspondant aux différents niveaux de la hiérarchie sont obtenues par un simple seuillage sur les valeurs des contours.

Dans les approches statistiques, chaque région est caractérisée par une fonction de densité de probabilité. Idéalement, il s'agit de fusionner les régions dont la fonction de densité de probabilité est la même. Des probabilités d'erreur sont calculées pour les hypothèses de fusion et non-fusion, et une décision est prise pour chaque couple de régions candidates. À chaque itération, les marges d'erreur tolérées augmentent [2].

3.8 Conclusions

Dans ce chapitre nous avons présenté des notions mathématiques en relation avec les hiérarchies de parties d'un ensemble, dont les hiérarchies de partitions emboîtées d'une image sont un cas particulier. Nous avons vu l'équivalence entre la notion de distance ultramétrique et celle de hiérarchie de parties d'un ensemble. Cette équivalence permet de combiner des hiérarchies moyennant des algèbres d'ultramétriques. Nous avons également présenté deux approches générales pour construire des hiérarchies : la construction de bas en haut, et la construction de haut en bas. Les hiérarchies peuvent se représenter à l'aide de graphes, ce qui simplifie leur manipulation. Nous avons vu deux représentations sous forme d'arbre : le dendrogramme et l'arbre aux arêtes valuées. Finalement, nous avons présenté le cas des partitions emboîtées d'une image ainsi qu'une brève description des approches qui ont été proposées pour la construction de hiérarchies de partitions d'une image.

Dans le chapitre 4 nous proposons différents outils permettant à l'utilisateur d'interagir avec une famille de partitions emboîtées d'une image afin de définir de manière simple et rapide les objets d'intérêt. Plus tard, dans le chapitre 6, nous traitons la construction de hiérarchies de partitions emboîtées d'images fixes basées sur des inondations, dont nous montrerons la relation avec l'arbre de poids minimum. Notre approche est donc une approche de construction ascendante de la hiérarchie. Ces techniques sont généralisées à la construction de hiérarchies de séquences vidéo dans le chapitre 7.

Chapitre 4

L'interaction avec l'utilisateur

4.1 Introduction

Ayant défini dans le chapitre 3 les caractéristiques principales d'une hiérarchie de partitions emboîtées, nous présentons ici les différentes possibilités qu'une telle approche offre du point de vue de l'interaction avec l'utilisateur.

Les partitions emboîtées s'adaptent très bien à la structure hiérarchique d'une scène. Par exemple, une personne peut être décomposée en tête et corps, ensuite la tête peut se décomposer en visage et cheveux, et encore le visage contient les yeux, la bouche, le nez. Cette structure hiérarchique est très intuitive et s'adapte bien à un système interactif. Les décompositions de la scène faites par les algorithmes n'étant pas souvent parfaites – encore moins s'il s'agit d'images tout venant – il est nécessaire que les outils mis à la disposition de l'utilisateur soient suffisamment flexibles pour permettre d'obtenir les objets d'intérêt avec peu d'effort.

Un certain nombre de logiciels commerciaux destinés à l'infographie offrent des « outils de masquage », qui permettent de définir des objets pour leur manipulation ultérieure. Ces outils sont plutôt simples et leurs résultats ne sont en général pas très satisfaisants. Dans la section 4.2 nous décrivons ces outils et suggérons des améliorations basées sur les hiérarchies. Mais l'existence d'une hiérarchie de partitions emboîtées ouvre les portes à de nouvelles possibilités d'interaction, que nous présentons dans la section 4.3. Dans tous les cas nous supposerons que nous sommes capables de calculer, pour une image donnée, une hiérarchie de partitions emboîtées. La qualité de la hiérarchie déterminera en grande partie l'efficacité des méthodes ici proposées. Nous décrirons les techniques de calcul de telles hiérarchies dans les chapitres suivants.

4.2 Les outils d'interaction existants sur le marché

Il existe des logiciels commerciaux d'édition tels que PaintShopPro, PhotoPaint ou PhotoShop qui offrent des fonctionnalités basiques de segmentation. D'autres logiciels, tels que KnockOut, sont destinés uniquement à la segmentation et proposent des algorithmes plus évolués. Bien que dans ces logiciels les résultats de la sélection soient toujours visualisés sous forme de contour, on peut classer les mécanismes d'interaction comme basés sur les régions ou basés sur les contours. Les mécanismes basés sur les régions cherchent à définir l'ensemble des pixels constituant l'objet, tandis que ceux basés sur les contours cherchent plutôt à définir les pixels séparant l'objet du fond. Notre hiérarchie étant basée sur les régions, elle pourra être

utilisée naturellement pour améliorer les outils basés sur les régions.

4.2.1 Description

Nous décrivons ici une partie des outils de segmentation – ou *masquage*, dans le langage infographique – que l'on trouve dans la plupart des logiciels commerciaux d'édition.

La baguette magique La baguette magique est un outil basé sur les régions ayant pour but de segmenter des régions homogènes en couleur. L'utilisateur clique avec la souris sur un point de l'image, et donne en même temps une tolérance de couleur. L'ensemble connexe de pixels contenant le point de départ et dont l'écart en couleur par rapport à ce point est inférieur à la tolérance est détecté. L'algorithme sous-jacent est une croissance de la région où la tolérance est utilisée comme critère d'arrêt. Les régions ainsi définies ont très souvent des contours très irréguliers et sont très sensibles au bruit. En effet, les pixels isolés à l'intérieur des grandes régions homogènes ne sont pas détectés. Ce problème ne peut pas être résolu en utilisant une tolérance plus grande, car cela provoquerait une fusion du fond.

Le pinceau Le pinceau est un outil purement manuel basé sur les régions. L'utilisateur se sert d'un pinceau de taille variable pour définir l'intérieur de l'objet. Des pinceaux de grande taille sont utilisés pour remplir rapidement l'intérieur des grands objets, mais un travail de précision avec des pinceaux très fins est nécessaire pour définir les contours exacts.

Le lasso Le lasso est un outil dont la mise en œuvre diffère selon les logiciels. Dans PaintShopPro, il regroupe toutes les interactions basées sur les contours. Il a une option *main levée*, où le contour est dessiné à la main, une option *point à point* permettant de définir l'objet par des morceaux de ligne droite et une option *contour intelligent*, qui correspond aux ciseaux intelligents que nous décrivons dans le paragraphe suivant. D'autres logiciels permettent d'approcher un contour par des courbes de type Bézier. PhotoPaint sépare les outils que nous venons de décrire dans des fonctions différentes, et propose un outil lasso qui permet à l'utilisateur de dessiner un contour approximatif qui doit être placé à l'extérieur de l'objet. En même temps, une tolérance de couleur est spécifiée. Le contour se contracte jusqu'à ce qu'une variation de couleur par rapport à la tolérance donnée soit détectée.

Les ciseaux intelligents Les ciseaux intelligents sont un outil basé sur les contours, où les contours de l'objet sont définis en plusieurs morceaux. L'utilisateur clique sur un point de départ appartenant au contour. Un rectangle est défini, ayant ce point comme point d'ancrage. Le contour le plus fort à l'intérieur du rectangle est détecté. La taille du rectangle est définie différemment selon les logiciels. Ainsi, dans certains cas elle est fixe, tandis que dans d'autres cas elle varie avec la position de la souris, le rectangle étant plus grand lorsque la position de la souris s'éloigne du point de départ. Quand l'utilisateur clique à nouveau, la portion du contour en cours est fixée et le point choisi sert d'ancrage à un nouveau morceau de contour.

Les contours externes et internes Le logiciel Corel KnockOut est un logiciel spécifiquement conçu pour la segmentation et dont les mécanismes d'interaction diffèrent considérablement de ceux que nous venons de présenter. Le mécanisme de base consiste à définir approximativement un contour externe et un contour interne de l'objet à segmenter. Le contour exact

est automatiquement détecté entre les deux contours définis par l'utilisateur. Bien qu'à l'heure actuelle ce logiciel semble avoir du succès parmi les infographistes, le fait que deux contours soient nécessaires pour définir l'objet peut être gênant.

4.2.2 Améliorations basées sur une famille de partitions emboîtées

Nous présentons ici des améliorations importantes qui peuvent être obtenues sur quelques uns des outils précédents lorsqu'on dispose d'une hiérarchie de partitions emboîtées. Nous ne donnons pas de détails de mise en œuvre, que nous aborderons dans le chapitre 6.

4.2.2.1 La baguette magique

La baguette magique a pour but de détecter les régions homogènes en couleur, la notion d'homogénéité étant donnée par une tolérance. Une version très améliorée de cet outil et basée sur une hiérarchie de partitions consiste à prendre la boule maximale de l'ultramétrique centrée sur le point donné par l'utilisateur telle que la distance entre la couleur du pixel et la couleur moyenne de la région soit plus petite que la tolérance. Autrement dit, on va chercher la région contenant le pixel sélectionné dans le niveau le plus grossier possible dans la hiérarchie de partitions, en prenant soin que la distance entre la couleur moyenne de la région et la couleur du pixel ne dépasse pas le seuil de tolérance. Des seuils de tolérance plus grands permettront de monter plus haut dans la hiérarchie, et donc de détecter des régions plus grandes.

Puisque la sélection ne se fait plus au niveau du pixel, mais au niveau des régions, la qualité des résultats obtenus du point de vue des contours et de la sensibilité au bruit augmente considérablement. Nous montrons dans la figure 4.1 les segmentations obtenues avec le logiciel PhotoPaint, et avec notre version améliorée. Les résultats sont présentés pour une seule intervention de l'utilisateur (un seul « click » de souris). On remarque les trous et la moins bonne qualité des contours obtenus avec PhotoPaint, ainsi que l'absence d'une partie de l'objet à cause d'une légère gradation de la couleur.

(a) Image originale (b) PhotoPaint (c) Notre version

FIG. 4.1 – Baguette magique basée sur une hiérarchie de partitions emboîtées.

4.2.2.2 Le pinceau intelligent

Le pinceau présente l'inconvénient principal de nécessiter beaucoup d'effort de la part de l'utilisateur pour définir les contours avec suffisamment de précision. Disposer d'un pinceau dont la forme s'adapterait aux contours des objets rendrait la tâche beaucoup plus facile. Or la mise en place d'un tel outil se fait très naturellement à partir de la hiérarchie de partitions

emboîtées. En effet, la forme d'un tel pinceau est donnée par la boule de l'ultramétrique centrée sur la position de la souris, la taille du pinceau déterminant le rayon de la boule. On peut exprimer cette idée de façon équivalente en disant que l'on prend la région contenant la position de la souris, et que la « taille » du pinceau correspond au niveau ou partition de la hiérarchie sur lequel les régions sont choisies. Ainsi, la forme du pinceau change au fur et à mesure que l'utilisateur déplace la souris sur l'image pour s'adapter aux contours des objets. La figure 4.2 illustre les contours obtenus avec l'outil pinceau classique (figure 4.2-(b)) et l'outil basé sur la hiérarchie (figure 4.2-(c)) pour une même trajectoire du pinceau (figure 4.2-(a)). Tandis que les contours produits avec le pinceau classique dépendent de la forme du pinceau (dans ce cas circulaire), ceux obtenus avec le pinceau intelligent basé sur la hiérarchie dépendent du contenu de l'image.

(a) Trajectoire (b) Pinceau fixe (c) Pinceau intelligent

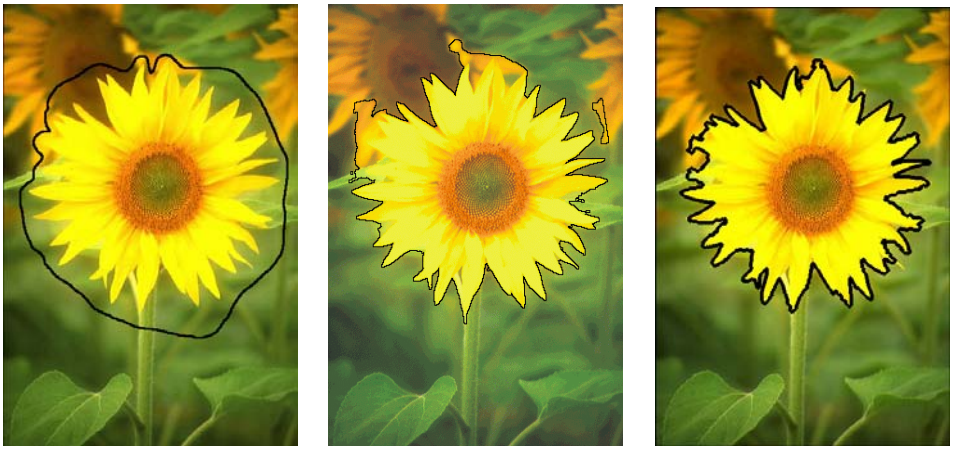
FIG. 4.2 – Comparaison entre les résultats obtenus avec le pinceau fixe et le pinceau intelligent.

4.2.2.3 Le lasso

Nous considérons ici la variante de l'outil lasso où l'utilisateur dessine un contour extérieur approximatif de l'objet. On va chercher à trouver un contour totalement contenu à l'intérieur du contour dessiné par l'utilisateur et correspondant à de forts contours de l'image. Ceci est fait en deux étapes :

- partant des boules de l'ultramétrique de rayon zéro (niveau plus fin de la hiérarchie), on fait croître le rayon tant que des boules extérieures au contour ne sont pas ajoutées ;
- à partir de l'ensemble de boules intérieures au contour obtenues dans l'étape précédente, des fusions entre boules internes au contour sont produites. Cet étape n'est pas équivalente à augmenter le rayon des boules, car les fusions de boules extérieures au contour sont interdites. L'ordre de fusion entre les boules est donné par leur distance ultramétrique (les boules plus proches sont fusionnées d'abord). Les fusions continuent jusqu'à ce que la région résultante ait une surface proche à un pourcentage près – réglable par l'utilisateur – de la surface incluse à l'intérieur du contour dessiné.

La figure 4.3 illustre les résultats obtenus avec l'outil lasso de PhotoPaint et l'outil que nous proposons. On remarquera que le lasso de PhotoPaint ajoute des régions parasites (en haut et à gauche du tournesol) qui n'appartiennent pas à l'objet. Cet effet est d'autant plus important lorsque le fond n'est pas homogène en couleur.



(a) Contour dessiné (b) Résultat PhotoPaint (c) Résultat hiérarchie

FIG. 4.3 – Comparaison entre l'outil lasso de PhotoPaint et le lasso basé sur une hiérarchie de partitions emboîtées.

4.2.3 Les contours externes et internes

Le type d'interaction par introduction de contours internes et externes que propose le logiciel KnockOut de Corel rappelle la segmentation morphologique avec des marqueurs, avec la différence qu'en segmentation morphologique il n'est pas toujours nécessaire que les marqueurs soient placés près des vrais contours de l'objet. En effet, en segmentation morphologique il suffit qu'ils soient complètement contenus à l'intérieur de chaque région d'intérêt, car la ligne de partage des eaux trouve les plus forts contours séparant deux marqueurs. Cependant, si les marqueurs se situent près des contours, les chances d'aboutir à des résultats satisfaisants sont plus importantes. En effet, si les marqueurs sont très séparés il est possible que le plus fort contour entre les deux marqueurs ne corresponde pas à celui de l'objet. Si les marqueurs sont près l'un de l'autre, la région d'incertitude où les contours seront placés est plus petite.

En utilisant les méthodes morphologiques, le type d'interaction par des contours intérieurs et extérieurs à l'objet peut être étendu à un cadre plus général, où un marqueur est dessiné pour chaque objet à segmenter, le fond étant aussi considéré comme un objet. Le résultat sera une partition de l'image avec autant de régions que de marqueurs différents. La figure 4.4 montre les marqueurs introduits par l'utilisateur et le résultat obtenu. L'idée est de trouver les frontières entre les objets de façon à ce que :

- chaque marqueur soit complètement inclus dans la région associée ;
- la dissimilarité entre régions soit maximisée.

Idéalement, si l'utilisateur a introduit n marqueurs, on voudrait trouver le rayon des boules qui donnent une partition de l'image en n régions, telle que chacune des boules contienne exactement un marqueur. Or, en général, les marqueurs introduits par l'utilisateur ne seront pas compatibles avec l'ultramétrique utilisée pour définir la hiérarchie. C'est-à-dire que la partition correspondant au niveau de la hiérarchie avec n régions ne contiendra pas un marqueur unique par région. En effet, si c'était le cas il n'y aurait pas besoin de marqueurs.

La segmentation à partir de marqueurs quand on dispose d'une famille de partitions emboîtées est alors conçue comme l'union de toutes les boules maximales de l'ultramétrique ayant pour centre l'un des points du marqueur et ne contenant pas d'autre marqueur. On trouvera

FIG. 4.4 – Marqueurs dessinés par l'utilisateur et résultat obtenu.

ainsi la meilleure (plus grande distance) séparation entre les marqueurs, du point de vue de l'ultramétrique.

4.3 De nouvelles formes d'interaction

Les outils présents dans les logiciels d'infographie simplifient le problème de la segmentation en le réduisant à la détection d'un seul objet. Ainsi, les versions améliorées de ces outils que nous venons de présenter sont orientées vers la détection d'un seul objet. Si plusieurs objets doivent être extraits, cela est fait objet par objet, en choisissant à chaque fois l'outil le plus pertinent. Les masques ainsi obtenus peuvent se recouvrir. Les logiciels commerciaux ajoutent souvent des fonctionnalités d'union ou d'intersection de plusieurs masques afin de pouvoir les combiner pour obtenir des masques plus complexes.

Cependant, dans certaines applications telles que MPEG-4, on cherche plutôt à obtenir une décomposition complète de la scène en objets sémantiques. Ainsi, l'objectif est d'obtenir une partition plus ou moins complexe des images. Les objets ne peuvent pas se recouvrir, et l'union des objets donne l'image entière. Les hiérarchies de partitions offrent des mécanismes d'interaction permettant à l'utilisateur de composer des partitions complexes en combinant différents niveaux de la hiérarchie. Les outils que nous présentons dans cette section sont orientés vers l'obtention de telles partitions.

La qualité de la hiérarchisation aura une influence directe sur la quantité de travail et de temps dont l'utilisateur aura besoin pour compléter sa tâche.

L'autre facteur qui aura une grande influence sur l'utilité d'un système interactif est la façon dont l'interaction avec l'utilisateur est menée. Il est souhaitable que l'utilisateur puisse opérer d'une façon intuitive et rapide, sans avoir à acquérir des connaissances spécifiques sur les algorithmes qui sont à la base du système.

4.3.1 Sélection d'un niveau

Étant donnée une famille de partitions emboîtées, l'utilisateur peut *naviguer* entre les différents niveaux de résolution de la hiérarchie pour choisir le niveau le mieux adapté au résultat recherché. Dans le cas idéal, l'utilisateur trouvera un niveau où l'objet d'intérêt se présente comme une seule région. Dans un cadre plus général, on pourra choisir un niveau comme point de départ pour d'autres interactions.

Ce type d'interaction consiste à choisir le rayon des boules de l'ultramétrique associées à la hiérarchie. Pour chaque valeur du rayon d on a une partition de l'image, d'autant plus fine que le rayon est petit.

La figure 4.5 montre les boules de l'ultramétrique pour la première image de la séquence « Bream » pour trois valeurs différentes du rayon. L'ultramétrique choisie comme indice de stratification pour cet exemple sera présentée dans le chapitre 6.

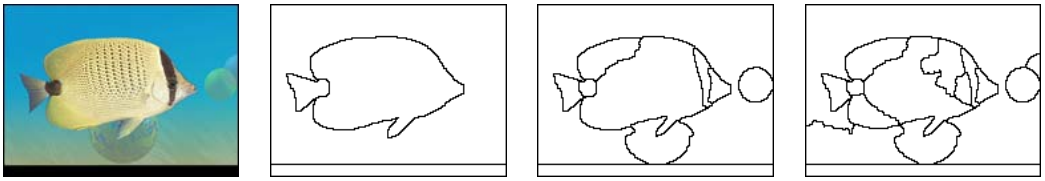


FIG. 4.5 – Image originale et boules maximales de l'ultramétrique pour trois valeurs différentes du rayon.

Du point de vue de l'utilisateur, la sélection d'un niveau de la hiérarchie peut se faire à l'aide d'une barre de défilement associée à une fenêtre de visualisation de la partition. L'utilisateur fait bouger la barre pour accéder à des partitions plus fines ou plus grossières de façon rapide et conviviale. Cette méthode a été expérimenté avec succès dans le logiciel d'extraction d'objets vidéo VOGUE (cf. chapitre 8).

4.3.2 Composition de différents niveaux

Bien qu'un seul niveau de la hiérarchie puisse donner une segmentation satisfaisante dans quelques cas, il est souvent nécessaire de combiner plusieurs niveaux de résolution pour arriver au résultat souhaité. Ceci est spécialement vrai quand certaines parties de l'image nécessitent d'être segmentées avec plus de résolution que d'autres.

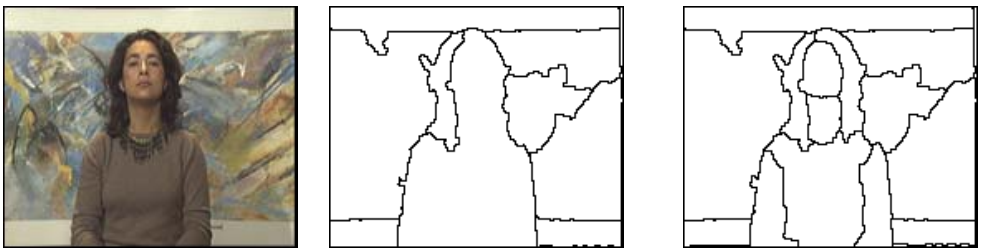
La composition de niveaux de la hiérarchie se fait en choisissant des boules de l'ultramétrique avec des rayons qui varient selon leur position dans l'image. C'est grâce à une des propriétés de l'ultramétrique que cette association de boules de rayons différents est possible. En effet, les boules ne peuvent pas se couper sans être contenues complètement l'une dans l'autre. En diminuant le rayon d'une boule, on obtient alors une partition de cette boule, formée par des boules plus petites.

Du point de vue de l'utilisateur, la combinaison de boules de différents rayons peut être présentée de plusieurs manières :

1. Initialement, l'utilisateur choisit un niveau de la hiérarchie comme point de départ, par exemple avec la barre de défilement décrite dans la section 4.3.1. Ceci donne une partition de l'image. Ensuite, l'utilisateur peut choisir une région (en cliquant avec la souris) et solliciter une resegmentation ou une fusion de la région choisie. Dans le cas d'une fusion, la hiérarchie détermine la région la plus similaire (plus proche du point de vue de la distance ultramétrique), avec laquelle la région sera fusionnée. Seulement la région choisie – ainsi que son entourage dans le cas d'une fusion – sera affectée, le reste de la partition restant intact. De façon interne, l'algorithme réduira ou augmentera respectivement le rayon de la boule sélectionnée, sans toucher aux boules restantes. Cette méthode a été choisie dans VOGUE, décrit dans le chapitre 8.

2. L'utilisateur se sert d'un pinceau dont la forme s'adapte aux régions et dont la taille est contrôlée par l'utilisateur. Cet outil est celui que nous avons présenté dans la section 4.2.2.2.

La figure 4.6 montre les possibilités issues du mélange de niveaux de la hiérarchie. La figure 4.6-(a) montre l'image originale. La figure 4.6-(b) montre un niveau de la hiérarchie avec 10 régions. Si l'on veut avoir la présentatrice segmentée avec plus de détail, il faudrait aller chercher un niveau plus fin de la hiérarchie. Cependant, des niveaux plus fins segmenteraient avec plus de détail non seulement la locutrice, mais aussi le fond, qui est très texturé. Il est alors possible de ne resegmenter qu'une ou plusieurs régions de la partition courante (figure 4.6-(b)), dans notre cas l'intérieur de la locutrice, sans toucher au reste de la partition. Le résultat est illustré dans la figure 4.6-(c).



(a) Image originale. (b) 10 régions (c) Resegmentation d'une région

FIG. 4.6 – Image originale, niveau de la hiérarchie avec 10 régions et résultat obtenu en combinant des niveaux de résolution différents pour la première image de la séquence « Silent Voice ».

4.3.3 Incorporation des outils dans un logiciel de segmentation

Nous venons de présenter un ensemble d'outils de segmentation basés sur une hiérarchie de partitions emboîtées. Ces outils peuvent être intégrés de diverses manières dans un logiciel d'édition selon le domaine d'application et selon que l'on cherche à extraire un seul objet ou à obtenir une partition de l'image. Lorsqu'on cherche à extraire un objet unique dans des logiciels d'infographie, il suffit de remplacer les outils traditionnels tels que le pinceau ou le lasso par et les nouvelles versions que nous avons proposé. On peut également introduire la segmentation par marqueurs. L'utilisateur choisit un outil, qui sert à obtenir un masque. Si le masque obtenu n'est pas complètement satisfaisant, il existe la possibilité de le modifier à la main (ajouter ou éliminer des pixels), ou bien de le combiner avec d'autres masques (union ou intersection), qui peuvent avoir été obtenus avec des outils différents. Une fois le masque obtenu est suffisamment satisfaisant, il est utilisé pour couper-coller l'objet sur une autre image, ou pour appliquer un traitement à l'objet sans toucher au reste de l'image, etc.

Dans des applications cherchant à obtenir une partition plus complexe, comme par exemple des logiciels d'édition MPEG-4, les outils de la section 4.3 sont plus convenables. Ici, l'utilisateur obtient une partition initiale à l'aide d'outils tels que l'accès à un niveau de la hiérarchie ou la segmentation par marqueurs. À partir de cette partition, l'utilisateur resegmente et fusionne localement des régions jusqu'à ce que la partition obtenue soit satisfaisante. À ce point il convient de remarquer que les fusions proposées par la hiérarchie ne correspondent pas toujours à ce que l'utilisateur cherche. Ainsi, afin d'obtenir une flexibilité maximale il convient de mettre également à disposition de l'utilisateur un outil de fusion manuelle des régions permettant de réaliser des fusions non prévues par la hiérarchie.

4.4 Conclusions

Dans ce chapitre, nous avons présenté une batterie d'outils d'interaction, tous basés sur une hiérarchie de partitions emboîtées. Certains de ces outils sont des versions – beaucoup plus performantes – d'outils que l'on trouve dans des logiciels commerciaux d'infographie. C'est le cas de la baguette magique, le lasso, le pinceau et les contours intérieurs/extérieurs. Nous avons également proposé une autre classe d'outils plus orientés vers la manipulation de partitions : ce sont la sélection du nombre de régions de la partition et la composition de niveaux différents par resegmentation/fusion de régions.

Remarquons que tous les outils présentés ont une interprétation très intuitive et que l'utilisateur n'a besoin d'aucune connaissance sur les algorithmes sous-jacents.

Il nous faut à présent voir comment on peut construire des hiérarchies sur des images fixes et des séquences. En effet, les outils d'interaction seront de peu d'utilité si les contours trouvés par les hiérarchies ne sont pas adaptés au contenu des images. Nous présentons la construction de telles hiérarchies dans les chapitres 6 et 7. Le chapitre 5 introduit l'étape de pré-traitement de notre système ainsi que la notion de zone quasi-plate, que nous utiliserons ensuite pour la construction des hiérarchies.

Chapitre 5

Prétraitement des images couleur

Le filtrage est le premier maillon d'une chaîne de traitement d'images. Il prépare l'image à l'application des algorithmes en réduisant le niveau de bruit et en la simplifiant. Certains types de filtres accomplissent cette tâche avec un coût : la perte de précision des contours. Un fort filtrage produit ainsi un lissage ou une déformation des contours. Ce coût n'est pas admissible dans des applications de segmentation, où l'information à extraire est précisément la localisation exacte des contours des objets.

Avec les opérateurs connexes [35], [91], [90] la morphologie mathématique permet une forte simplification des images à traiter tout en conservant la position des contours des régions qui persistent après filtrage. Cependant, la seule condition de connexité n'assure pas que le sens de variation des contours soit respecté, et n'impose aucune condition sur l'amplitude des contours à la sortie par rapport à leur amplitude à l'entrée du filtre. Ainsi, un filtre connexe peut très bien transformer une transition faible à l'entrée en une forte transition à la sortie. Cette caractéristique nuit à la segmentation, qui a besoin de quantifier l'intensité des contours. Les nivellements [54], [55], [47], [90], sont des opérateurs connexes qui, moyennant l'imposition de conditions additionnelles, fournissent un outil idéal pour la segmentation.

La notion d'opérateur connexe est étroitement liée à celle de zone plate. En effet, les opérateurs connexes ne font qu'élargir les zones plates. D'autre part, la présence de grandes zones plates dans l'image à traiter simplifie énormément la segmentation.

Dans ce chapitre, nous présentons le filtrage par nivellement en rapport avec la segmentation. Les nivellements ayant été initialement définis pour les images à niveaux de gris, peu de travail a encore été fait pour les images couleur. La première partie de ce chapitre traite les nivellements, aussi bien pour les images à niveaux de gris que pour les images en couleur. Dans un deuxième temps, nous présentons les zones plates et leur rapport étroit avec la segmentation. Nous montrons que, pour les images à niveaux de gris, les techniques de segmentation basées sur les zones plates peuvent être améliorées sensiblement moyennant l'introduction de la notion de zone *quasi*-plate. D'autre part, la couleur est souvent traitée dans la littérature en appliquant indépendamment sur chacune des bandes des algorithmes conçus pour les images à niveaux de gris. Nous proposons une extension de la notion de zone quasi-plate aux images couleur, ce qui permet de mieux exploiter l'information couleur.

La section 5.1 introduit la nomenclature que nous utiliserons par la suite. Ensuite, la section 5.2 présente les nivellements pour les images à niveaux de gris. La section 5.3 présente les travaux antérieurs étendant les nivellements aux images couleur. Nous proposons une nouvelle approche dans la section 5.4. La section 5.5 aborde les zones plates et quasi-plates pour

les images à niveaux de gris et leur application à la segmentation. La section 5.6 présente les travaux antérieurs proposant l'application de la notion de zone plate aux images couleur. Enfin, nous proposons dans la section 5.7 une extension de la notion de zone quasi-plate aux images couleur. Finalement, dans la section 5.8 nous analysons les caractéristiques des zones quasi-plates couleur associées aux nivellements.

5.1 Images à niveaux de gris et images couleur

Une image à niveaux de gris est une fonction scalaire définie sur un support bidimensionnel $f : \mathbb{R}^2 \to \mathbb{R}$. Les images 3D à niveaux de gris sont définies sur un support tridimensionnel, $f : \mathbb{R}^3 \to \mathbb{R}$.

La numérisation des images implique une quantification du support ainsi que des valeurs des images. Ainsi, en général on travaille avec des images définies sur les entiers, c'est-à-dire, sur des fonctions $f : \mathbb{Z}^2 \to \mathbb{Z}$. En pratique, une image f n'est définie que pour un sous-ensemble $E \subset \mathbb{Z}^2$, qu'on appelle *support* de la fonction. L'ensemble des fonctions à niveaux de gris forme un treillis pour la relation d'ordre :

$$f \leq g \Leftrightarrow f(x) \leq g(x) \, \forall x \in E \tag{5.1}$$

dont le sup et l'inf sont donnés par :

$$(f \vee g)(x) = f(x) \vee g(x) \quad \forall x \in E \tag{5.2}$$
$$(f \wedge g)(x) = f(x) \wedge g(x) \quad \forall x \in E \tag{5.3}$$

La définition de ce treillis a amené au développement de la morphologie mathématique pour les fonctions scalaires.

Les images couleur sont des fonctions *vectorielles* définies sur un support bi ou tridimensionnel $f : \mathbb{R}^2 \to \mathbb{R}^3$, ou, après numérisation, $f : \mathbb{Z}^2 \to \mathbb{Z}^3$. De la même façon que pour les images scalaires, le plus souvent la fonction f n'est définie que sur un sous ensemble $E \subset \mathbb{Z}^2$, qui donne le support de la fonction. À cause de l'inexistence d'une relation d'ordre naturelle entre vecteurs, la morphologie mathématique ne peut pas s'étendre de manière directe aux fonctions vectorielles.

Étant donnée une fonction f, scalaire ou vectorielle, nous noterons par f_s la valeur de la fonction f au point $s \in E$, où E est le support de la fonction. Ainsi, f_s notera une valeur scalaire ou un vecteur, selon que la fonction f représente une image à niveaux de gris ou une image couleur.

5.2 Filtrage sur des images à niveaux de gris

5.2.1 Zones plates et opérateurs connexes

Définition 5.1 *(G. Matheron et J. Serra) Étant donnés un ensemble I et l'ensemble $\mathcal{P}(I)$ des parties de I, une* **connection** *est une famille $\mathcal{C} \subseteq \mathcal{P}(I)$ telle que :*

$$\varnothing \in \mathcal{C} \tag{5.4}$$
$$\forall x \in I : \{x\} \in I \tag{5.5}$$
$$\textit{pour toute famille } C_i \subseteq \mathcal{C} : \bigcap_i C_i \neq \varnothing \Rightarrow \bigcup_i C_i \in \mathcal{C} \tag{5.6}$$

Classe d'équivalence associée à une connexion On peut associer à une connexion $\mathcal{C}$ une relation binaire $\mathcal{R}$ de la manière suivante :

$$\forall x, y \in I \,:\, x \,\mathcal{R}\, y \;\Leftrightarrow\; \exists C \in \mathcal{C} \,|\, x, y \in C \tag{5.7}$$

La relation binaire $\mathcal{R}$ ainsi définie est une relation d'équivalence. En effet, elle est réflexive, car $\{x\} \in \mathcal{C}$. Elle est évidemment symétrique, car $x, y \in C \;\Leftrightarrow\; y, x \in C$. Finalement, $\mathcal{R}$ est bien transitive :

$$\forall x, y, z \in I \,:\, (x \,\mathcal{R}\, y) \;\wedge\; (y \,\mathcal{R}\, z) \;\Leftrightarrow\; (x, y \in C_1) \;\wedge\; (y, z \in C_2)$$

et comme $C_1 \cap C_2 \neq \varnothing$, on a, par l'axiome 5.6 de la connexité :

$$(x, y \in C_1) \;\wedge\; (y, z \in C_2) \;\Rightarrow\; x, y, z \in (C_1 \cup C_2) \quad\Leftrightarrow\quad x \,\mathcal{R}\, z$$

Étant donnée la relation d'équivalence $\mathcal{R}$ associée à la connection $\mathcal{C}$, l'ensemble des classes d'équivalence de $\mathcal{R}$ correspond aux classes connexes maximales de $\mathcal{C}$, ce qui constitue une partition. Inversement, à toute partition $P = \{P_i \mid i = 1, \ldots, n\}$ on peut faire correspondre une relation d'équivalence $\mathcal{R}$ telle que $x \,\mathcal{R}\, y \Leftrightarrow \exists\, i \mid x, y \in P_i$. Ainsi, toute classe d'équivalence définit une partition, et à toute partition on peut associer une classe d'équivalence.

Connexion induite par une classe d'équivalence Étant donnée une classe d'équivalence $\mathcal{R}$ définie sur l'ensemble $\mathcal{P}(I)$, l'ensemble $\{P_i\} \cup \{x\} \cup \varnothing$, (où $\{P_i\}$ sont les classes de la partition associée à $\mathcal{R}$, et $\{x\}$ est l'ensemble des éléments isolés ou « singletons ») définit une connexion. En effet, il est simple de voir que les axiomes de la définition 5.1 sont vérifiés. On voit bien donc qu'une classe d'équivalence induit une connexion.

Construction d'une relation d'équivalence à partir d'une relation binaire À partir d'une relation binaire réflexive et symétrique $\mathcal{R}$, c'est-à-dire une relation binaire qui vérifie, pour toute paire de points $p, q \in I$:

$$p \,\mathcal{R}\, p \tag{5.8}$$

$$p \,\mathcal{R}\, q \;\Rightarrow\; q \,\mathcal{R}\, p \tag{5.9}$$

il est possible de définir une relation d'équivalence $\tilde{\mathcal{R}}$ de la manière suivante :

$$p \,\tilde{\mathcal{R}}\, q \;\Leftrightarrow\; \exists\, a_1, a_2, \ldots, a_n \in I \mid p \,\mathcal{R}\, a_1 \,\mathcal{R}\, a_2 \,\mathcal{R} \ldots \mathcal{R}\, a_n \,\mathcal{R}\, q \tag{5.10}$$

On vérifie aisément que $\tilde{\mathcal{R}}$ est bien une relation d'équivalence :
- la réflexivité est donnée par la relation $\mathcal{R}$, qui est elle-même réflexive ;
- la relation $\tilde{\mathcal{R}}$ est également symétrique, car $\mathcal{R}$ l'est, et donc à toute série $a_1 \,\mathcal{R}\, a_2 \,\mathcal{R} \ldots \mathcal{R}\, a_n$ correspond la série inverse $a_n \,\mathcal{R} \ldots \mathcal{R}\, a_2 \mathcal{R}\, a_1$;
- $\tilde{\mathcal{R}}$ est aussi transitive – même si $\mathcal{R}$ ne l'est pas – grâce à la définition de $\tilde{\mathcal{R}}$ par des paires de points vérifiant $\mathcal{R}$. En effet, étant donnés trois points $p, q, r \in I$, si $p \,\tilde{\mathcal{R}}\, q$ alors il existe $p \,\mathcal{R}\, a_1 \,\mathcal{R}\, a_2 \,\mathcal{R} \ldots \mathcal{R}\, a_n \,\mathcal{R}\, q$. De la même manière, si $q \,\tilde{\mathcal{R}}\, r$, il existe un $q \,\mathcal{R}\, a'_1 \,\mathcal{R}\, a'_2 \,\mathcal{R} \ldots \mathcal{R}\, a'_n \,\mathcal{R}\, r$. La série formée par l'union des a_i et les a'_i vérifie la relation $p \,\tilde{\mathcal{R}}\, r$.

Remarquons qu'on n'impose pas de conditions sur les a_i, sauf d'être en nombre dénombrable. En pratique, nous travaillerons sur des images, où les points sont définis sur la grille numérique. De manière générale, les points $a_1, a_2, \ldots, a_n$ ne sont pas nécessairement voisins. Cependant, dans notre cas nous travaillerons avec des chemins de points sur lesquels on imposera des relations de voisinage (les $a_1, a_2, \ldots a_n$ doivent être voisins deux à deux), ainsi que des restrictions sur leur niveau de gris. Ainsi, dans notre contexte, $\mathcal{R}$ représentera à la fois une relation de voisinage et une relation entre les valeurs des pixels. On peut à titre d'exemple considérer une fonction f et une relation $\mathcal{R}$ telle que :

$$p\,\mathcal{R}\,q \;\Leftrightarrow\; f_p = f_q \quad \forall\, p, q \text{ voisins}$$

qui inclut en même temps une relation de voisinage et une condition sur les niveaux de gris des pixels p, q. On obtient ainsi une connexion, donnée par les classes d'équivalence de $\tilde{\mathcal{R}}^f$, où $\tilde{\mathcal{R}}^f$ indique la relation d'équivalence $\tilde{\mathcal{R}}$ sur les valeurs de la fonction f.

Opérateur connexe

Définition 5.2 *Un opérateur connexe ϕ associé à la relation d'équivalence $\tilde{\mathcal{R}}$ transforme une fonction f en une fonction $g = \phi(f)$ de façon à ce que la relation suivante soit vérifiée :*

$$p\,\tilde{\mathcal{R}}^f\,q \;\Rightarrow\; p\,\tilde{\mathcal{R}}^{\phi(f)}\,q \qquad \forall\, p, q \in I \tag{5.11}$$

Une opérateur connexe élargit les classes connexes de f, car deux points appartenant à la même classe connexe pour f appartiennent aussi à la même classe connexe pour $\phi(f)$.

Remarque : Il est évident que si on impose la condition plus forte $p\,\mathcal{R}^f\,q \;\Leftrightarrow\; p\,\mathcal{R}^{\phi(f)}\,q$, alors on a aussi que $p\,\tilde{\mathcal{R}}^f\,q \;\Rightarrow\; p\,\tilde{\mathcal{R}}^{\phi(f)}\,q$ et on a donc un opérateur connexe.

Pour une étude approfondie et générale de la connexité ainsi que des exemples plus détaillés, le lecteur pourra consulter [90], [85].

Nous particularisons maintenant la relation $\mathcal{R}$, afin de définir la connexion associée aux **zones plates**, de la manière suivante :

$$p\,\mathcal{R}\,q \;\Leftrightarrow\; f_p = f_q \quad \forall\, p, q \text{ voisins}$$

ce qui résulte en la définition de zone plate :

Définition 5.3 *Deux points $x, y \in E$ appartiennent à la même* **zone plate** *d'une fonction f si et seulement si il existe une chaîne de points $(p_1, p_2, \ldots, p_n)$ telle que $p_1 = x$ et $p_n = y$ et, pour tout i, $(p_i, p_{i+1}) \in I$ sont voisins et $f_{p_i} = f_{p_{i+1}}$.*

Notons qu'il n'y a pas de restriction sur la taille minimale des zones plates, et en conséquence elles peuvent être constituées d'un seul point. Notons aussi que l'ensemble des zones plates d'une image constitue une partition du support de cette image puisque tout pixel de l'image appartient à une et une seule zone plate. En opposition avec les zones plates nous définissons les *zones de transition* : deux points voisins (p, q) forment une transition s'ils n'appartiennent pas à la même zone plate ($f_p > f_q$ ou $f_p < f_q$).

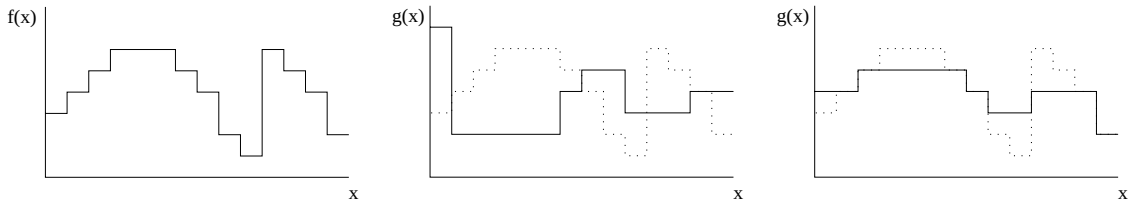
Pour cette connexion, l'opérateur connexe ϕ s'exprime :

$$\forall (p, q)\ \textit{voisins} : \qquad f_p \;=\; f_q \;\Rightarrow\; g_p \;=\; g_q \tag{5.12}$$

C'est-à-dire, à toute zone plate dans f correspond forcément une zone plate dans g. En conséquence, le résultat g de l'opérateur a, au plus, le même nombre de zones plates que la

fonction de départ f, et à toute transition dans g doit correspondre une transition dans f. Cet opérateur agit donc par fusion des zones plates de la fonction de départ.

La figure 5.1 montre deux exemples d'opérateurs connexes sur une fonction monodimensionnelle. La figure 5.1-(a) montre la fonction de départ f, sur laquelle deux opérateurs connexes ont été appliqués pour donner les résultats illustrés dans les figures 5.1-(b) et 5.1-(c). La fonction f (en pointillé) est superposée à la fonction résultat g pour une meilleure appréciation de l'évolution des zones plates. L'opérateur montré dans la figure 5.1-(c) est, en plus, un nivellement, que nous introduisons dans la section 5.2.2.



(a) Fonction de départ (b) Opérateur connexe (c) Nivellement

FIG. 5.1 – Des opérateurs connexes sur une fonction monodimensionnelle.

5.2.2 Les nivellements

5.2.2.1 Définition

Définition 5.4 *Une fonction g est un* **sous-nivellement** *d'une fonction f si et seulement si pour toute paire de points voisins p et q :*

$$g_p > g_q \quad \Rightarrow \quad g_q \geq f_q \tag{5.13}$$

Définition 5.5 *Une fonction g est un* **sur-nivellement** *d'une fonction f si et seulement si pour toute paire de points voisins p et q :*

$$g_p > g_q \quad \Rightarrow \quad g_p \leq f_p \tag{5.14}$$

Définition 5.6 *Une fonction g est un* **nivellement** *de la fonction f si et seulement si elle est un sous- et un sur-nivellement de f :*

$$g_p > g_q \quad \Rightarrow \quad f_p \geq g_p > g_q \geq f_q \tag{5.15}$$

Les nivellements agissent en imposant deux conditions supplémentaires à la condition de connexité :

- La réduction de la dynamique des transitions à la sortie. C'est-à-dire, pour chacune des transitions de la fonction g, la transition correspondante sur f a une amplitude égale ou plus grande que celle de g.
- Le couplage du sens de variation des transitions. Ainsi, à une transition de haut en bas sur la fonction g correspondra forcément une transition de haut en bas sur f, et vice versa.

La figure 5.1-(c) illustre un nivellement. La fonction f est dessinée en pointillé, le nivellement g en ligne continue.

5.2.2.2 Algorithmes

Étant donnée une fonction f, que nous appellerons fonction *référence*, et une fonction quelconque g que nous appellerons *marqueur*, il existe des algorithmes permettant de transformer la fonction marqueur g en un nivellement g' de f.

On obtient le sous- ou sur-nivellement de f à partir d'un marqueur g par itération jusqu'à idempotence des algorithmes suivants, où δ et ϵ sont respectivement la dilatation et érosion morphologiques par l'élément structurant de taille 1 :

1. Sous-nivellement. Ces deux algorithmes sont équivalents :
   - Algorithme niv^- : pour toute paire de pixels voisins (p, q), si $g_p < f_p$ et $g_q > g_p$, alors $g'_p = f_p \wedge g_q$.
   - Algorithme niv^δ : pour tout pixel p, si $g_p < f_p$, alors $g'_p = f_p \wedge (\delta g)_p$.
2. Sur-nivellement.
   - Algorithme niv^+ : pour toute paire de pixels voisins (p, q), si $g_p > f_p$ et $g_p > g_q$, alors $g'_p = f_p \vee g_q$.
   - Algorithme niv^ϵ : pour tout pixel p, si $g_p > f_p$, alors $g'_p = f_p \vee (\epsilon g)_p$.

En appliquant le sous-nivellement quand $g < f$ et le sur-nivellement quand $g > f$ on obtient un algorithme parallèle autodual des nivellements, détaillé dans 5.1, et qui consiste à itérer jusqu'à idempotence la transformation élémentaire $g' = (f \wedge \delta g) \vee \epsilon g$. Cet algorithme est donné par souci de simplicité, mais les nivellements peuvent être programmés de façon plus efficace moyennant des files d'attente hiérarchiques [51].

Algorithme 5.1 Algorithme pour la construction des nivellements

```
repeat
  stop ← vrai
  for all p do
    if g_p > f_p then
      g'_p ← niv^δ
    else
      g'_p ← niv^ε
    end if
    if g'_p ≠ g_p then
      stop ← faux
    end if
  end for
until stop = vrai
```

Le choix de la fonction marqueur g détermine le degré de simplification obtenu. Ainsi, des simplifications plus ou moins fortes peuvent être obtenues en choisissant des marqueurs plus ou moins éloignés de la fonction de référence f.

La figure 5.2 montre le nivellement d'une image à niveaux de gris, avec le marqueur correspondant. Dans ce cas, le marqueur utilisé est un filtre alterné séquentiel sans reconstruction

de taille 5 appliqué sur l'image originale. La figure 5.3 montre un autre nivellement (plus précisément, un sous-nivellement) de la même image avec un marqueur beaucoup plus simple, qui vaut zéro partout sauf aux maxima de l'image de dynamique supérieure à 130, où la valeur de la fonction marqueur est égale à la dynamique. Remarquons que dans ce cas la simplification obtenue est beaucoup plus importante.

(a) Référence (b) Marqueur (c) Nivellement

FIG. 5.2 – Nivellement de l'image Lena, en utilisant comme marqueur un filtre alterné séquentiel de taille 5.

(a) Marqueur (b) Nivellement

FIG. 5.3 – Nivellement de l'image Lena, en utilisant comme marqueur les maxima de dynamique plus grande que 130.

5.2.3 Nivellements étendus

Les définitions 5.4 et 5.5 peuvent être généralisées en considérant de nouvelles relations binaires. Ainsi, la version généralisée des nivellements est donnée par les définitions suivantes :

Définition 5.7 *Une fonction g est un* **sous-nivellement** *d'une fonction f si et seulement si pour toute paire de points voisins p et q :*

$$g_p \succ g_q \quad \Rightarrow \quad g_q \sqsupseteq f_q \tag{5.16}$$

où $\succ$ *et* $\sqsupseteq$ *sont des relations binaires.*

Définition 5.8 *Une fonction* g *est un* **sur-nivellement** *d'une fonction* f *si et seulement si pour toute paire de points voisins* p *et* q *:*

$$g_p \succ g_q \quad \Rightarrow \quad g_p \sqsubseteq f_p \tag{5.17}$$

Selon la définition des relations $\succ$ et $\sqsubseteq$ on a des nivellements différents. À titre d'exemple, on peut considérer les relations définies comme :

$$g_p \succ g_q \Leftrightarrow g_p > g_q + \lambda \tag{5.18}$$
$$g_q \sqsupseteq f_q \Leftrightarrow g_q \geq f_q \tag{5.19}$$

Cette définition nous mène vers une redéfinition de la notion de transition, et donc de celle de zone plate. En effet, on dira que deux points p et q voisins forment une transition si $g_p \succ g_q$ ou $g_q \succ g_p$, c'est-à-dire si $g_p > g_q + \lambda$ ou $g_q > g_p + \lambda$. Dans le cas contraire, nous dirons que les deux points p et q appartiennent à la même zone plate. Ainsi, une zone plate (ou « quasi plate ») sera composée de points tels que pour tout couple de voisins $g_q - \lambda \leq g_p \leq g_q + \lambda$. La figure 5.4 montre un exemple de ce type de nivellement, avec $\lambda = 3$ et le même marqueur que dans la figure 5.2.

De la même façon, toute re-définition des relations binaires a pour conséquence de produire de nouvelles zones « quasi plates », aux caractéristiques différentes.

(a) Référence (b) Nivellement

FIG. 5.4 – Nivellement étendu de l'image Lena avec $\lambda = 3$.

5.3 Travaux antérieurs sur les nivellements couleur

À défaut d'une formulation générale de la morphologie mathématique vectorielle, quelques efforts ont été réalisés pour donner des versions vectorielles des nivellements, que nous décrivons par la suite.

5.3.1 Nivellements composante par composante

L'approche la plus simple consiste à appliquer un nivellement scalaire aux trois composantes couleur indépendamment. C'est l'extension la plus directe et immédiate des nivellements

scalaires. Elle présente comme inconvénient de ne pas traiter les couleurs comme un tout et de ne pas être invariante par rotation des axes coordonnés. La solution dépend également de l'espace couleur choisi.

5.3.2 Nivellements pseudo-scalaires et autarciques

Dans [20], Gomila propose deux solutions au problème :

- Les nivellements **pseudo-scalaires**, qui consistent à :
  1. Projeter les vecteurs sur une droite, dite de référence.
  2. Appliquer les nivellements scalaires au module des vecteurs projetés.
  3. Reprojeter les nouvelles valeurs scalaires dans l'espace vectoriel.

  Les résultats du nivellement dépendent étroitement du choix de la droite de référence.
- Les nivellements **autarciques** s'inspirent de l'algorithme de calcul des nivellements scalaires : $g' = f \wedge \delta g \vee \epsilon g$. Cet algorithme peut aussi s'énoncer en disant : « si tous les voisins du pixel p sont du même côté par rapport à la référence f (c'est-à-dire, tous en dessous ou tous au dessus), alors on remplace le marqueur g_p par son voisin le plus proche de la référence, autrement on le remplace par la référence ». Cette interprétation est extrapolée aux vecteurs, mais pour cela il faut définir ce qu'on entend par « même côté ». Dans [20], on considère que g_p et g_q sont du même côté de f_p si l'angle $\widehat{g_p f_p g_q}$ est aigu.
  Le nivellement autarcique a l'avantage de ne pas introduire de nouvelles couleurs et d'être invariant par rotation. Cependant, on ne dispose pas de définition formelle donnant le rapport entre les fonctions marqueur et référence.

5.3.3 Fonction séparatrice dans l'espace vectoriel

Dans [58], F. Meyer établit un cadre théorique permettant d'introduire une définition vectorielle des nivellements. À cet effet, une nouvelle relation d'ordre est introduite, basée sur la notion de *fonction séparatrice*.

Définition 5.9 ***Fonction séparatrice (cas scalaire).*** *Étant données les fonctions scalaires g, h, f, on dit que h sépare g et f, et on note $(g\,h\,f)$ ou $(f\,h\,g)$, si et seulement si pour tout $x \in E$, la série $(g_x\,h_x\,f_x)$ est monotone, c'est-à-dire : $\forall x \in E$ on a $g_x \leq h_x \leq f_x$ ou $g_x \geq h_x \geq f_x$.*

L'extension de cette notion aux espaces vectoriels se fait à travers la définition d'éléments géométriques :

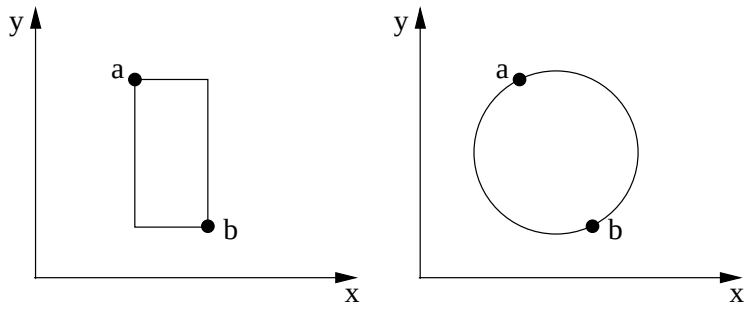
- on définit Boîte(a, b) comme le rectangle ayant les points a et b comme sommets opposés et dont les côtés sont parallèles aux axes de coordonnées ;
- on définit Disque(a, b) comme le cercle ayant le segment $\overline{ab}$ comme diamètre.

La figure 5.5 illustre ces définitions. Les points a, b représentent les valeurs vectorielles de deux fonctions f et g au point p. Par simplicité nous représentons les valeurs vectorielles en deux dimensions (x, y). Dans le cas des images couleur on travaille cependant sur des vecteurs à trois dimensions. Les boîtes sont alors des cubes et les disques des sphères.

Notons qu'il peut exister des configurations particulières :

- si les points a et b ont une coordonnée en commun, alors Boîte(a, b) est une droite ;
- si $a = b$, alors Boîte(a, b) et Disque(a, b) deviennent un seul point.

Finalement, notons que les boîtes ont la propriété agréable que leur intersection est encore une boîte, ce qui n'est pas le cas des disques. À l'inverse, les disques sont invariants par rotation, tandis que la configuration des boîtes dépend du choix du système de coordonnées.



(a) Boîte(a,b) (b) Disque(a,b)

FIG. 5.5 – Définition de boîte et disque.

Étant donnée une boîte B (ou disque D) d'extrémité f_p, on définit l'*opposé* du point f_p comme le point le plus éloigné de f_p appartenant à la boîte (au disque), et on le note Opposé(f_p, B).

Définition 5.10 ***Fonction séparatrice (cas vectoriel).*** *Soient les fonctions vectorielles g, h, f définies sur un support E. On dit que h sépare g et f, et on le note $(g\,h\,f)$ ou $(f\,h\,g)$, si et seulement si pour tout $p \in E$ le point h_p appartient à Boîte(g_p, f_p) (au Disque(g_p, f_p)).*

Une fois établie la notion de fonction séparatrice, il est possible de comparer des fonctions vectorielles.

Définition 5.11 *On dit que g est plus loin de f que h, ou que g est plus grande que h pour l'ordre f, et on écrit $g >_f h$, si et seulement si h sépare g et f :*

$$g >_f h \Leftrightarrow (f\,h\,g) \tag{5.20}$$

Considérons d'abord le cas des boîtes. Étant donné un ensemble de fonctions h^i et une fonction f, l'inf de cet ensemble pour la relation $<_f$ est donné par la plus grande fonction plus petite que tous les h^i. Cette fonction existe et est donnée par :

$$\left(\bigwedge_f h^i\right)_p = \text{Opposé}\left(f_p, \bigcap_i \text{Boîte}\left(h_p^i,\ f_p\right)\right) \quad \forall p \in E \tag{5.21}$$

En effet, l'intersection des boîtes ayant pour extrémités les points f_p et h_p^i donne l'ensemble des fonctions plus petites que tous les h_i pour l'ordre $<_f$. Ainsi, la plus grande de ces fonctions pour l'ordre $<_f$ est donnée par l'opposé du point f_p dans l'intersection, qui est aussi une boîte.

Les définitions précédentes nous permettent maintenant de donner la *soustraction de Minkowski* d'une fonction g par un élément structurant B :

$$g \ominus_f B = \bigwedge_f^{x \in B} g_{-x} \tag{5.22}$$

où g_{-x} est la translation de la fonction g par le vecteur $-x$.

Cette définition revient à :

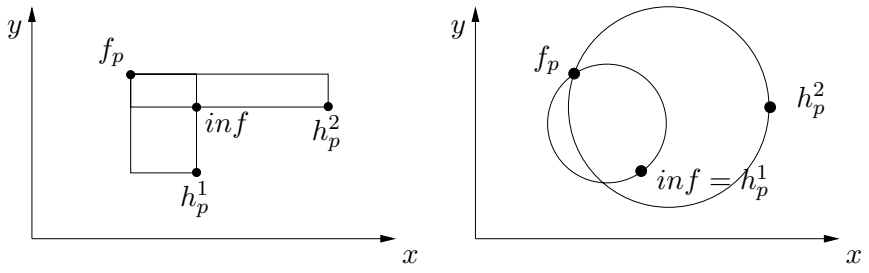
$$(g \ominus_f B)_p = \text{Opposé}\left(f_p, \bigcap_{t \in B} \text{Boîte}\ (f_p, g_{p-t})\right) \tag{5.23}$$

Considérons maintenant le cas des disques. La soustraction de Minkowski est donnée par :

$$(g \ominus_f B)_p = \text{Opposé}\left(f_p, \bigcap_{t \in B} \text{Disque}\ (f_p, g_{p-t})\right) \tag{5.24}$$

Rappelons que l'intersection de disques n'est pas un disque. Nous notons cependant par « opposé » le point le plus éloigné de f_p appartenant à l'intersection.

La figure 5.6 illustre le calcul de l'inf de deux fonctions h^1 et h^2 pour le cas des boîtes et des disques.



(a) Inf de boîtes (b) Inf de disques

FIG. 5.6 – Inf de fonctions vectorielles.

On peut maintenant définir un *nivellement* dans l'espace de vecteurs.

Définition 5.12 *Une fonction g est un $B-$nivellement d'une fonction f si et seulement si g est invariant pour la soustraction de Minkowski par l'élément structurant B :*

$$g = g \ominus_f B \tag{5.25}$$

Ainsi, étant données deux fonctions vectorielles f et g appelées respectivement référence et marqueur, la fonction g peut être transformée en un nivellement g' de f en itérant des soustractions de Minkowski jusqu'à idempotence.

Notons quelques caractéristiques de ces nivellements :

- Dans le cas des boîtes, il est équivalent d'appliquer le nivellement vectoriel par soustraction de Minkowski ou d'appliquer le nivellement scalaire indépendamment à chacune des composantes couleur. Il s'agit donc d'un nivellement *séparable*. Dans ce cas, ce type de nivellement correspond exactement aux nivellements composante par composante (section 5.3.1). Cependant, ce n'est pas le cas des disques, où on ne peut pas traiter séparément les coordonnées, mais pour lesquels on gagne en échange une indépendance vis à vis des axes de coordonnées.
- Les nivellements vectoriels basés sur des disques sont invariants par rotation, ce qui n'est pas le cas des boîtes.

5.3.4 Remarques

Faisons quelques remarques sur les approches que nous venons de présenter :

Nivellements pseudo-scalaires et autarciques Les nivellements pseudo-scalaires dépendent fortement du choix de la droite dite de référence. D'ailleurs, ils ne sont pas invariants par rotation. Les nivellements autarciques présentent la particularité de ne pas créer de nouvelles couleurs, étant donné qu'ils agissent par propagation des couleurs de la fonction marqueur. D'autre part, ces nivellements sont invariants par rotation. Dans les deux cas, l'inconvénient majeur vient du fait que les rapports entre le nivellement et la fonction de référence sont mal définis, leur définition étant donnée de manière algorithmique. En particulier, ils ne sont pas connexes : à une zone plate de la fonction référence ne correspond pas nécessairement une zone plate du nivellement.

Approche basée sur la définition de fonction séparatrice Les nivellements vectoriels définissent un contexte théorique fondé sur la notion de fonction séparatrice, ce qui permet de comparer des vecteurs par rapport aux valeurs de la fonction de référence f. Des deux types proposés, séparables et non séparables, les nivellements séparables dépendent du choix des axes de coordonnées, ce qui peut poser des problèmes dans certaines applications. Les nivellements non séparables ne posent pas ce problème, mais leur calcul est beaucoup plus complexe, car il faut calculer des intersections de plusieurs sphères. De plus, leur calcul est non associatif et non commutatif.

5.4 Nouvelle approche pour les nivellements vectoriels

Nous interprétons ici les nivellements scalaires de deux points de vue différents, ce qui nous permet de nouvelles généralisations au cas vectoriel.

Rappelons d'abord la définition de nivellement scalaire :

$$\forall p, q \text{ voisins} \quad g_p > g_q \Rightarrow f_p \geq g_p > g_q \geq f_q \tag{5.26}$$

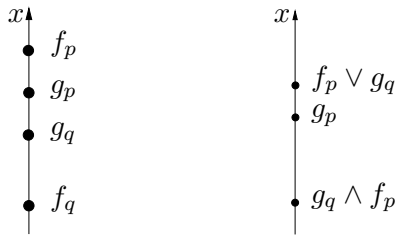
qui peut se formuler de façon équivalente :

$$\forall p, q \text{ voisins} \quad f_p \wedge g_q \leq g_p \leq f_p \vee g_q \tag{5.27}$$

Ces conditions sont exprimées sur la droite des réels dans la figure 5.7, qui montre les positions relatives des points pour que la condition de nivellement soit remplie.

5.4.1 Généralisation moyennant des distances

Nous avons vu qu'il est difficile d'établir une relation d'ordre significative dans l'espace couleur. Comment peut-on établir qu'une couleur est plus grande ou plus petite qu'une autre ? On pourrait par exemple proposer des relations d'ordre basées sur le module des vecteurs, mais cette approche n'exploiterait qu'une petite partie de l'information, et dans la plupart des cas ne serait pas d'accord avec notre perception. Cependant, la notion de distance a un sens clair dans l'espace couleur. En effet, cela a un sens de dire qu'une couleur ressemble plus ou moins à une autre, bien que les mesures de ressemblance existantes n'aient que de lointains rapports



(a) Equation 5.26 (b) Equation 5.27

FIG. 5.7 – Interprétation géométrique des nivellements scalaires.

avec la perception humaine. On peut conclure que si on arrive à formuler une définition en termes de distance, alors l'extension vectorielle est directe. Tel est le cas des nivellements.

Essayons d'abord d'exprimer l'équation 5.26 en termes de distance :

Définition 5.13 *Une fonction g est un nivellement de f si pour toute paire de points p, q voisins, si $g_p \neq g_q$ alors :*

$$d(g_p, g_q) \leq d(f_p, f_q) \tag{5.28}$$

$$\begin{aligned} d(g_p, f_q) &\leq d(f_p, f_q) \\ d(g_q, f_p) &\leq d(f_p, f_q) \end{aligned} \tag{5.29}$$

$$\begin{aligned} d(f_p, g_p) &\leq d(f_p, g_q) \\ d(f_q, g_q) &\leq d(f_q, g_p) \end{aligned} \tag{5.30}$$

où d est une fonction distance définie entre tout couple de points.

- la condition 5.28 exprime une contrainte de distance maximale entre les deux points marqueurs : l'amplitude de toute transition du nivellement doit être plus petite que celle de la référence ;
- la condition 5.29 limite l'éloignement du nivellement par rapport à la référence ;
- finalement, la condition 5.30 garantit que g_p soit plus proche de f_p que g_q, et de même pour g_q.

Ces conditions peuvent maintenant être directement étendues aux champs de vecteurs moyennant la définition d'une distance entre vecteurs. L'interprétation géométrique pour des vecteurs à deux dimensions et avec la distance euclidienne est donnée dans la figure 5.8 : les vecteurs (dans ce cas des points de l'espace bidimensionnel) g_p et g_q doivent appartenir au cercle ayant les points f_p et f_q comme extrémités du diamètre. De plus, g_p doit être plus proche de f_p que de f_q, et de même pour g_q.

5.4.2 Généralisation moyennant des intervalles

Considérons maintenant l'équation 5.27 définissant les nivellements scalaires. Cette définition peut être interprétée en disant que le point g_p doit être compris dans l'intervalle défini par

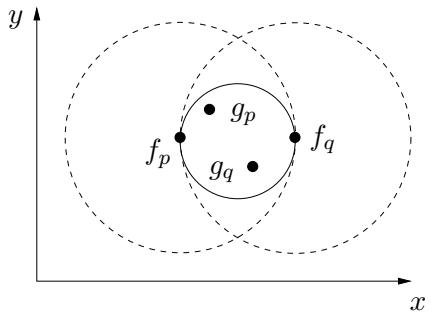


FIG. 5.8 – Interprétation géométrique de la définition 5.13.

f_p et g_q. L'extension vectorielle passe par la définition d'intervalle dans l'espace vectoriel. Une possibilité est de définir une sphère ayant f_p et g_q comme extrémités de son diamètre. C'est ce que nous avons appelé $Disque(f_p, g_q)$ dans la section 5.3.3. Une autre possibilité est de définir une bande aux côtés perpendiculaires au vecteur $\overrightarrow{f_p g_q}$, tel qu'illustré dans la figure 5.9. Dans ce cas le point g_p peut être très éloigné de f_p, même si f_p et g_q sont très proches. Nous avons préféré pour cette raison la définition basée sur des disques, où la distance entre f_p et g_p dépend de la distance entre f_p et g_q.

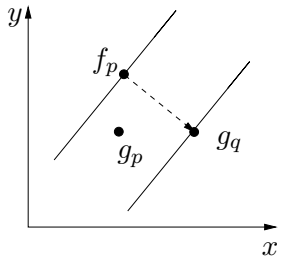


FIG. 5.9 – Nivellement basé sur la définition d'une bande.

Définition 5.14 *Une fonction g est un nivellement vectoriel d'une fonction f si et seulement si pour toute paire de points voisins (p, q) : $g_p \in Disque(f_p, g_q)$.*

Nivellements étendus Dans le même esprit que pour les nivellements étendus à niveaux de gris, on peut définir des λ-nivellements pour des images couleur, afin de produire des filtrages plus forts. Ainsi, la définition 5.14 se transforme en $g_p \in \text{Disque}\left(f_p, g_q + \lambda \frac{\overrightarrow{f_p g_q}}{|f_p g_q|}\right)$, $\lambda \in \mathbb{R}$.

5.4.3 Différences entre les deux généralisations

Si dans le cas scalaire les formulations 5.26 et 5.27 sont équivalentes, ce n'est pas le cas des nivellements vectoriels. Nous analysons maintenant les différences entre les deux généralisations. A ce propos nous étudions quelques implications de la définition 5.14 en rapport avec la définition 5.13.

Proposition 5.1

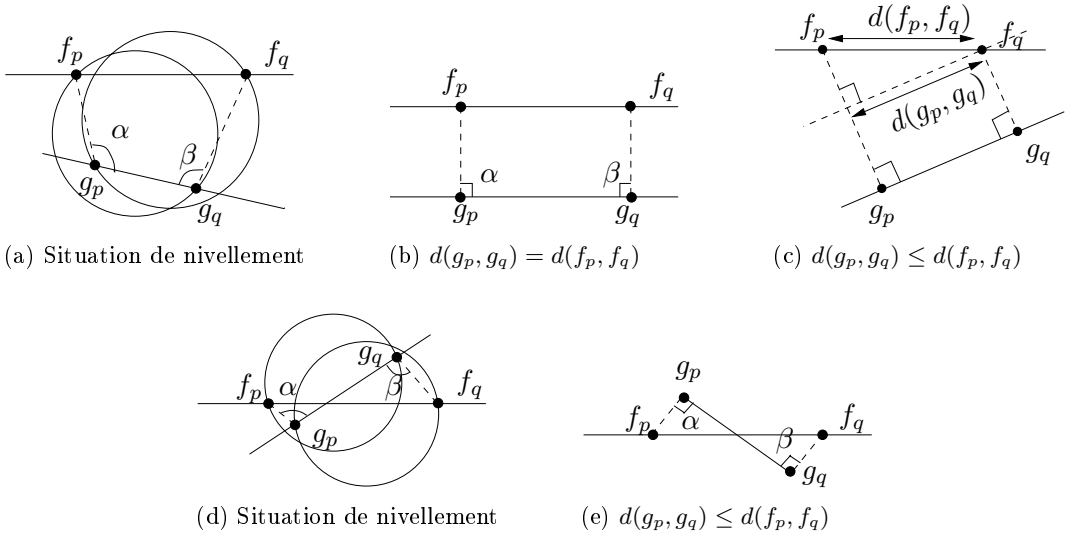
$$g_p \in Disque(f_p, g_q) \Rightarrow d(f_p, g_p) \leq d(f_p, g_q)$$
$$g_q \in Disque(f_q, g_p) \Rightarrow d(f_q, g_q) \leq d(f_q, g_p)$$

Démonstration : si g_p est inclus dans la sphère ayant les points f_p et g_q comme diamètre, forcément la distance entre f_p et g_p est plus petite que le diamètre.

Proposition 5.2

$$\left.\begin{array}{l} g_p \in Disque(f_p, g_q) \\ g_q \in Disque(f_q, g_p) \end{array}\right| \Rightarrow \quad d(g_p, g_q) \leq d(f_p, f_q)$$

Démonstration : si g_p appartient à la sphère d'extrémités f_p et g_q, l'angle $\alpha = \widehat{f_p g_p g_q}$ doit être obtus. Par symétrie, c'est aussi le cas pour l'angle $\beta = \widehat{f_q g_q g_p}$. Les deux situations où cela arrive sont illustrées dans les figures 5.10-(a) et (d). Pour le cas de la figure 5.10-(a), le cas limite, qui donne une $d(g_p, g_q)$ maximale, c'est quand $\alpha = \beta = 90^o$ (figure 5.10-(b)). Si la droite reliant g_p et g_q est parallèle à celle reliant f_p et f_q, on a $d(g_p, g_q) = d(f_p, f_q)$. Si ce n'est pas le cas, la droite entre f_p et f_q est l'hypoténuse d'un triangle rectangle où l'un des côtés a une longueur égale à $d(g_p, g_q)$ (figure 5.10-(c)). Par conséquent, on a toujours $d(g_p, g_q) \leq d(f_p, f_q)$. Pour le cas de la figure 5.10-(d), quand $\alpha = \beta = 180^o$, on a $d(g_p, g_q) = d(f_p, f_q)$. L'autre cas extrême est quand $\alpha = \beta = 90^o$ (figure 5.10-(e)), et dans ce cas on a deux triangles rectangles dont l'addition des hypoténuses est $d(f_p, f_q)$ et dont la somme des côtés est $d(g_p, g_q)$, ce qui implique que $d(g_p, g_q) \leq d(f_p, f_q)$.



(a) Situation de nivellement (b) $d(g_p, g_q) = d(f_p, f_q)$ (c) $d(g_p, g_q) \leq d(f_p, f_q)$

(d) Situation de nivellement (e) $d(g_p, g_q) \leq d(f_p, f_q)$

FIG. 5.10 – Démonstration de la proposition 5.2.

Les propositions 5.1 et 5.2 correspondent aux conditions 5.28 et 5.30 de la définition 5.13. Cependant, la définition 5.14 n'implique pas la condition restante : $d(g_p, f_q) \leq d(f_p, f_q)$ et $d(g_q, f_p) \leq d(f_p, f_q)$. En effet, le point g_p peut être très éloigné de la référence pour la définition 5.14, en particulier dans le cas de la figure 5.10-(b) ($\alpha = \beta = 90^o$ et droites $\overrightarrow{g_p g_q}$ et $\overrightarrow{f_p f_q}$ parallèles), où les points g_p et g_q peuvent être aussi éloignés de f_p et f_q qu'on veut. La définition 5.14 est donc moins restrictive que la définition 5.13.

5.4.4 Solution retenue

Nous venons de présenter deux généralisations possibles des nivellements scalaires aux fonctions vectorielles, mais nous n'avons pas pour l'instant parlé de méthodes de construction. Cependant, pour que les nivellements vectoriels soient utilisables en pratique, il faut trouver des méthodes de construction efficaces. Nous avons vu également que la première définition est plus restrictive que la deuxième. Nous n'avons pas réussi à trouver des méthodes de construction correspondant à la définition 5.13 produisant une simplification suffisante de l'image pour des applications de segmentation. En effet, nous avons toujours obtenu des résultats trop proches de la référence pour être utiles. C'est pourquoi nous avons décidé de retenir la définition 5.14 comme définition des nivellements vectoriels, pour laquelle nous avons pu trouver des algorithmes intéressants. Toutes les propriétés et algorithmes que nous présentons par la suite sont basés sur cette définition.

5.4.5 Algorithmes

Nous proposons maintenant deux algorithmes correspondant à la solution retenue (définition 5.14). Ces algorithmes nous donnent une étape de base, qu'il faut itérer jusqu'à idempotence pour obtenir le nivellement.

Algorithme par projection Pour tout couple de points (p, q) voisins, si $g_p \notin \text{Disque}(f_p, g_q)$, projeter g_p sur le $\text{Disque}(f_p, g_q)$, comme le montre la figure 5.11.

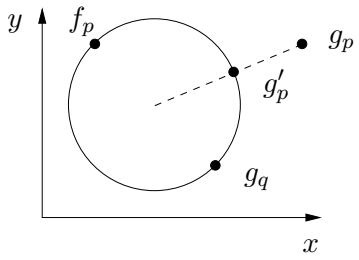


FIG. 5.11 – Algorithme par projection.

Algorithme par intersection Pour tout couple de points (p, q) voisins, si $g_p \notin \text{Disque}(f_p, g_q)$, trouver le point le plus éloigné de f_p sur l'intersection de $\text{Disque}(f_p, g_q)$ et $\text{Disque}(f_p, g_p)$ (figure 5.12)

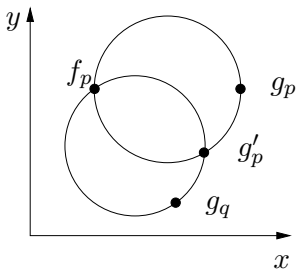


FIG. 5.12 – Algorithme par intersection.

Remarquons que l'algorithme par intersection produit des résultats qui sont plus proches de la fonction marqueur que ceux de l'algorithme par projection, ce qui résulte en des filtrages plus importants. Notons aussi le lien entre cet algorithme et la définition des nivellements de la section 5.3.3 dans le cas des Disques. En effet, pour les nivellements basés sur la relation d'ordre, l'étape de nivellement se fait en calculant l'inf, qui consiste à trouver le point le plus éloigné sur l'intersection des sphères ayant comme diamètre le point f_p et l'un des g_q voisins. Le calcul de l'intersection de plusieurs sphères est plutôt compliqué. Par contre, si on prend les sphères deux à deux, le calcul est beaucoup plus simple. Ici, nous trouvons le point le plus éloigné, non sur l'intersection de toutes les sphères, mais seulement sur l'intersection de deux sphères, correspondant au point g_p et un de ses voisins g_q. Cet algorithme est donc similaire à celui proposé dans la section 5.3.3, mais dans notre cas le calcul est beaucoup plus simple.

5.4.5.1 Calculs détaillés pour l'algorithme par intersection

Pour l'algorithme par intersection, le point résultant se trouve toujours sur le plan défini par f_p, g_p et g_q. Nous nous centrons donc sur ce plan, et regardons en détail le calcul de cet algorithme. Pour cela, nous nous basons sur la figure 5.13 :

- si $g_p \in Disque(f_p, g_q)$, on a $g'_p = g_p$;
- si $g_q \in Disque(f_p, g_p)$, on a $g'_p = g_q$;
- autrement, nous avons le cas de la figure 5.13, et le point g'_p est calculé par des raisonnements géométriques.

  D'abord, le point g'_p peut s'exprimer comme la somme vectorielle du vecteur $\overrightarrow{f_p g_p}$ et une portion du vecteur $\overrightarrow{g_p g_q}$:

$$\overrightarrow{f_p g'_p} = \overrightarrow{f_p g_p} + \nu \cdot \left(\overrightarrow{f_p g_q} - \overrightarrow{f_p g_p}\right) = \nu \cdot \overrightarrow{f_p g_q} + (1-\nu) \cdot \overrightarrow{f_p g_p} \qquad 0 \leq \nu \leq 1 \tag{5.31}$$

  D'autre part, les vecteurs $\overrightarrow{f_p g'_p}$ et $\overrightarrow{g_p g_q}$ doivent être orthogonaux :

$$\overrightarrow{f_p g'_p} \cdot \left(\overrightarrow{f_p g_p} - \overrightarrow{f_p g_q}\right) = 0$$

  ce qui équivaut à :

$$\overrightarrow{f_p g'_p} \cdot \overrightarrow{f_p g_q} = \overrightarrow{f_p g'_p} \cdot \overrightarrow{f_p g_p} \tag{5.32}$$

  En remplaçant $\overrightarrow{f_p g'_p}$ par sa valeur dans 5.32 et en isolant ν, on obtient :

$$\nu = \frac{\overrightarrow{f_p g_p} \cdot \left(\overrightarrow{f_p g_p} - \overrightarrow{f_p g_q}\right)}{\left(\overrightarrow{f_p g_q} - \overrightarrow{f_p g_p}\right)^2} \tag{5.33}$$

  En remplaçant cette valeur dans l'équation 5.31, on obtient la formule suivante pour le calcul du vecteur $\overrightarrow{f_p g'_p}$:

$$\overrightarrow{f_p g'_p} = \frac{\overrightarrow{g_p f_p} \cdot \overrightarrow{g_p g_q}}{|\overrightarrow{g_q g_p}|^2} \cdot \overrightarrow{f_p g_q} + \frac{\overrightarrow{g_q f_p} \cdot \overrightarrow{g_q g_p}}{|\overrightarrow{g_q g_p}|^2} \cdot \overrightarrow{f_p g_p} \tag{5.34}$$

  Et le point g'_p est obtenu comme $g'_p = f_p + \overrightarrow{f_p g'_p}$.

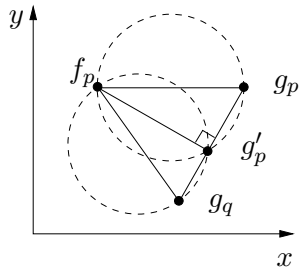


FIG. 5.13 – Calculs détaillés pour l'algorithme par intersection.

5.4.5.2 Adaptation aux images d'entiers

Les deux algorithmes précédents nécessitent de calculs en point flottant. Cependant, on travaille souvent sur des images d'entiers, et il est souhaitable d'avoir des images de sortie qui soient aussi définies sur les entiers. La question se pose alors sur la meilleure manière d'arrondir les valeurs résultantes des calculs en point flottant. On a quelques contraintes :

- l'arrondi ne doit pas éloigner le point de la référence, afin d'assurer la convergence de l'algorithme ;
- l'arrondi ne doit pas éloigner le point du centre du Disque(f_p, g_q). Si cette condition n'est pas vérifiée, à la prochaine itération la condition de nivellement ne sera pas vérifiée et on aura des oscillations.

Ces deux conditions se résument en une seule : l'arrondi doit être à l'intérieur du $Disque(f_p, g_q)$.

Avec cette contrainte, on cherche le meilleur arrondi du vecteur g'_q de composantes exprimées en point flottant. Pour un vecteur de trois composantes, on a 2^3 possibles arrondis. Il faudra choisir, parmi ceux à l'intérieur du disque, celui qui est le plus proche de la vraie valeur de g'_q. Toutefois, il est possible qu'il n'existe pas de vecteur entier à l'intérieur du disque. Cela arrive lorsque les points f_p et g_q sont très proches. Dans ce cas, on arrondit à la valeur du vecteur de référence f_p. La figure 5.14 montre un exemple pour des points en deux dimensions. Parmi les quatre arrondis possibles pour g_q, seulement trois sont dans le disque. Parmi eux, le plus proche de la vraie valeur g_q est choisi.

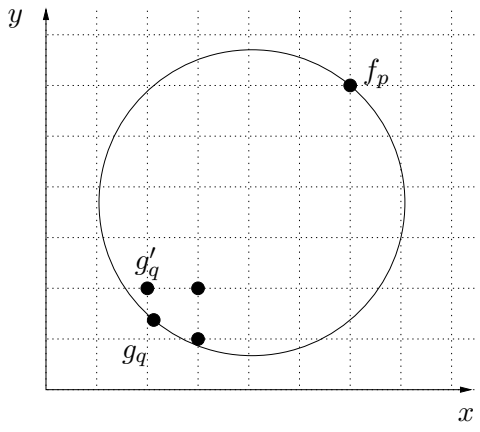


FIG. 5.14 – Arrondi des valeurs pour les images d'entiers.

Notons que la valeur choisie pour l'arrondi satisfait toujours à la condition de nivellement et qu'elle est plus proche de la référence que la valeur réelle. Cette méthode donne de très bonnes propriétés de convergence de l'algorithme en réduisant le nombre d'itérations nécessaires pour atteindre la convergence.

5.4.5.3 Propriétés

Connexité Nous avons vu par la proposition 5.2 que $d(g_p, g_q) \leq d(f_p, f_q)$. En conséquence, $d(f_p, f_q) = 0 \Rightarrow d(g_p, g_q) = 0$. Ainsi, ces nivellements sont des opérateurs connexes.

Idempotence Une fois la définition 5.14 satisfaite pour toute paire de points voisins, aucun changement n'a plus lieu si la transformation élémentaire est itérée. Il s'agit donc d'une transformation idempotente.

5.4.6 Résultats

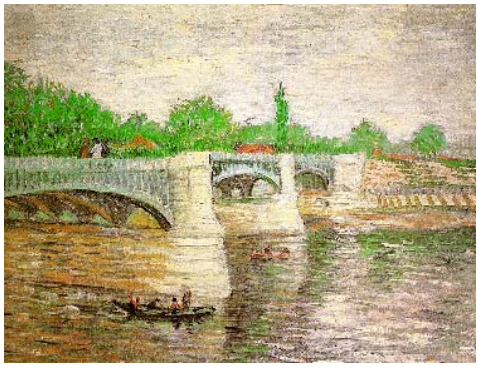
Nous présentons ici des résultats correspondant à notre nouvelle définition de nivellement couleur et ce pour les deux algorithmes présentés. Le marqueur utilisé est un filtre médian vectoriel [1] calculé sur un noyau de taille 3 itéré 3 fois. La figure 5.15 montre les résultats obtenus sur une reproduction du tableau de Van Gogh « Le Pont de la Grand Jatte » avec les algorithmes par projection et intersection, avec $\lambda = 0$ et $\lambda = 5$. Les mêmes résultats sont illustrés dans la figure 5.16 pour l'image « Lena ». On constate que, pour $\lambda = 0$, l'algorithme par projection donne des résultats très proches de la référence, tandis que l'algorithme par intersection produit des simplifications plus importantes. Par ailleurs, la simplification est plus forte pour $\lambda = 5$. La simplification causée par la forte valeur de λ réduit les différences entre les algorithmes par projection et par intersection.

5.4.6.1 Choix du marqueur

Le degré de simplification obtenu par le nivellement dépend de la valeur du paramètre λ, ainsi que du choix du marqueur. Un marqueur qui s'écarte de la référence produit un filtrage plus fort qu'un marqueur qui ressemble à l'image de départ. Toutefois, il est difficile de quantifier le degré de simplification qui sera obtenu avec un marqueur donné. N'importe quelle image peut être utilisée comme marqueur. Des versions filtrées de l'image de départ donnent de bons résultats pour la segmentation. On peut citer par exemple les filtres alternés séquentiels et les filtres passe-bas appliqués sur chaque composante couleur, ou le filtre médian vectoriel. La figure 5.17 montre les différents degrés de simplification obtenus par l'algorithme par intersection en prenant comme marqueur un filtre alterné séquentiel de taille 3 et 10 respectivement. Remarquons que bien que la simplification soit plus importante pour le marqueur de taille 10, cette simplification n'est pas en proportion avec le degré de simplicité du marqueur de taille 10 par rapport au marqueur de taille 3.

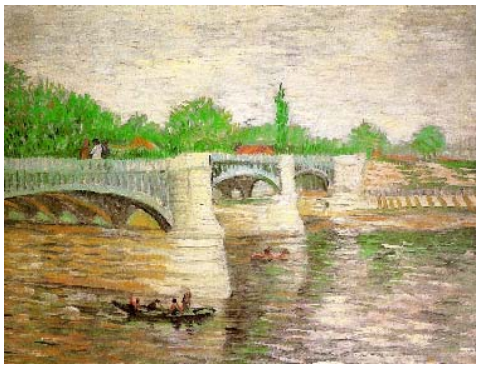
5.4.6.2 Quelques exemples avec des marqueurs particuliers

Nous montrons ici quelques résultats obtenus avec des marqueurs particuliers. La figure 5.18 illustre le résultat obtenu par l'algorithme par intersection en prenant un seul point comme marqueur, de la même couleur que sur l'image originale. Le calcul a été effectué sur l'espace RGB. Nous observons que le résultat est dans la même teinte que le marqueur, avec

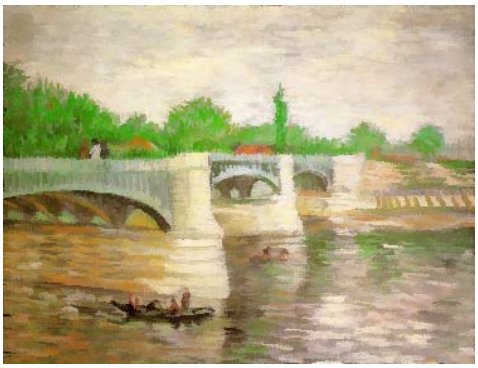


(a) Image originale

(b) Image marqueur



(c) Algorithme par projection ($\lambda = 0$)



(d) Algorithme par intersection ($\lambda = 0$)

(e) Algorithme par projection ($\lambda = 5$)

(f) Algorithme par intersection ($\lambda = 5$)

FIG. 5.15 – Résultats avec les algorithmes par projection et intersection sur une reproduction du tableau de Van Gogh « Le Pont de la Grande Jatte ».

(a) Image originale (b) Image marqueur

(c) Alg. par projection ($\lambda = 0$) (d) Alg. par intersection ($\lambda = 0$)

(e) Alg. par projection ($\lambda = 5$) (f) Alg. par intersection ($\lambda = 5$)

FIG. 5.16 – Résultats avec les algorithmes par projection et intersection pour l'image « Lena ».

(a) Image originale

(b) Filtre alterné séquentiel de taille 3

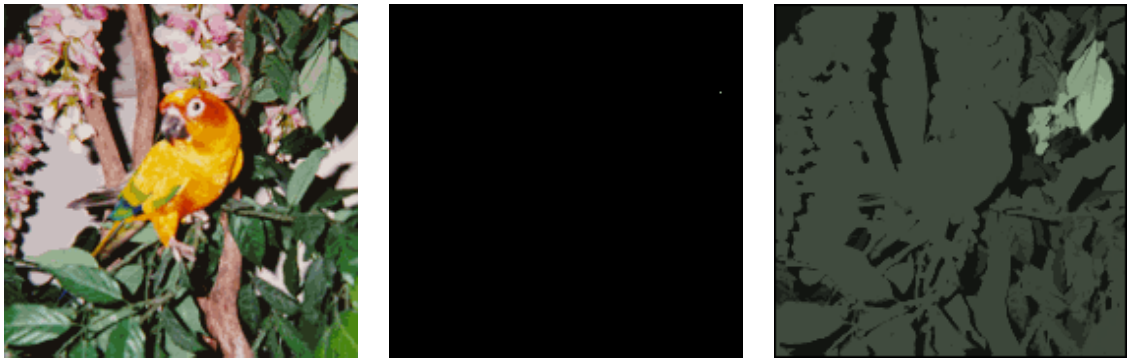
(c) Nivellement couleur

(d) Filtre alterné séquentiel de taille 10

(e) Nivellement couleur

FIG. 5.17 – Résultats du nivellement couleur pour l'algorithme par intersection avec deux marqueurs différents.

des luminances différentes. Ce résultat est une conséquence de la construction de l'algorithme. En effet, le résultat g'_p du nivellement se trouve toujours sur la droite définie par les points g_p et g_q (figure 5.13). En conséquence, si tous les point de l'image marqueur se trouvent sur une même droite, comme c'est le cas de l'exemple que nous venons de présenter, tous les points du nivellement appartiendront à cette même droite. On trouve un cas particulièrement intéressant quand le marqueur est une image à niveaux de gris. Dans ce cas, tous les points du marqueur se trouvent sur la droite $R = G = B$ (pour les images RGB) et le résultat est aussi une image à niveaux de gris. Bien entendu, ce résultat ne se produirait pas si on effectuait les calculs sur un espace couleur où les valeurs de gris ne se trouvaient pas sur une droite. Notons que cette propriété n'est plus vérifiée lorsque la méthode de calcul sur les entiers proposée dans la section 5.4.5.2 est appliquée, les erreurs d'arrondi provoquant un déplacement des points en dehors de la droite des valeurs de gris. Elle n'est pas non plus vérifiée par l'algorithme par projection. La figure 5.19 illustre des résultats avec un marqueur à niveaux de gris, pour les algorithmes par projection et par intersection. Ces résultats ont été obtenus sans appliquer l'algorithme d'arrondi de la section 5.4.5.2. C'est-à-dire que les calculs ont été effectués en point flottant et l'arrondi n'a été appliqué qu'à la fin de l'algorithme. On observe que le résultat se trouve sur la droite des valeurs de gris pour l'algorithme par intersection, mais ce n'est pas le cas pour l'algorithme par projection.



(a) Image originale (b) Marqueur (c) Nivellement

FIG. 5.18 – Nivellement en prenant un seul point comme marqueur, pour l'image « Bird ».

(a) Marqueur (b) Algorithme par projection (c) Algorithme par intersection

FIG. 5.19 – Nivellement couleur en prenant un marqueur à niveaux de gris.

5.5 Les zones plates pour les images à niveaux de gris

Nous analysons maintenant l'importance qu'ont les zones plates dans certains systèmes de segmentation. Nous présentons ensuite quelques généralisations de la notion de zone plate permettant d'obtenir des zones quasi-plates mieux adaptées aux objets existants dans l'image.

5.5.1 Zones plates et segmentation

Des sous ensembles des zones plates ont été utilisés comme marqueurs en segmentation morphologique. Dans [14], ce sous ensemble est sélectionné selon des critères de contraste et de surface, constituant un ensemble de marqueurs. Les zones plates non sélectionnées sont dans un deuxième temps fusionnées avec les zones plates marqueurs afin d'obtenir la segmentation finale. Dans [43], une partition fine est obtenue en choisissant les zones plates au dessus d'un seuil de surface. Ces zones plates constituent les marqueurs d'un algorithme de croissance de régions aboutissant à la production d'une partition fine. Dans le même esprit, la méthode de croissance de régions peut être remplacée par une LPE, comme c'est le cas dans [75].

Dans toutes ces approches, le problème est de trouver les zones plates les plus significatives, et le choix n'est pas facile. En effet, dans une image à niveaux de gris il y a, dans la plupart des cas, beaucoup trop de zones plates. Bien qu'idéalement un objet visuellement uniforme devrait être constitué d'une seule et grande zone plate, en réalité il est composé d'un grand nombre de zones plates ayant des niveaux de gris très proches. Essayer de trouver les plus significatives par des critères de surface n'est pas toujours concluant, et les critères de contraste peuvent négliger des grands objets à contraste moyen. Les filtres connexes peuvent constituer une solution à ce problème, car ils ont un effet d'élargissement des zones plates. Ainsi, on peut espérer que le filtrage fusionnera les zones plates les plus similaires et leur taille commencera à être plus discriminante. Mais pour obtenir des zones plates suffisamment grandes il est nécessaire d'appliquer des filtres de grande taille, ce qui d'un côté fait augmenter le temps de calcul de l'algorithme et de l'autre risque de trop simplifier l'image et d'éliminer des détails importants.

5.5.2 Les zones quasi-plates

Les zones quasi-plates permettent de remédier au problème posé par la sélection des zones plates en élargissant leur définition. Ainsi, au lieu de considérer des régions à niveau de gris constant, des variations du niveau de gris selon des modèles précis sont permises dans une même zone quasi-plate. Dans [84], les zones quasi-plates sont détectées moyennant des seuils du gradient. Le gradient exprimant des différences de niveau de gris entre pixels voisins, un seuil sur le gradient permet de détecter les régions de l'image où la variation entre niveaux de gris de pixels voisins est petite. Dans [74], sont considérées comme zones quasi-plates les régions connexes de l'image où la variation du niveau de gris entre pixels voisins est plus petite ou égale à 1. Une contrainte additionnelle est cependant imposée sur le processus d'étiquetage, qui veut que la différence entre le niveau de gris d'un nouveau pixel ajouté à la zone plate et la moyenne du niveau de gris des pixels appartenant déjà à la zone plate ne peut pas être supérieur à 10. Notons que cette contrainte dépend de l'ordre d'étiquetage des pixels.

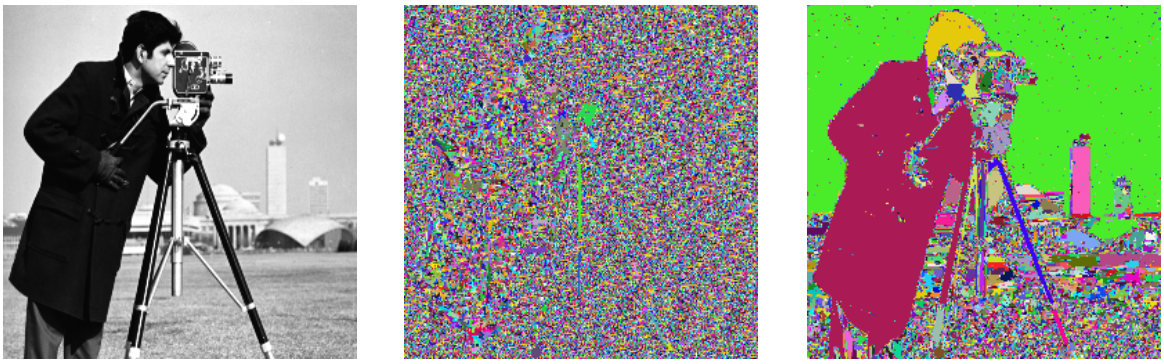
Définition 5.15 *Deux points $x, y \in E$ appartiennent à la même* **zone quasi-plate** *d'une fonction f si et seulement si il existe une chaîne de points $(p_1, p_2, \ldots, p_n)$ telle que $p_1 = x$ et $p_n = y$ et, pour tout i, $(p_i, p_{i+1}) \in E$ sont voisins et $f_{p_i} \approx f_{p_{i+1}}$.*

Des zones quasi-plates avec des caractéristiques différentes sont obtenues selon la définition de la relation $\approx$.

5.5.2.1 Zones quasi-plates associées aux nivellements

Les nivellements étendus, que nous avons présentés dans la section 5.2.3, introduisent les zones quasi-plates moyennant la re-définition de la relation binaire entre pixels voisins. Nous en avons vu un exemple avec les zones λ-plates, où la nouvelle relation est définie comme $g_p \succ g_q \Leftrightarrow g_p > g_q + \lambda$. La relation $\approx$ dans la définition 5.15 devient $f_p \approx f_q \Leftrightarrow f_q - \lambda < f_p < f_q + \lambda$.

La figure 5.20 compare les zones plates obtenues selon la définition classique (zones plates « strictes ») et les zones λ-plates. Les zones λ-plates étant mieux adaptées au contenu de l'image, la sélection des plus représentatives est très simplifiée.



(a) Image originale (b) Zones plates « strictes » ($\lambda = 0$) (c) Zones quasi-plates avec $\lambda = 4$

FIG. 5.20 – Zones λ-plates de l'image « Cameraman » pour $\lambda = 0$ et $\lambda = 4$.

D'autres relations binaires sont proposées dans [55].

5.6 Travaux antérieurs sur les zones plates couleur

Crespo et Schafer [13] ont proposé l'extension du concept de segmentation basée sur les zones plates aux images couleur. Dans leur approche, les zones plates sont détectées indépendamment sur chacune des composantes couleur. On obtient de cette façon trois images I_r, I_g et I_b, sur lesquelles on choisit les zones plates les plus importantes, selon des critères de surface et de gradient. Les parties de l'image appartenant aux zones plates non sélectionnées sont mises à zéro. Ensuite, les zones plates couleur sont définies comme l'intersection des images I_r, I_g et I_b, les zones à zéro donnant zéro sur l'intersection. L'image résultante est utilisée comme image marqueur pour la segmentation, chacune des zones plates donnant lieu à une région dans la segmentation finale moyennant une procédure de fusion de régions sur une des bandes de l'image. Cette approche a l'inconvénient de produire le plus souvent des zones plates trop petites. En effet, une zone plate ne peut être marqueur que si tous ses pixels ont la même valeur sur les trois composantes couleur en même temps, et que si en plus elle satisfait à certains critères de surface et contraste pour chacune des composantes. Avec cette procédure, il est facile de négliger des régions perceptuellement uniformes qui ont de très légères variations sur les valeurs de leurs composantes. Si ces régions ne contiennent pas des

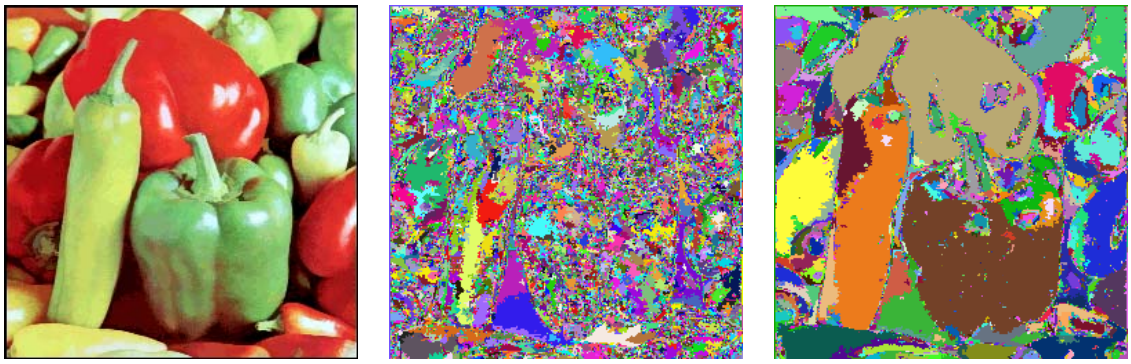
parties complètement homogènes en couleur suffisamment grandes, il n'y aura pas de marqueur correspondant à cette région et la procédure de fusion de régions l'attribuera à l'un des marqueurs voisins. Des pré-filtrages importants peuvent réduire le problème, en élargissant les zones plates, mais un filtrage trop fort peut amener à l'élimination d'informations importantes dans l'image. D'ailleurs, la procédure de fusion n'ayant lieu que sur une des bandes de l'image, l'information couleur n'est pas traitée de manière optimale. Cet inconvénient peut être résolu en appliquant une méthode de croissance de régions basée sur une distance couleur [52]. Toutefois, cette méthode permet difficilement de trouver les régions perceptuellement homogènes les plus importantes dans l'image.

5.7 Les zones quasi-plates couleur

Dans le même esprit que pour les images à niveaux de gris, il est possible d'élargir la notion de zone plate. Cette approche exprime l'idée que de petites variations de la couleur d'une région ne sont pas perçues visuellement. On peut alors considérer ces régions comme étant plates, ou quasi-plates. Ici, nous retiendrons la définition la plus simple vue pour les images à niveaux de gris : les λ-zones plates, que nous alors étendre aux images couleur.

Définition 5.16 *Étant donnée une distance $s : \mathbb{R}^n \to \mathbb{R}$, deux points $x, y \in E$ appartiennent à la même* **λ-zone plate** *d'une fonction vectorielle f si et seulement s'il existe une chaîne de points $(p_1, p_2, \ldots, p_n)$ telle que $p_1 = x$ et $p_n = y$ et, pour tout i, $(p_i, p_{i+1}) \in E$ sont voisins et $d\left(f_{p_{i+1}}, f_{p_i}\right) \leq \lambda$.*

La figure 5.21 illustre les zones plates couleur avec $\lambda = 0$ (zones plates strictes) et $\lambda = 12$ pour l'image « Peppers ». L'image originale a été légèrement filtrée par nivellement afin d'éliminer des détails trop fins ainsi qu'une partie du bruit. La distance d utilisée est la distance euclidienne sur l'espace couleur RGB. Notons que les zones plates obtenues avec $\lambda = 12$ sont beaucoup plus représentatives du contenu de l'image que dans le cas $\lambda = 0$. La sélection des plus importantes peut être faite à partir d'un critère de taille. Notons également la façon dont les zones plates sont distribuées. Les grandes zones plates occupent les zones homogènes de l'image, tandis que de très petites régions s'accumulent sur les régions de transition qui entourent les objets. C'est dans ces régions de transition que se trouvent les vrais contours de l'objet, que l'on cherche à trouver moyennant la segmentation.



(a) Image filtrée (b) Zones plates couleur, $\lambda = 0$ (c) Zones plates couleur, $\lambda = 12$

FIG. 5.21 – Zones λ-plates couleur pour $\lambda = 0$ et $\lambda = 12$.

5.7.1 Choix de l'espace couleur

La définition des zones plates couleur implique le choix préalable d'un espace couleur dans lequel effectuer le calcul de distance. Un espace couleur est dit *uniforme* si la proximité dans l'espace couleur correspond à une proximité perceptuelle. L'espace RGB ainsi que ses transformations linéaires, tels que YUV et YCrCb ne sont pas uniformes. En conséquence, cela n'a pas de sens, théoriquement, d'utiliser une mesure de distance - telle que la distance euclidienne - dans ces espaces. Certains espaces couleur, tels que Lab et HSV, sont uniformes et en conséquence des mesures de distance dans ces espaces mesurent des dissimilarités de couleur.

Dans l'espace Lab la dissimilarité entre deux couleurs $c_1 = (L_1, a_1, b_1)$ et $c_2 = (L_2, a_2, b_2)$ est mesurée avec la distance euclidienne :

$$d(c_1, c_2) = \sqrt{(L_1 - L_2)^2 + (a_1 - a_2)^2 + (b_1 - b_2)^2} \tag{5.35}$$

Dans l'espace cylindrique HSV, la dissimilarité entre deux couleurs $c_1 = (h_1, s_1, v_1)$ et $c_2 = (h_2, s_2, v_2)$ est donnée par la distance euclidienne dans cet espace :

$$d(c_1, c_2) = \frac{1}{\sqrt{5}} \cdot \sqrt{(v_1 - v_2)^2 + (s_1 \cos h_1 - s_2 \cos h_2)^2 + (s_1 \sin h_1 - s_2 \sin h_2)^2} \tag{5.36}$$

On pourrait espérer que les zones λ-plates de meilleure qualité soient obtenues pour les espaces uniformes. Cependant, ce n'est pas le cas pour les applications de segmentation. En effet, en segmentation on cherche à trouver les contours des objets à signification sémantique. Il est courant qu'une partie de ces contours ne soit pas très fortement contrastée avec les objets voisins. Les mesures de distance dans des espaces uniformes ont tendance à fusionner très vite les régions qui ne sont pas séparées par un contour continu et très marqué visuellement. Or ces fusions sont fatales pour la segmentation. Nous avons constaté que dans des espaces non uniformes tels que RGB ou YUV de tels contours arrivent à mieux survivre.

Nous avons comparé (figure 5.22) les zones plates obtenues pour les espaces couleur RGB, YUV, HSV et Lab. Pour les espaces RGB, YUV et Lab la distance euclidienne (équation 5.35) a été utilisée, tandis que pour l'espace HSV nous avons utilisé la distance de l'équation 5.36. L'image originale a d'abord été filtrée et normalisée sur les trois composantes. Ensuite, les zones λ-plates couleur ont été détectées dans chaque espace couleur. Afin de pouvoir comparer les résultats, les zones plates ont été détectées avec une valeur de λ telle que, une fois éliminées les zones de transition (régions en dessous d'une certaine taille, ici 5 pixels) la surface de l'image couverte par les zones plates soit d'environ 90%. On constate que les espaces RGB et YUV donnent des résultats comparables et satisfaisants du point de vue de la segmentation, mais l'espace HSV fusionne des régions qui, tout en étant semblables en couleur, ont une signification sémantique différente. L'espace Lab montre la même tendance à fusionner des régions de couleur similaire, mais en même temps dans certains cas il sépare des régions très proches en couleur, comme c'est le cas de la partie haute du fond derrière la vierge dans la figure 5.22-(f). Dans des applications de segmentation, la fusion de régions de couleur similaire n'est pas souhaitable si elles appartiennent à des objets différents. Ainsi, notre objectif étant de détecter les régions très homogènes en couleur, la similarité perceptuelle ne nous intéresse pas au dessus d'un certain seuil. Les espaces dits uniformes ont tendance à réduire les distances entre couleurs perceptuellement proches, ce qui amène à une augmentation des fuites.

En conséquence, même si les espaces HSV et Lab sont perceptuellement uniformes, nous trouvons que du point de vue de la segmentation il est plus intéressant de travailler sur RGB ou YUV.

(a) Image originale (b) Image filtrée et normalisée (c) RGB

(d) YUV (e) HSV (f) Lab

FIG. 5.22 – Zones plates couleur obtenues pour différents espaces couleur.

5.8 Zones plates associées aux nivellements

En tant que filtres connexes, les nivellements agissent en élargissant les zones plates de l'image. Les caractéristiques des zones plates couleur de l'image filtrée sont différentes pour les différents nivellements couleur.

La figure 5.24 montre les zones λ-plates correspondant aux nivellements de la figure 5.23. Afin de pouvoir comparer leur performance du point de vue de la segmentation, les valeurs de λ choisies pour chaque image sont celles pour lesquelles les zones plates, une fois éliminées les régions de transition (régions de moins de 5 pixels) couvrent environ 90% de l'image. Les zones plates résultantes sont illustrées en fausses couleurs dans la figure 5.24, les zones en noir correspondant aux zones de transition (zones plates de moins de 5 pixels). Ces mêmes zones plates, sans éliminer les régions de transition, sont montrées remplies avec leur couleur moyenne. On constate que le nivellement scalaire produit des zones plates moins adaptées au contenu en couleur de l'image. Ce phénomène s'explique du fait que l'application d'un λ-nivellement aux trois composantes de l'image est plus restrictive que si, comme pour les nivellements couleur, on travaille avec une valeur *globale* de λ. En effet, le nivellement composante par composante imposera des distances entre pixels voisins de moins de λ *pour toutes les trois composantes*, tandis que les nivellements couleur permettront qu'une des composantes s'éloigne plus de cette distance si les autres sont suffisamment proches pour compenser. En conséquence, le nivellement composante par composante, plus restrictif, produit beaucoup plus de petites zones plates en dessous du seuil de 5 pixels que les nivellements par projection et intersection. Ainsi, la valeur de λ nécessaire pour que les zones plates restantes couvrent 90% de l'image est relativement élevée, ce qui fait qu'elles sont plus larges que celles obtenues pour les autres nivellements.

5.9 Conclusion

Dans ce chapitre nous avons présenté deux aspects très importants de la segmentation morphologique : le pré-filtrage, sous forme de nivellement, et la détection des zones plates. Les travaux antérieurs portant principalement sur les images à niveaux de gris, nous nous sommes intéressés aux images en couleur.

En ce qui concerne les nivellements, nous avons présenté une nouvelle généralisation aux fonctions vectorielles qui répond aux principaux inconvénients des méthodes antérieures en donnant d'une part une définition formelle et de l'autre en présentant une technique de calcul simple.

Nous avons décrit des travaux qui utilisent la détection des zones plates pour obtenir une segmentation. Dans tous ces travaux, ce sont les zones plates strictes qui sont utilisées. Cependant, les récents travaux de Meyer sur les nivellements [54], [55], amènent à la notion de zone quasi-plate, permettant d'obtenir des zones plates mieux adaptées au contenu de l'image. Nous avons étendu et appliqué cette idée aux images couleur, pour obtenir des zones quasi-plates beaucoup mieux adaptées au contenu de l'image que dans les approches antérieures. Dans le chapitre 6 nous montrerons comment ces zones quasi-plates couleur peuvent être utilisées pour obtenir de façon très simple et rapide une partition fine de très bonne qualité, sur laquelle nous construirons notre hiérarchie.

Finalement, nous avons étudié les nivellements couleur du point de vue des zones quasi-plates couleur associées et nous avons vu que ces zones plates correspondent mieux au contenu

(a) Image originale

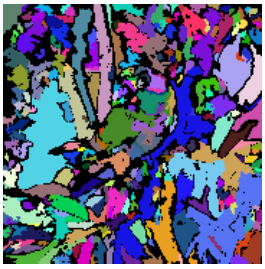
(b) Image marqueur

(c) Algorithme par projection

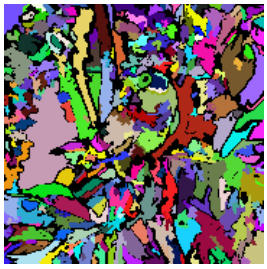
(d) Algorithme par intersection

(e) Nivellement scalaire

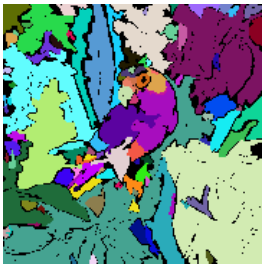
FIG. 5.23 – Des différents nivellements couleur avec $\lambda_{niv} = 5$ pour l'image « Bird ».



(a) Zones plates alg. projection



(b) Zones plates alg. intersection



(c) Zones plates nivellement scalaire

(d) Couleur moyenne alg. projection

(e) Couleur moyenne alg. intersection

(f) Couleur moyenne niv. scalaire

FIG. 5.24 – Comparaison entre les zones plates obtenues par les différents nivellements couleur pour l'image « Bird ».

en couleur de l'image que l'approche qui consiste à niveler chaque composante couleur indépendamment.

Chapitre 6

Segmentation multiéchelle d'images fixes

6.1 Introduction

Nous avons vu dans le chapitre 4 comment, à partir d'une famille de partitions emboîtées, il est possible de développer des mécanismes puissants d'interaction. Dans ce chapitre, nous envisageons le problème de la construction d'une telle hiérarchie.

On trouve dans la littérature plusieurs techniques de construction de hiérarchies. La plupart proposent des méthodes de fusion de bas en haut avec différents critères, dont nous avons parlé dans le chapitre 2. Dans la majorité des cas, les critères employés se basent uniquement sur des mesures d'homogénéité locale des régions. De tels critères fonctionnent bien jusqu'à l'échelle où les régions homogènes ont été détectées, mais pour des échelles plus grossières où l'on cherche à détecter des objets à signification sémantique – des objets vidéo dans le langage MPEG4 – la qualité de la segmentation se voit considérablement réduite. Certains auteurs [10] considèrent que, une fois les régions homogènes détectées, le seul moyen d'obtenir les objets vidéo est à travers l'introduction de sémantique, sous forme de modèles ou d'interaction. Nous pensons cependant qu'il est possible de faire mieux automatiquement avant de passer à l'étape sémantique, en utilisant des critères globaux de contraste et de taille des régions. Dans ce chapitre nous utilisons des mesures topographiques sur une image gradient permettant d'obtenir des hiérarchies de partitions emboîtées très bien adaptées au contenu des images, même pour des niveaux grossiers de la hiérarchie. Nous verrons qu'il existe des algorithmes rapides pour le calcul de ces distances, ce qui constitue un avantage important dans un système interactif.

À côté des critères de construction de la hiérarchie qui déterminent la qualité des partitions résultantes, la représentation détermine la rapidité d'accès aux différents niveaux de résolution. Une représentation très simplifiée de la hiérarchie et une organisation efficace de l'information réduiront en effet le temps de réponse de l'algorithme. Idéalement, ce temps de réponse doit être imperceptible pour l'utilisateur. En combinant des critères de hiérarchisation puissants avec une représentation très simplifiée, nous obtenons une hiérarchie très adaptée à la segmentation interactive.

Dans la section 6.2 nous introduisons des hiérarchies définies sur différents supports et qui dérivent très naturellement de certaines notions de la morphologie mathématique. Dans la section 6.3, nous nous centrons sur le support donné par les régions de la LPE (ligne

de partage des eaux), sur lequel de très différentes hiérarchies peuvent être dérivées, et nous discutons quelques représentations possibles des hiérarchies ainsi obtenues. Dans la section 6.4 nous étendons les idées de la section précédente à la construction de hiérarchies à partir d'une partition fine quelconque. Ensuite, dans la section 6.5 nous proposons des techniques permettant d'augmenter la qualité et la robustesse des hiérarchies obtenues. Dans la section 6.6 nous présentons deux manières d'introduire de l'information *a priori* dans le processus de calcul de la hiérarchie : les hiérarchies lexicographiques et les marqueurs flous. Finalement, la section 6.7 détaille la représentation choisie et la mise en place des mécanismes d'interaction qui en dérivent.

6.2 Hiérarchies sur des régions à support croissant

Certains opérateurs morphologiques donnent naturellement lieu à des familles de partitions emboîtées. Nous en présentons ici quelques exemples et en discutons l'utilité pratique.

6.2.1 Hiérarchies sur des zones plates croissantes

Nous avons vu dans le chapitre 5, en rapport avec les nivellements, qu'il est possible de définir de nouvelles notions de zone plate, que nous appelons des zones « quasi-plates ». Nous avons en particulier donné l'exemple des zones λ-plates, où un ensemble de pixels est considéré comme appartenant à la même zone λ-plate si pour toute paire de pixels il existe un chemin tel que l'écart entre deux pixels adjacents n'est pas supérieur au paramètre λ.

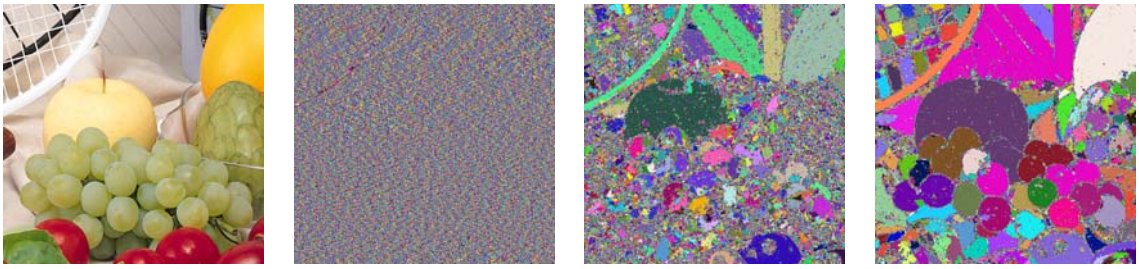
Rappelons que l'ensemble des zones plates – ou quasi-plates – forme une partition de l'image, et considérons maintenant les partitions données par l'ensemble des zones λ-plates d'une fonction pour des valeurs de λ croissants. Ces partitions sont emboîtées. En effet, le fait d'augmenter la valeur du paramètre λ ne peut qu'élargir les zones plates, en fusionnant les unes avec les autres.

Nous obtenons ainsi une hiérarchie de partitions emboîtées, dont la partition la plus fine est donnée par les zones plates strictes ($\lambda = 0$) de la fonction de départ. La mesure de dissimilarité entre pixels est leur distance couleur. L'ordre de fusion des régions est défini par les sauts minimaux entre les valeurs de pixels voisins. L'ultramétrique associée à cette hiérarchie est donc l'ultramétrique sous-dominante.

La figure 6.1 illustre trois niveaux d'une telle hiérarchie. Pour $\lambda = 0$, l'image est pratiquement décomposée en pixels. Au fur et à mesure que λ augmente, les régions homogènes s'élargissent. Cependant, et même pour des valeurs de λ assez grands, il reste toujours une multitude de petites régions localisées à l'intérieur des zones de transition de l'image.

6.2.2 Hiérarchies de nivellements

Les nivellements sont des filtres connexes, c'est-à-dire, les zones plates – ou quasi-plates – de la fonction de départ sont préservées ou fusionnées avec ses voisines, mais jamais découpées. Considérons maintenant une séquence de nivellements où le résultat du nivellement précédent est utilisé comme référence du nivellement suivant. On a ainsi une famille de fonctions Λ_n telle



(a) Image originale (b) $\lambda = 0$ (c) $\lambda = 5$ (d) $\lambda = 8$

FIG. 6.1 – Partitions emboîtées obtenues par détection des zones λ-plates, pour des λ croissants. La distance utilisée est la distance euclidienne dans l'espace RGB.

que :

$$\begin{aligned} \Lambda_0 &= f \\ \Lambda_1 &= \Phi_{g_1}(\Lambda_0) \\ \Lambda_2 &= \Phi_{g_2}(\Lambda_1) \\ &\vdots \\ \Lambda_n &= \Phi_{g_n}(\Lambda_{n-1}) \end{aligned} \tag{6.1}$$

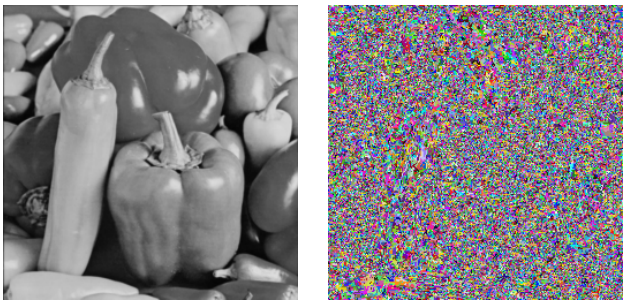
où $\Phi_g(f)$ représente le nivellement de la fonction f en prenant la fonction g comme marqueur.

Puisque les nivellements sont des filtres connexes, les zones plates des fonctions Λ_n augmentent de taille avec n. Ainsi, les zones plates des nivellements Λ_n forment une famille de partitions emboîtées. Le degré d'élargissement des zones plates est contrôlé par le marqueur g_n. Afin d'obtenir des simplifications intéressantes, la famille de marqueurs g_n doit être composée de fonctions de plus en plus éloignées de la fonction de départ f. Dans la figure 6.2, nous illustrons quelques partitions obtenues en appliquant cette méthode à une image à niveaux de gris. La méthode reste valable pour des images couleur en utilisant les nivellements couleur décrits dans le chapitre 5. La famille de marqueurs utilisée correspond à des filtres par ouverture-fermeture de taille croissante (des incréments de taille de 5 pixels à chaque itération). Les régions de la partition correspondent aux zones plates strictes du nivellement. Après la $5^{\text{ème}}$ itération, le nombre de régions s'est réduit de 44898 sur l'image de départ à 12852. Notons cependant qu'il reste de nombreuses petites régions sans aucune signification sémantique.

Avant la construction de la hiérarchie, la définition *a priori* de la distance ultramétrique qui résulte en cette hiérarchie n'est pas évidente. Mais une fois la hiérarchie construite, et comme pour toute hiérarchie, ont peut définir *a posteriori* la distance ultramétrique entre paires de pixels comme la valeur de n pour laquelle les deux pixels se retrouvent dans la même zone plate pour la première fois.

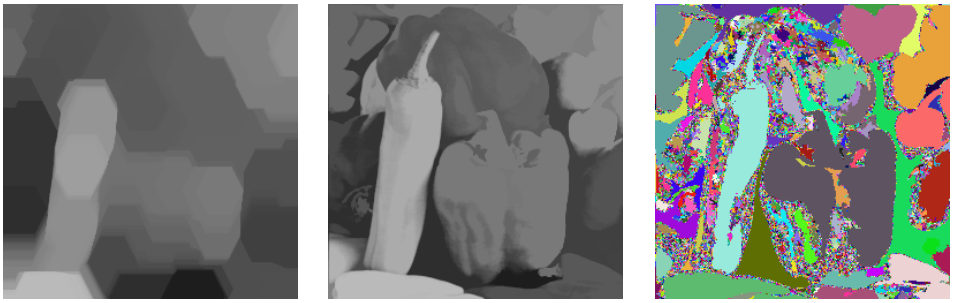
6.2.3 Hiérarchie donnée par la LPE

La Ligne de Partage des Eaux est très utilisée en segmentation morphologique. Les lecteurs n'étant pas familiarisés avec cette transformation morphologique trouveront une introduction dans l'annexe A. La mise en œuvre traditionnelle de cette méthode consiste à choisir des



(a) Image originale, Λ_0 (b) Zones plates de Λ_0

(c) Marqueur g_1 (d) Nivellement Λ_1 (e) Zones plates de Λ_1



(f) Marqueur g_5 (g) Nivellement Λ_5 (h) Zones plates de Λ_5

FIG. 6.2 – Partitions emboîtées obtenues par nivellements successifs.

marqueurs contenus dans les régions d'intérêt. La LPE est utilisée sur l'image gradient pour trouver les meilleurs contours séparant les marqueurs. Dans cette approche, seul le résultat obtenu à la fin de l'algorithme est utilisé. Cependant, les simulations d'inondation permettent d'extraire d'autres informations utiles pour la construction d'une hiérarchie.

Dans ce qui suit, et sauf indication contraire, nous parlerons toujours de la LPE *sans marqueurs*, c'est-à-dire, la LPE à partir de tous les minima de l'image *gradient*.

6.2.3.1 Minima, bassins versants et régions de la LPE

La mise en place sur la grille numérique de la LPE résulte en des frontières fermées d'un pixel d'épaisseur définissant les différentes régions de la segmentation. Dans notre cas, nous cherchons des partitions de l'image, où chacun des pixels de l'image appartient à une région. La ligne de frontière de la LPE nous gène, car les pixels appartenant à une frontière ne sont attribués à aucune région. Ainsi, et tel que nous l'expliquons dans l'annexe A, dans l'algorithme utilisé les frontières sont éliminées en les assignant à une des régions frontalières. Les différentes régions sont identifiées par une étiquette, et nous les appelerons « régions de la LPE », même si cela constitue un abus de langage. Ainsi, quand nous parlons de régions de la LPE, nous signifions les régions de la partition qui résultent de cet algorithme, et qui sont identifiées par leur étiquette unique sur la partition.

Il y a une équivalence entre minima, bassins versants de l'image et régions de la LPE. En effet, chaque minimum local de l'image est contenu dans un bassin versant, et inversement chaque bassin versant contient forcément un minimum. Pendant l'inondation, un lac se forme à l'intérieur de chaque bassin versant, qui donnera lieu à une région de la LPE. Les régions de la LPE correspondent exactement aux bassins versants de l'image. Ainsi, nous parlerons indistinctement de hiérarchie de minima, de bassins versants ou de régions de la LPE.

6.2.3.2 L'ultramétrique d'inondation

Considérons maintenant l'inondation pas à pas. Lorsque l'inondation commence, aucun bassin versant n'est complètement rempli. Ensuite, des lacs se forment à l'intérieur des bassins versants, s'étalant au fur et à mesure que l'eau monte. Quand l'eau atteint certains niveaux d'inondation, des lacs appartenant à des bassins versants différents se rejoignent. Dans la LPE classique, une digue est construite à ce moment pour les séparer. Cependant, considérons ici que nous permettons à de tels lacs de fusionner. Ainsi, le nombre de lacs diminue au fur et à mesure que le niveau de l'eau monte jusqu'à ce que, à la fin de l'inondation, un seul lac recouvre toute l'image.

On peut définir une distance ultramétrique entre bassins versants comme la hauteur minimale de l'eau pour laquelle leur lacs respectifs ont fusionné. Vérifions qu'il s'agit en effet d'une distance ultramétrique :

1. $d(x,x) = 0$. Évident, puisqu'un lac est toujours fusionné avec lui même.
2. $d(x,y) = d(y,x)$. En effet, cette distance est symétrique, puisque la hauteur de l'eau est la même du point de vue des deux bassins versants.
3. $d(x,y) \leq Max\,(d(x,z), d(z,y))$. Cette condition est vérifiée puisque le terme à droite indique la hauteur de l'eau pour laquelle *les trois* lacs x, y et z ont fusionné, et elle doit être forcément plus grande ou égale à la hauteur pour laquelle seulement deux de ces lacs x et y ont fusionné, représentée par le terme à gauche.

Cette notion est illustrée dans la figure 6.3. Les bassins versants A et B se trouvent à une distance h_1 entre eux. La distance entre les bassins versants B et C est h_2. La distance entre les bassins versants A et C est la hauteur de l'eau pour laquelle leurs lacs respectifs auront fusionné, c'est-à-dire, h_2. Remarquons que $h_2 = d(A,C) = Max\,(d(A,B), d(B,C))$.

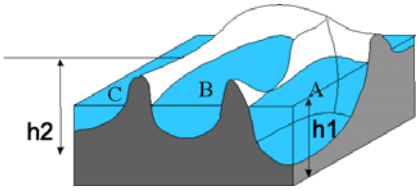


FIG. 6.3 – Ultramétrique d'inondation.

6.2.3.3 La hiérarchie associée

Nous avons vu dans le chapitre 3 qu'il est équivalent de se donner une famille de partitions emboîtées ou une distance ultramétrique. Les régions des partitions associées aux différents niveaux de la hiérarchie correspondent aux boules maximales de l'ultramétrique pour des valeurs croissantes du rayon.

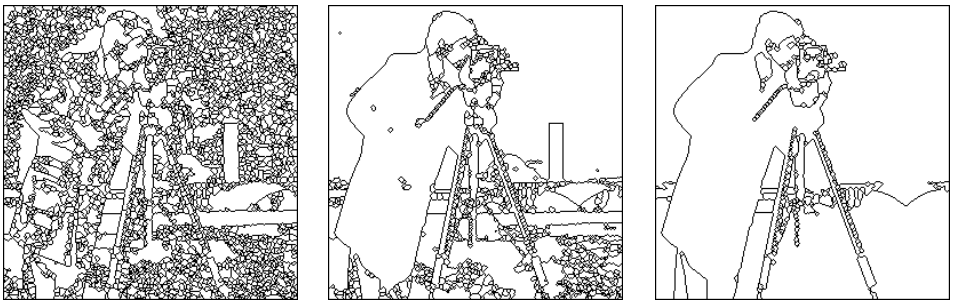
Pour l'ultramétrique d'inondation, la boule maximale de rayon inférieur ou égal à h et de centre c est formée par l'ensemble de bassins versants contenant c et dont les lacs ont fusionné quand l'eau a atteint le niveau d'inondation h. À chaque valeur de h on fait correspondre une partition, où chaque région est formée par l'ensemble de bassins versants ayant fusionné pour le niveau d'inondation h. Chacune des régions de cette partition correspond à une boule de l'ultramétrique. Pour $h = 0$, on a le niveau le plus fin de la hiérarchie. Pour des valeurs de h de plus en plus grandes, plus de lacs ont fusionné et on a donc des partitions de plus en plus grossières. Remarquons cependant le fait suivant : deux lacs fusionnent seulement quand l'eau atteint le point le plus bas sur la frontière qui sépare deux bassins versants. Ainsi, au lieu de faire correspondre une partition à chaque niveau d'inondation, il suffit de retenir les hauteurs associées au point le plus bas au long de chacune des frontières entre bassins versants.

La figure 6.4 illustre la hiérarchie donnée par l'ultramétrique d'inondation. L'image a été filtrée par nivellement en prenant comme marqueur un filtre alterné séquentiel de taille 10 (taille de l'image : 256x256), afin d'obtenir un gradient morphologique moins bruité. Les figures 6.4-(d), (e) et (f) illustrent les boules de l'ultramétrique d'inondation pour différents rayons. Les boules de rayon nul donnent la partition la plus fine, formée par l'ensemble des bassins versants de l'image gradient. Les boules maximales de rayon plus petit ou égal à 5 donnent une partition de l'image en 1000 régions. Cette partition fusionne les bassins versants dont les lacs ont fusionné quand l'eau a atteint le niveau 5. De la même façon l'ensemble des boules maximales de rayon plus petit ou égal à 23 donnent une partition de l'image en 300 régions.

6.2.4 Discussion

Nous venons de présenter trois hiérarchies qui dérivent naturellement de certaines transformations morphologiques. Nous discutons maintenant de leur performance à deux points de vue : la façon dont les zones de transition sont traitées, et la qualité de la segmentation obtenue pour des niveaux de la hiérarchie avec peu de régions.

(a) Image originale (b) Image filtrée (c) Gradient



(d) $r = 0$ (e) $r = 5$ (f) $r = 23$

FIG. 6.4 – Famille de partitions emboîtées donnée par l'ultramétrique d'inondation.

Zones de transition Dans le cas des zones plates croissantes, les partitions obtenues présentent de grandes zones plates significatives, entourées d'un grand nombre de petites régions sans intérêt pour la segmentation. Quand on monte dans la hiérarchie, les grandes zones plates fusionnent entre elles, tandis que les groupes de petites régions ont tendance à persister. Ces petites régions correspondent à des zones de l'image très peu plates, c'est-à-dire, soit à des zones très texturées, soit à des forts contours. Si un pré-filtrage a été appliqué sur l'image avant la détection des zones plates, on peut supposer que les petites textures ont été éliminées et que les ensembles de petites régions correspondent à des zones de transition, ou contours, de l'image. Or ce que l'on cherche à déterminer dans la segmentation c'est précisément la position exacte des contours, et l'atomisation des régions de transition n'aide pas à la résolution du problème. Un problème similaire apparaît lorsqu'on utilise les zones plates des nivellements. La hiérarchie donnée par la LPE ne présente pas ce problème, puisque les contours des régions sont précisément définis par les bassins versants issus de la LPE.

Niveaux élevés de la hiérarchie Dans le contexte de la segmentation interactive dans des applications multimédia on s'applique souvent à extraire les objets les plus importants dans l'image. Ainsi, on cherche des partitions contenant très peu de régions, une pour chaque objet d'intérêt. Ces partitions correspondent aux niveaux les plus élevés de la hiérarchie. L'évaluation de la qualité du point de vue de la segmentation interactive passe donc par la comparaison des performances pour les niveaux de ces hiérarchies contenant très peu de régions. Cependant, en examinant ces hiérarchies pour des niveaux élevés (deux ou trois régions), il devient clair que les segmentations obtenues ne sont pas significatives d'un point de vue sémantique. La raison de ce comportement est l'utilisation de critères basés exclusivement sur des différences

de couleur ou de niveau de gris (zones plates), ou sur des contrastes locaux (cas de la LPE). Ainsi, les régions ayant une couleur similaire sont fusionnées indépendamment de leur taille ou d'autres caractéristiques globales. Les régions qui survivent à la fin sont les plus contrastées, qui souvent correspondent à de très petites régions sans importance visuelle, voire à du bruit. À titre d'exemple considérons la hiérarchie donnée par la LPE, illustrée dans la figure 6.4. Dans cette figure nous avons montré les partitions de la hiérarchie avec 1000 et 300 régions, qui donnent une bonne représentation du contenu de l'image. Mais regardons maintenant la figure 6.5, qui montre la partition de cette même hiérarchie avec seulement 10 régions (boules de rayon 112). Il est clair que ces 10 régions ne représentent pas correctement le contenu de l'image. Ce problème est présent dans la plupart des techniques de segmentation hiérarchique existantes.

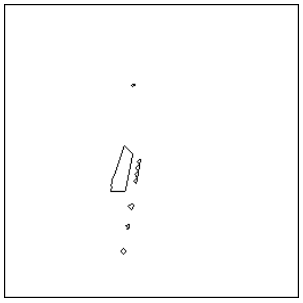


FIG. 6.5 – Partition avec 10 régions (r = 112) appartenant à la hiérarchie de la figure 6.4.

6.3 Hiérarchies sur les régions de la LPE

Nous avons vu dans la section précédente trois façons de dériver des familles de partitions emboîtées à partir de certains opérateurs morphologiques. Parmi eux, seule la LPE ne présente pas le problème d'atomisation des régions de transition. En effet, les régions de la LPE présentent un bon support pour une hiérarchie, puisqu'elles représentent la totalité des contours de l'image. Nous avons vu également que la hiérarchie associée à l'inondation donne de bonnes représentations du contenu de l'image pour des niveaux de la hiérarchie relativement fins. Cependant, pour des niveaux avec peu de régions la hiérarchisation n'est pas adaptée au contenu de l'image. Tout en gardant le support donné par les régions de la LPE, nous présentons ici d'autres ultramétriques permettant de produire des hiérarchies sur les régions de la LPE et dont les partitions avec peu de régions sont plus pertinentes.

Dans ce qui suit, nous considérons l'inondation sur une *image gradient*. Cette image peut être obtenue à partir d'une image à niveaux de gris en calculant son gradient morphologique (défini comme la dilatation moins l'érosion). Sur des images couleur, le calcul du gradient se fait à partir de la définition d'une distance couleur, comme nous le détaillerons dans la section 6.4.3.

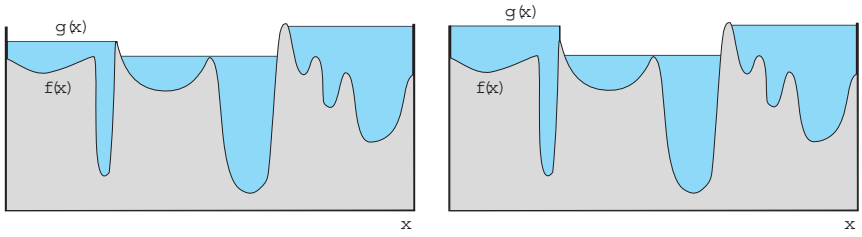
Nous présentons d'abord un cadre général définissant les familles d'inondations, puis nous nous concentrons sur les hiérarchies obtenues avec un type particulier d'inondation : l'inondation synchrone.

6.3.1 Les inondations [57]

Définition et propriétés

Définition 6.1 *Une fonction g est une* **inondation** *d'une fonction f si et seulement si $g \geq f$ et pour tout couple de points voisins (p, q) : $g_p > g_q \Rightarrow g_p = f_p$.*

La figure 6.6 illustre deux exemples de fonction g : un qui est une inondation et un autre qui ne l'est pas. Sur une inondation g, si un point p a un voisin q de valeur plus petite, c'est parce que g_p a atteint un point de col, c'est-à-dire $g_p = f_p$.



(a) Inondation (b) Inondation impossible

FIG. 6.6 – Inondation g(x) d'une fonction f(x) : cas correcte et cas impossible.

Par négation de la définition, on a, pour tout couple de points p, q voisins :

$$\left.\begin{array}{l} g_q > f_q \\ g_p > f_p \end{array}\right| \Rightarrow \quad g_p = g_q$$

Ainsi, la fonction g est composée de zones plates partout où elle n'est pas égale à la fonction f. Ces zones plates constituent les *lacs* de l'inondation g.

On peut considérer des familles d'inondations telles que g^i est une inondation de g^{i-1}, avec $g^0 = f$. Les lacs, ou zones plates, de g^{i-1} sont toujours incluses dans celles de g^i. En effet, si g^i est une inondation de g^{i-1}, alors g^i contient soit des zones plates, soit des points où $g^i = g^{i-1}$. Ainsi, on a que $g^{i-1}_p = g^{i-1}_q \Rightarrow g^i_p = g^i_q$ (connexité). Au fait, les inondations sont des nivellements particuliers.

Le processus d'inondation de la LPE est un cas particulier de famille d'inondations g^h obtenues pour chaque niveau h de l'eau : $g^h = \begin{cases} f & \text{si } f > h \\ h & \text{si } f \leq h \end{cases}$

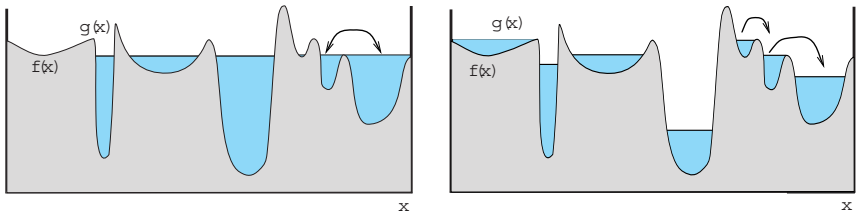
Le critère suivant donne une définition alternative d'inondation et permet de transformer une fonction quelconque h en un nivellement g de f.

Critère 6.1 *Une fonction g est un nivellement d'une fonction f si et seulement si $g = f \vee \epsilon g$.*

Ainsi, une fonction h quelconque peut être transformée en inondation de f par itération de l'érosion géodésique de h dans f jusqu'à idempotence, c'est-à-dire en calculant la fermeture par reconstruction de f en prenant h comme marqueur.

Hiérarchies associées aux inondations Considérons une famille d'inondations g^i d'une fonction f, telle que chaque g^i est une inondation du niveau précédent g^{i-1} avec $g^0 = f$. Considérons également l'ensemble des bassins versants BV^i des fonctions g^i, qui constitue une partition de l'image. Chaque bassin versant de g^i est l'union de bassins versants de f. Ainsi, les bassins versants des fonctions g^i constituent une hiérarchie de partitions emboîtées dont le niveau le plus fin est donné par les bassins versants de f. Le nombre de bassins

versants de g^i diminue par rapport à celui de g^{i-1} lorsque le niveau de l'eau (valeur de la fonction g^i) atteint un point de col de f. Cet événement peut signifier une fusion de deux ou plusieurs lacs (fig. 6.7-(a)), ou qu'un lac s'est rempli (fig. 6.7-(b)) (absorption). Chaque fusion ou absorption de bassins versants se produit pour un niveau i de la famille de fonctions. Une distance ultramétrique peut être définie entre les bassins versants de f comme le niveau i le plus petit pour lequel il existe un bassin versant dans g^i les contenant tous les deux.



(a) Fusion de lacs (b) Absorption

FIG. 6.7 – Fusion et absorption de bassins versants.

Dans la section suivante nous présentons des hiérarchies d'inondations autres que celle donnée par la LPE, et qui sont mieux adaptées au contenu des images.

6.3.2 L'inondation synchrone

6.3.2.1 Définition

L'inondation synchrone [56] diffère de l'inondation classique ou *uniforme* en ce que les bassins versants ne sont pas inondés à hauteur de l'eau constante. Ici, on suppose qu'une source alimente en eau chacun des bassins versants en faisant augmenter de manière constante avec le temps un certain paramètre associé aux lacs. Des exemples de cette quantité, que nous appellerons *critère d'inondation*, sont la profondeur, la surface ou le volume des lacs. Ainsi, l'inondation à profondeur constante fait croître les lacs de manière à ce qu'en tout moment la hauteur de la colonne d'eau à l'intérieur du bassin versant soit la même pour tous les lacs. De la même manière, pour l'inondation à surface constante, la source associée à chaque bassin versant produit autant d'eau que nécessaire pour que la surface des lacs soit la même, et de même pour le volume. Remarquons que l'inondation uniforme est un cas particulier d'inondation synchrone où le paramètre qui augmente régulièrement par unité de temps est la hauteur de l'eau. Nous parlerons de la *vitesse* de croissance d'un lac comme le nombre d'unités augmentées par unité de temps. Les unités peuvent correspondre à des unités de profondeur, de surface ou de volume, selon le type d'inondation.

Pendant l'inondation synchrone, les bassins versants sont remplis à des instants t différents. Quand un bassin versant A atteint la ligne de crête, on l'empêche de déborder dans le lac voisin B en arrêtant sa source. On considérera dès cet instant que le bassin versant A a été absorbé par le bassin versant B, dont le lac continuera à progresser. On peut également établir une relation d'appartenance entre bassins versants, en disant que le bassin versant A appartient au bassin versant B. Puisque la source du bassin versant A a été arrêtée, mais que la source de B est encore active, au moment où le niveau de l'eau du lac B atteindra le point de col séparant A et B, ces deux lacs continueront de progresser ensemble. Dans le cas particulier de l'inondation uniforme, les deux bassins versants A et B atteignent la ligne de crête au même instant t.

Remarquons ainsi que, pour l'inondation synchrone, tous les lacs partagent une même mesure (choisie au préalable), *sauf ceux qui sont remplis* qui, eux, s'arrêtent en attendant que le niveau de l'eau dans un des bassins versants voisins atteigne le point de col le plus bas.

La figure 6.8 illustre ce type d'inondation à une dimension pour le cas de profondeur constante. Les lacs sont inondés de façon à ce que la profondeur de l'eau soit la même en tout moment. Les premiers lacs à se remplir, à l'instant t_1, sont les moins profonds, A et E, qui arrêtent à ce moment leur source. On dit alors que le bassin versant A a été absorbé par le bassin versant B, et que le bassin versant E a été absorbé par le bassin versant F. Ensuite, à l'instant t_2, ce sont les lacs C et F qui se remplissent. Ces deux lacs arrêtent leurs sources respectives et sont absorbés par leurs voisins, D et G, respectivement. À l'instant t_3 le lac G est rempli, et la hauteur du lac B atteint la hauteur du lac A, ce qui fait que le lac A pourra maintenant continuer à progresser, animé par la source du lac B. Le lac B est rempli à l'instant t_4, lorsque l'eau atteint le point de col le séparant du bassin versant C.

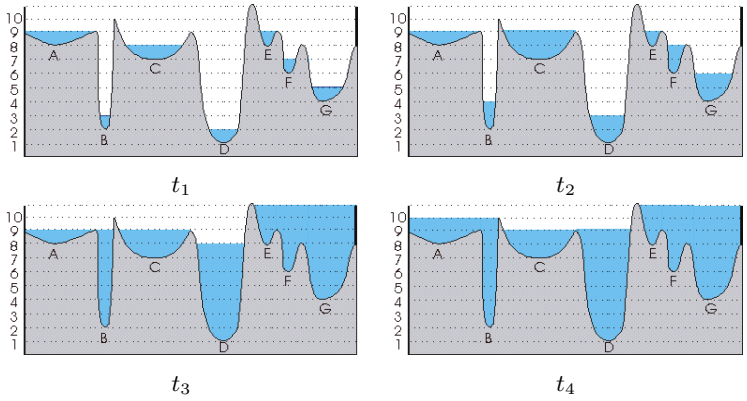


FIG. 6.8 – Inondation synchrone à profondeur constante.

Ainsi, l'inondation synchrone définit une hiérarchie d'inondations donnée par l'état des lacs aux instants t_i où des absorptions se produisent. C'est-à-dire, si w_p donne le niveau de l'eau pour le pixel p à un instant t_i, l'inondation associée g^i est donnée par :

$$g_p^i = \begin{cases} f_p & \text{si } f_p > w_p \\ w_p & \text{si } f_p \leq w_p \end{cases}$$

6.3.2.2 Ultramétriques associées

Il est possible d'obtenir de nouvelles distances ultramétriques basées sur l'inondation synchrone en définissant la distance entre deux bassins versants comme l'instant t où, moyennant le processus d'inondation synchrone, ils appartiennent a un même bassin versant. De cette manière, dans l'exemple de la figure 6.8, on a $d(A, B) = t_1$ et $d(C, D) = t_2$.

Pour vérifier qu'il s'agit bien d'une ultramétrique, il est nécessaire de vérifier l'inégalité ultramétrique. En effet, trois lacs x, y et z sont regroupés dans le même bassin versant forcément plus tard que seulement deux de ces lacs.

Comme pour l'inondation uniforme, la série de partitions associée aux absorptions successives de bassins versants constitue une hiérarchie, où l'indice de stratification est le temps t. Les lacs prenant plus de temps à se remplir seront à plus grande distance de leurs voisins et apparaîtront ainsi aux niveaux les plus élevés de la hiérarchie.

6.3.2.3 Chemins d'inondation

Sur un relief topographique, il existe de multiples chemins à travers lesquels deux minima A et B peuvent se rejoindre.

Considérons d'abord le cas de l'inondation uniforme à partir de tous les minima de l'image. Dans ce processus, l'eau s'étend en inondant d'abord les points du relief les plus bas, et ce n'est qu'une fois ces points inondés que le niveau de l'eau peut continuer à monter. Ainsi, les lacs provenant de deux minima A et B vont se rejoindre en passant par le chemin dont la hauteur maximale est plus petite. Si l'on définit le coût d'un chemin comme sa hauteur maximale, alors le chemin suivi par l'eau lors de l'inondation uniforme est celui de coût minimum.

Pour l'inondation synchrone, les absorptions entre les bassins versants se produisent dans un ordre différent, selon le critère d'inondation choisi. Cependant, les chemins suivis par l'eau pour passer d'un bassin versant à l'autre sont les mêmes. En effet, pour l'inondation synchrone comme pour l'inondation uniforme, un bassin versant fusionnera toujours d'abord avec celui de ses voisins qu'il peut atteindre par le col le plus bas (hauteur maximale plus petite).

On peut conclure en disant que, pour tout processus d'inondation, deux minima se rejoignent toujours en suivant le chemin de hauteur maximale plus petite.

6.3.3 Caractéristiques des hiérarchies obtenues

Les hiérarchies obtenues par inondation synchrone diffèrent énormément selon le critère d'inondation choisi.

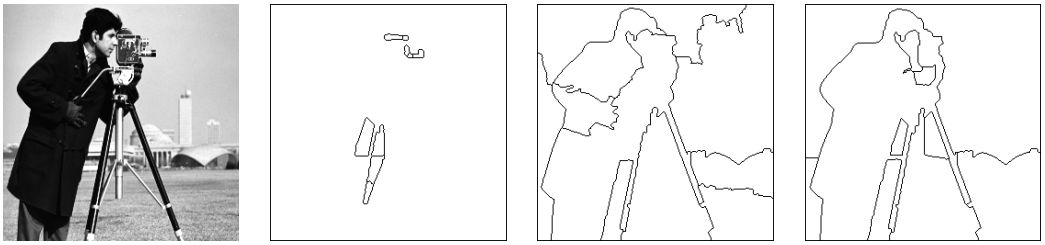
Pour l'inondation synchrone à profondeur constante, ce sont les bassins versants les plus profonds qui prennent le plus de temps à se remplir. Ce sont ceux qui persisteront jusqu'à la fin, et qu'on trouvera sur les partitions les plus grossières de la hiérarchie. L'inondation ayant lieu sur une image gradient, les bassins versants profonds correspondent à des régions de l'image très contrastées.

Dans le cas de l'inondation à surface constante, les bassins versants ayant une petite surface seront absorbés en premier, indépendamment de leur profondeur. Ainsi, ce sont les structures de grande taille qui se situeront en haut de la hiérarchie.

Le volume représente un compromis entre surface et contraste des régions. Les bassins versants qui persisteront seront ceux ayant un grand volume, c'est-à-dire les structures de l'image à la fois grandes et contrastées. Les petites structures persisteront seulement si elles sont suffisamment contrastées pour compenser le volume des structures plus grandes et moins contrastées.

La figure 6.9 illustre les résultats obtenus avec ces trois critères. Remarquons que, par rapport à l'inondation synchrone basée sur la surface, le volume place en haut de la hiérarchie des régions plus petites, mais plus constrastées, comme c'est le cas du visage.

Afin de comparer la qualité des hiérarchies obtenues avec les trois critères de manière objective, nous avons construit pour chaque partition une représentation de l'image en associant à chaque région sa couleur moyenne. À partir de cette représentation nous avons calculé le PSNR. Le PSNR est une mesure objective et facile à calculer de la qualité d'une image. Cependant, elle ne correspond pas aux critères de notre perception. En effet, le PSNR pénalise la disparition de régions contrastées, tandis que la perception humaine ne le fait pas si ces régions sont suffisamment petites. Toutefois, le PSNR peut être utilisé, avec précaution, en tant que mesure objective. La figure 6.10 illustre l'évolution du PSNR par rapport au nombre de régions pour les hiérarchies associées à la profondeur, à la surface et au volume. Comme



(a) Image originale (b) Profondeur (c) Surface (d) Volume

FIG. 6.9 – Niveau de la hiérarchie avec 10 régions pour l'inondation synchrone basée sur la profondeur, la surface et le volume.

nous nous intéressons principalement aux partitions avec peu de régions, nous présentons avec plus de détail les premiers 200 niveaux de la hiérarchie. On remarque la supériorité de la profondeur, ce qui est prévisible puisque le PSNR pénalise la disparition de régions contrastées. Toutefois, pour les niveaux les plus grossiers (moins de 18 régions), c'est le volume qui offre la représentation avec la plus faible erreur. Nous montrons également les images correspondantes au niveau de la hiérarchie avec 20 régions. On constate que la supériorité du critère de profondeur du point de vue du PSNR ne correspond pas aux critères visuels.

Les figures 6.11 à 6.14 présentent des résultats obtenus pour différents types d'images. Pour chaque image, nous présentons les hiérarchies basées sur la profondeur, la surface et le volume pour 100, 50 30 et 10 régions.

Des trois hiérarchies, celle basée sur le volume correspond le mieux à la perception humaine. C'est la hiérarchie que nous utiliserons dans un contexte générique où on n'a pas de connaissances *a priori* sur le type d'images à traiter. On remarque toutefois que cette hiérarchie offre les meilleures performances lorsque les objets à segmenter sont grands et situés sur le premier plan de l'image. Les hiérarchies basées sur la profondeur et la surface seront utiles dans des applications spécifiques où les caractéristiques des objets à détecter sont connues à l'avance.

6.3.4 Représentations

L'information minimale que doit contenir la représentation d'une hiérarchie est :
- les éléments minimaux de la hiérarchie (partition fine) ;
- la structure ou chemin des fusions ;
- l'ordre dans lequel ces fusions se produisent.

De plus, dans certains types d'application il peut être nécessaire de stocker d'autres informations concernant par exemple les caractéristiques des régions ou la totalité des relations de voisinage. Le type d'informations devant être stockées déterminera en partie la représentation choisie.

D'autre part, il est intéressant qu'une telle représentation ait une structure aussi simple que possible et permette un accès rapide aux informations. Cela est particulièrement vrai pour des applications interactives, où le temps de réponse aux actions de l'utilisateur doit être comparable à son temps de réaction.

Les représentations sous forme de graphe synthétisent l'information de façon très efficace. De plus, de telles représentations permettent d'exploiter des résultats de recherches menées dans d'autres domaines où les graphes sont aussi utilisés. Dans le chapitre 3, section 3.5, nous

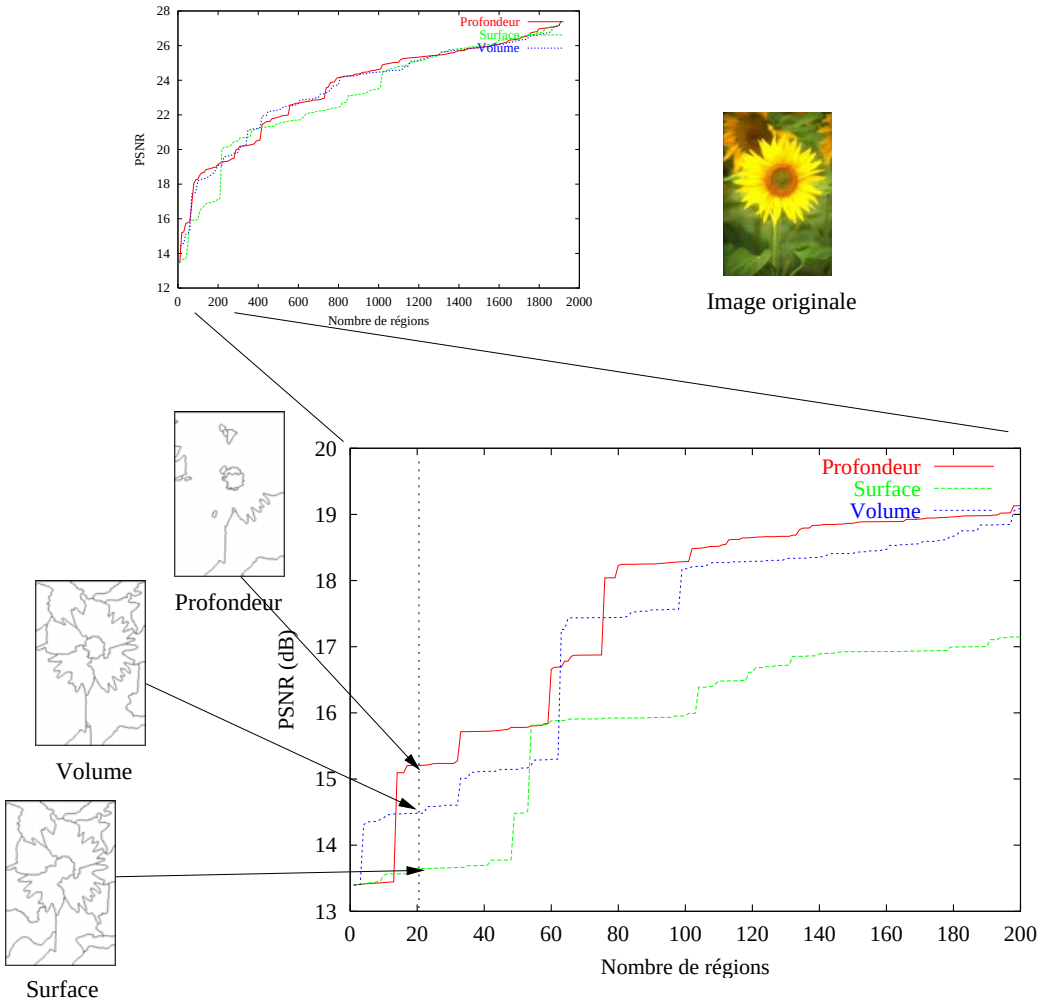


FIG. 6.10 – PSNR associé aux différents niveaux des hiérarchies basées sur la profondeur, la surface et le volume.

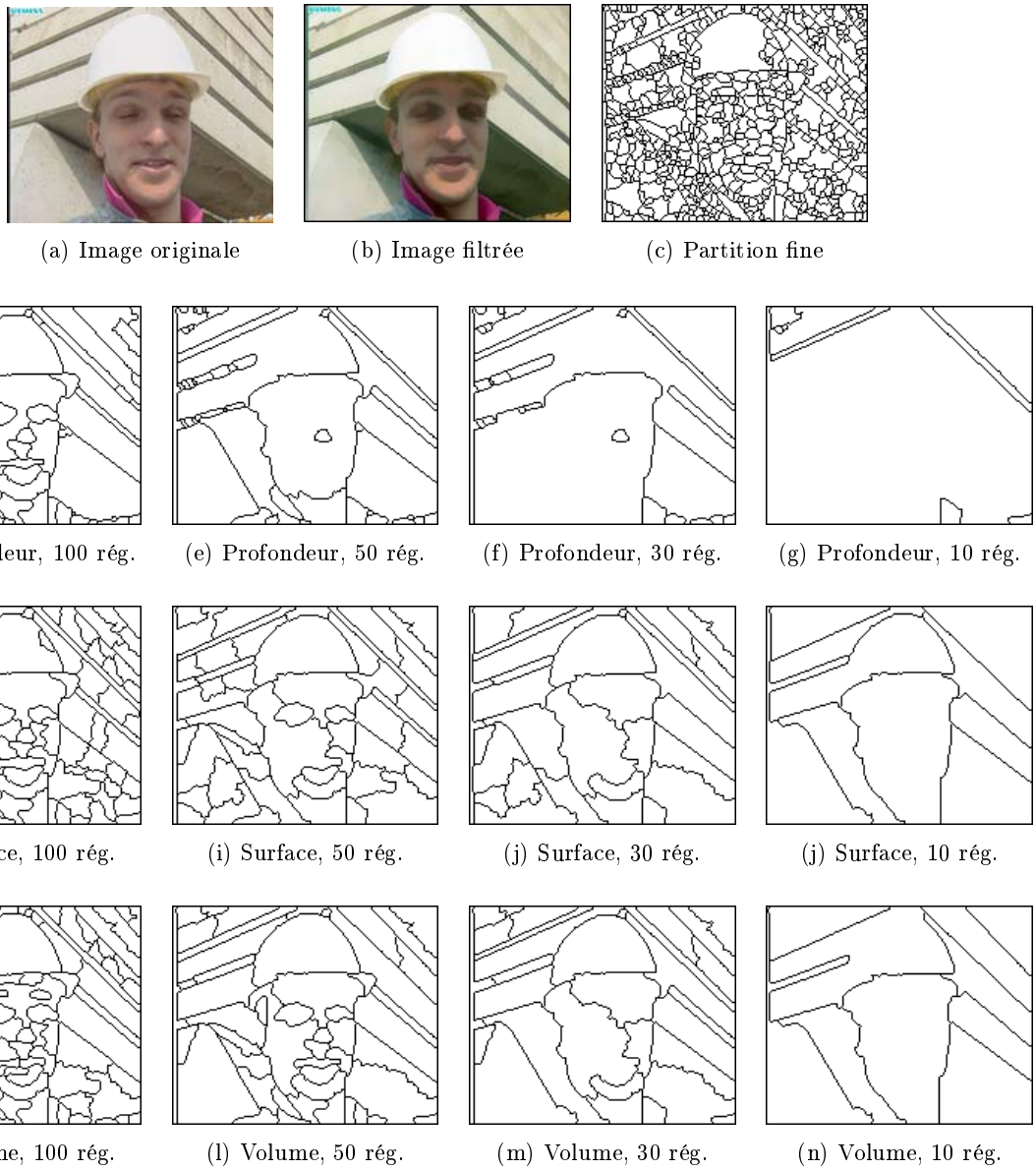
(a) Image originale (b) Image filtrée (c) Partition fine

(d) Profondeur, 100 rég. (e) Profondeur, 50 rég. (f) Profondeur, 30 rég. (g) Profondeur, 10 rég.

(h) Surface, 100 rég. (i) Surface, 50 rég. (j) Surface, 30 rég. (j) Surface, 10 rég.

(k) Volume, 100 rég. (l) Volume, 50 rég. (m) Volume, 30 rég. (n) Volume, 10 rég.

FIG. 6.11 – Hiérarchies obtenues par inondation synchrone basée sur la profondeur, la surface et le volume avec 100, 50, 30 et 10 régions.



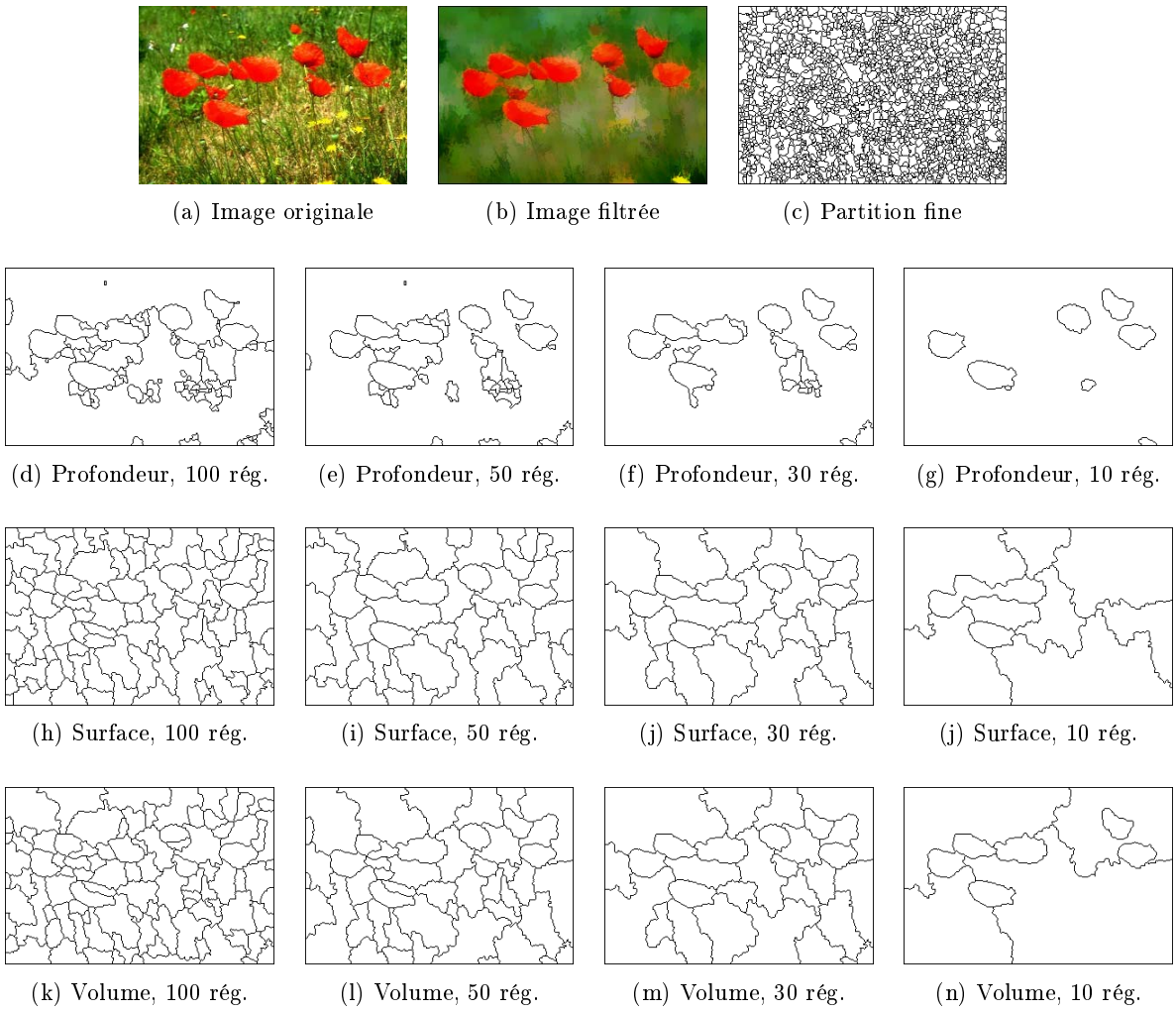
(a) Image originale (b) Image filtrée (c) Partition fine

(d) Profondeur, 100 rég. (e) Profondeur, 50 rég. (f) Profondeur, 30 rég. (g) Profondeur, 10 rég.

(h) Surface, 100 rég. (i) Surface, 50 rég. (j) Surface, 30 rég. (j) Surface, 10 rég.

(k) Volume, 100 rég. (l) Volume, 50 rég. (m) Volume, 30 rég. (n) Volume, 10 rég.

FIG. 6.12 – Hiérarchies obtenues par inondation synchrone basée sur la profondeur, la surface et le volume avec 100, 50, 30 et 10 régions.



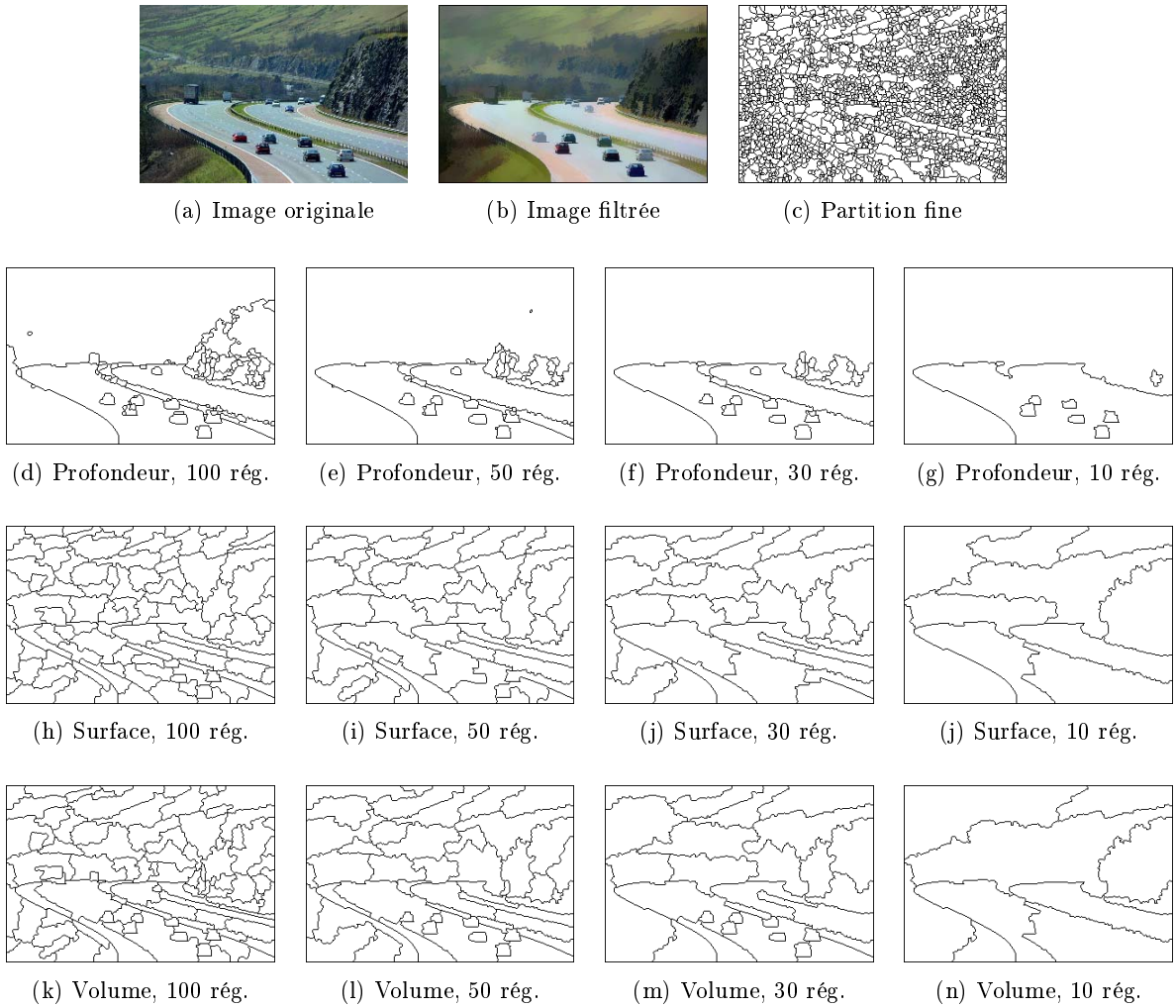
(a) Image originale (b) Image filtrée (c) Partition fine

(d) Profondeur, 100 rég. (e) Profondeur, 50 rég. (f) Profondeur, 30 rég. (g) Profondeur, 10 rég.

(h) Surface, 100 rég. (i) Surface, 50 rég. (j) Surface, 30 rég. (j) Surface, 10 rég.

(k) Volume, 100 rég. (l) Volume, 50 rég. (m) Volume, 30 rég. (n) Volume, 10 rég.

FIG. 6.13 – De haut en bas, hiérarchies obtenues par inondation synchrone basée sur la profondeur, la surface et le volume avec 100, 50, 30 et 10 régions.



(a) Image originale (b) Image filtrée (c) Partition fine

(d) Profondeur, 100 rég. (e) Profondeur, 50 rég. (f) Profondeur, 30 rég. (g) Profondeur, 10 rég.

(h) Surface, 100 rég. (i) Surface, 50 rég. (j) Surface, 30 rég. (j) Surface, 10 rég.

(k) Volume, 100 rég. (l) Volume, 50 rég. (m) Volume, 30 rég. (n) Volume, 10 rég.

FIG. 6.14 – Hiérarchies obtenues par inondation synchrone basée sur la profondeur, la surface et le volume avec 100, 50, 30 et 10 régions.

avons vu deux possibles représentations d'une hiérarchie sous forme d'arbre : le dendrogramme et l'arbre de poids minimum. Nous analysons ici, avec un peu plus de détail, leur adéquation à la représentation des hiérarchies données par les régions de la LPE.

6.3.4.1 L'arbre de lacs critiques

L'arbre de lacs critiques [12] est un dendrogramme des absorptions entre bassins versants. Les nœuds terminaux correspondent aux minima de l'image. Chaque absorption est représentée par un nouveau nœud sur l'arbre. Chaque nœud a un indice de stratification associé, qui peut être donné par n'importe quelle fonction croissante de l'ordre d'absorption. En particulier, et dans le contexte d'inondation synchrone, il est intéressant de prendre l'instant t où l'absorption s'est produite. Dans certaines applications, il peut être important de connaître non seulement le moment auquel l'absorption s'est produite, mais aussi quel lac a été absorbé et quel lac a survécu. Ces informations peuvent être enregistrées sur l'arbre de lacs critiques en étiquetant chaque nœud avec l'étiquette de celui de ses successeurs ayant survécu lors de l'absorption.

L'accès aux différents niveaux de la hiérarchie se fait en ne gardant que les nœuds dont l'indice de stratification est inférieur au niveau recherché et en prenant les sommets de l'arbre résultant, tel qu'illustré dans la figure 6.15, où le nœud d'indice le plus haut, G, a été supprimé. Les nouveaux sommets de l'arbre, E et F, donnent la segmentation résultante, en fusionnant leurs nœuds terminaux. .

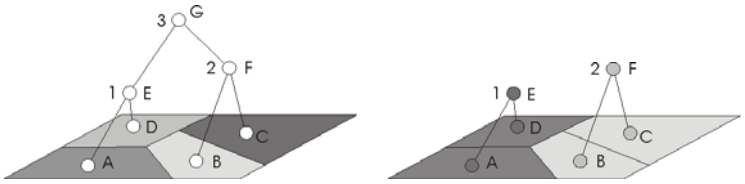


FIG. 6.15 – Accès à la segmentation associée à un niveau de la hiérarchie sur l'arbre de lacs critiques.

Un tel dendrogramme se construit de bas en haut, en parallèle avec le processus d'inondation. Initialement, un nœud du dendrogramme est associé à chaque minimum de l'image gradient. Puis, pendant l'inondation, à chaque fois que deux lacs se rencontrent un nouveau nœud du dendrogramme est créé, représentant l'absorption d'un des bassins versants par son voisin. Également, l'instant t où cette absorption s'est produite est associé au nouveau nœud. La figure 6.16 illustre la procédure.

Étudions maintenant la représentation sous forme d'arbre aux arêtes valuées.

6.3.4.2 Arbre aux arêtes valuées

Nous parlons ici de la représentation de la hiérarchie sous forme d'arbre aux arêtes valuées. Nous avons défini, dans le chapitre 3, l'arbre de poids minimum :

Définition 6.2 *Étant donné un ensemble d'éléments et une distance, ou poids, associé à toute paire d'éléments, l'arbre de poids minimum est un arbre reliant tous les éléments et de poids total minimum.*

Nous avons vu dans la section 6.3.2.3 que le chemin suivi par l'eau lors de l'inondation est le même pour tous les types d'inondation. Deux bassins versants fusionnent toujours à

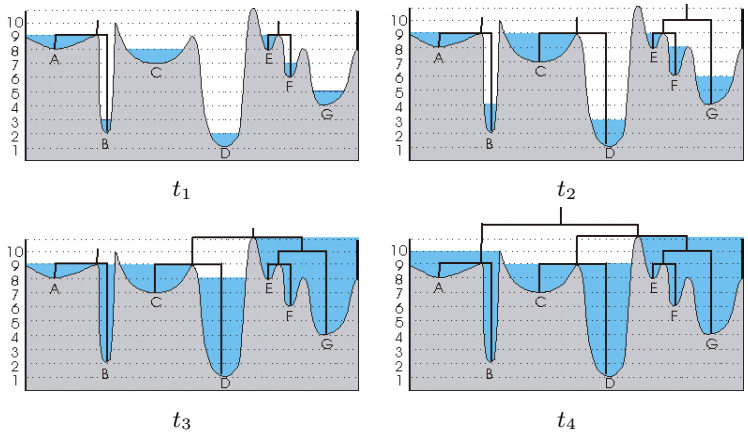


FIG. 6.16 – Construction de l'arbre de lacs critiques pendant l'inondation synchrone.

travers le chemin de hauteur maximale plus petite. Le théorème suivant établit la relation entre l'arbre de poids minimum et l'inondation.

Théorème 6.1 *[31] Parmi tous les chemins possibles entre deux éléments, celui choisi par l'arbre de poids minimum est celui dont la valeur maximale est plus petite.*

Dans notre cas de figure, l'ensemble d'éléments est constitué par les bassins versants de l'image. Puisque l'inondation choisit toujours le chemin de hauteur maximale plus petite pour rejoindre deux bassins versants, nous pouvons conclure que l'arbre de poids minimum représente exactement les chemins d'inondation. Ainsi, cet arbre peut être utilisé pour représenter de façon simplifiée la structure des absorptions. Les arêtes de l'arbre de poids minimum sont valuées avec la hauteur du point de col séparant les bassins versants. Or l'ordre de fusion donné par ces valeurs d'arêtes correspond à l'ordre de fusion dans l'inondation uniforme, et non pas à celui de l'inondation synchrone. Pour que cet arbre représente la hiérarchie obtenue par inondation synchrone, il est nécessaire de valuer les arêtes avec l'instant t où les lacs associés à deux bassins versants fusionnent, c'est-à-dire que c'est l'ordre de remplissage des lacs qui détermine la hiérarchie.

L'accès aux différents niveaux de la hiérarchie se fait dans une telle structure en éliminant plus ou moins d'arêtes par ordre de valeurs selon le niveau de la hiérarchie auquel on veut accéder. Chacune des composantes connexes sur l'arbre correspondra ainsi à une région sur la partition. Pour obtenir une partition en n régions, il faut éliminer les $n-1$ arêtes de plus forte valeur.

La figure 6.17 illustre l'accès aux différents niveaux de la hiérarchie représentée par l'arbre aux arêtes valuées. La suppression des $N-1$ plus fortes arêtes produit une forêt avec N arbres. Chaque arbre correspond à une composante connexe sur la segmentation résultante. La segmentation résultante est illustrée dans la figure 6.17 par des régions colorées avec le même niveau de gris. Quand toutes les arêtes sont supprimées, on obtient la partition fine.

6.3.4.3 Discussion

Le dendrogramme et l'arbre aux arêtes valuées ne sont que deux manières de représenter la même hiérarchie. Cependant, les différences liées à leur stockage méritent d'être discutées.

Le dendrogramme (arbre de lacs critiques) ayant plus de nœuds et d'arêtes que l'arbre de

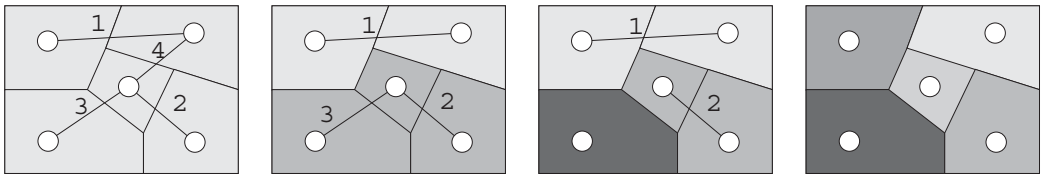


FIG. 6.17 – Accès aux différents niveaux de la hiérarchie donnée par l'arbre de poids minimum.

poids minimum, il nécessite plus de mémoire pour son stockage que l'arbre aux arêtes valuées. En effet, le nombre de nœuds de l'arbre de lacs critiques est $M = 2n - 1$, supposant que les absorptions se produisent toujours entre des paires de lacs et qu'il y a n bassins versants dans l'image : n bassins versants représentés chacun par un nœud, plus un nœud supplémentaire par fusion, et il y a $n - 1$ fusions, ce qui résulte en $2n - 1$ nœuds. Et le nombre d'arêtes sur un arbre est égal au nombre de nœuds moins 1, ce qui donne $N = 2n - 2$ arêtes.

Pour l'arbre aux arêtes valuées on a (théorème 3.1) $M = n$ nœuds et $N = n - 1$ arêtes.

L'accès aux niveaux de la hiérarchie se fait aussi différemment pour les deux représentations. Pour le dendrogramme, il s'agit de seuiller les nœuds pour chaque valeur de l'indice de stratification. La partition résultante se construit en prenant la région correspondante aux sommets de l'arbre résultant. Pour l'arbre aux arêtes valuées, le seuillage des arêtes donne lieu à une forêt, où chaque composante connexe correspond à une région de la partition.

En raison de sa plus grande simplicité, nous adopterons la représentation sous forme d'arbre aux arêtes valuées. De plus, nous avons vu dans le chapitre 3 que cette représentation contient plus d'information que l'arbre de lacs critiques, puisqu'il donne les chemins de fusion des régions, qui n'apparaissent pas sur le dendrogramme. Nous avons également vu qu'il est possible de passer à tout moment d'une représentation sous forme d'arbre aux arêtes valuées à une représentation sous forme de dendrogramme.

6.3.5 Algorithme de calcul

Il existe plusieurs algorithmes de calcul de l'arbre de poids minimum [39]. Mais nous avons vu que les chemins suivis par l'eau lors de l'inondation correspondent à ceux choisis par l'arbre de poids minimum. Puisque de toute manière nous sommes obligés de calculer une inondation afin de déterminer la forme des bassins versants, il est possible et intéressant de calculer l'arbre de poids minimum en même temps que l'inondation, ce qui se fait avec un très petit coût additionnel.

L'annexe A décrit un algorithme efficace pour l'inondation uniforme. Dans cet algorithme, une étiquette est associée à chaque minimum. Cette étiquette est propagée sur l'image à partir des minima du gradient et donne à la fin la forme des régions de la partition. Cette partition, qui constitue le niveau le plus bas de la hiérarchie, permet de faire le lien entre l'arbre et l'image. Pour obtenir l'arbre en même temps que l'inondation il faut une deuxième étiquette par lac qui nous permet de suivre les fusions réalisées. Cette étiquette donne la classe à laquelle chaque lac appartient à tout instant. Initialement, chaque lac appartient à une classe différente. En même temps, un nœud de l'arbre est créé pour chaque bassin versant de l'image. L'inondation commence, et à mesure que le niveau de l'eau monte, des absorptions se produisent entre les différents lacs. À chaque fois qu'une absorption a lieu entre deux bassins versants *appartenant à des classes différentes*, une arête est ajoutée sur l'arbre entre les nœuds associés à ces bassins versants. Cette arête est valuée avec la hauteur h de l'eau pour laquelle

la fusion s'est produite. Les deux classes sont alors fusionnées dans une seule. À la fin de l'inondation, tous les nœuds de l'arbre sont connectés par un chemin unique. Les valeurs des arêtes indiquent l'ordre de fusion des nœuds.

La mise en place d'un algorithme d'inondation synchrone est beaucoup plus compliquée puisque l'ordre d'inondation n'est plus donné uniquement par le niveau de gris des pixels. Nous avons cependant vu que l'inondation synchrone et l'inondation uniforme partagent quelques points en commun :

- les chemins d'inondation sont les mêmes et correspondent à ceux donnés par l'arbre de poids minimum ;
- deux lacs fusionnent toujours sur le point de col de la frontière séparant les deux bassins versants ;
- au moment de la fusion, au moins un des deux lacs est rempli – les deux dans le cas de l'inondation uniforme –. Dans le cas de l'inondation synchrone, c'est le lac le plus petit selon la mesure choisie pour l'inondation qui est rempli.

Ainsi, bien que l'ordre des fusions soit différent pour les deux types d'inondation, les chemins d'inondation sont les mêmes. Autrement dit, les mêmes rencontres de lacs se produisent pour les deux inondations, mais à des instants différents. C'est précisément l'ordre de rencontre des lacs qui établit la hiérarchie. Cet ordre d'inondation est donné dans l'inondation synchrone par l'instant t où la fusion se produit. L'instant t est lié à la mesure du lac comme $temps = mesure/vitesse$. La vitesse de croissance étant constante, le paramètre t est directement proportionnel à la mesure du lac, et donc cette mesure peut être utilisée à la place du temps t sans changer à la hiérarchie. Étant donné que la mesure (profondeur, surface, volume) est la même pour le lac le plus petit des deux rencontrés pour les deux types d'inondation (synchrone et uniforme) et que c'est ce paramètre qui détermine l'ordre de fusion, nous pouvons calculer la hiérarchie obtenue par inondation synchrone à partir de l'inondation uniforme. Ainsi, l'algorithme d'inondation uniforme est utilisé à la place d'un algorithme d'inondation synchrone, plus coûteux. L'arbre est construit pendant l'inondation. Quand deux lacs appartenant à des classes différentes se rencontrent, une arête est ajoutée à l'arbre. Mais cette fois ci, l'arête n'est pas valuée avec la hauteur de l'eau pour laquelle la fusion se produit, mais avec la mesure du lac le plus petit.

La figure 6.18 montre un exemple monodimensionnel de calcul de la hiérarchie basé sur le volume. L'arbre contient un nœud pour chacun des minima/bassins versants de l'image. Les premiers lacs à se retrouver lors de l'inondation sont ceux correspondant aux minima 2 et 3. À ce moment, le lac 2 a un volume de 4, tandis que le lac 3 a un volume de 6. On ajoute alors une arête entre les nœuds 2 et 3, valuée avec le volume du lac le plus petit, c'est-à-dire 4. Le lac 2, plus petit, est absorbé par le lac 3. Le volume du lac 3 est actualisé pour refléter cette absorption en lui ajoutant le volume du lac absorbé. Ainsi, le nouveau volume du lac 3 est $6 + 4 = 10$. Les lacs 2 et 3 appartiennent maintenant à la même classe. L'inondation continue, et les prochains lacs à fusionner sont ceux correspondant aux bassins versants 3 et 4. Comme ils appartiennent à des classes différentes, une nouvelle arête est ajoutée sur l'arbre avec le volume du lac de volume plus petit, c'est-à-dire, $\min(18, 7) = 7$, et le volume du lac plus grand est actualisé en lui ajoutant le volume du lac plus petit : $18 + 7 = 25$ pour refléter l'absorption. Les derniers lacs à fusionner sont d'un côté celui composé par les bassins versants 2, 3 et 4, et de l'autre côté celui correspondant au bassin versant 1. Une arête est alors ajoutée entre les nœuds correspondant aux bassins versants voisins 1 et 2, valuée encore avec le volume du bassin versant de moindre volume, c'est-à-dire, $\min(9, 40) = 9$.

La nécessité de grouper les lacs dans des classes ne peut pas être illustrée dans un exemple à

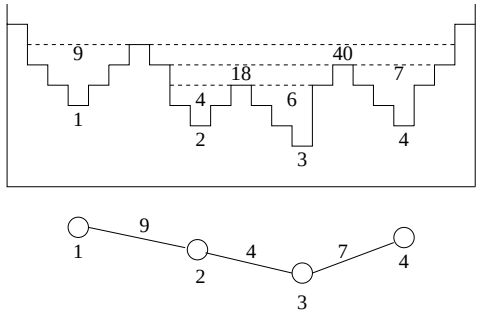


FIG. 6.18 – Exemple de calcul de l'ultramétrique donnée par l'inondation synchrone basé sur l'inondation uniforme.

une dimension. À une dimension, les lacs ne peuvent fusionner qu'en suivant un chemin unique. Cependant, sur une image à deux dimensions il peut exister plusieurs chemins reliant les mêmes deux lacs. Imaginons une situation où un ensemble de lacs A, B, C et D ont fusionné selon la structure A-C-B-D. Ainsi, A et D ne se sont jamais rencontrés, mais ils appartiennent au même lac et il existe déjà un chemin les reliant sur l'arbre de poids minimum. Si à un moment donné ces deux lacs se rencontrent, il est nécessaire de pouvoir détecter qu'ils appartiennent déjà à la même classe et qu'on ne doit donc pas ajouter d'arête sur l'arbre, ce qui entraînerait la création d'un cycle.

6.3.6 Rapport avec les valeurs d'extinction

Les valeurs d'extinction ont été introduites par Grimaud [24], [23] dans ses études sur la dynamique, et étendues par Vachier [99], [98].

Les valeurs d'extinction d'une fonction f se définissent par rapport à une famille d'opérateurs connexes extensifs (resp. antiextensifs) dépendant d'un paramètre λ. De tels opérateurs produisent des versions de la fonction f avec de moins en moins de minima (resp. maxima) régionaux pour des valeurs de λ croissants. La valeur d'extinction d'un minimum (resp. maximum) est alors la valeur maximale de λ pour laquelle le minimum (resp. maximum) n'a pas encore disparu.

Définition 6.3 *Soit M un minimum (resp. maximum) régional d'une fonction f, et $\Psi_\lambda(f)$ une famille croissante de transformations connexes extensives (resp. antiextensives). La valeur d'extinction de M par rapport à Ψ est la valeur maximale de λ telle que M reste un minimum (resp. maximum) régional de $\Psi_\lambda(f)$:*

$$\varepsilon(M) = sup\{\lambda \geq 0 \,|\, \forall \mu \leq \lambda, M \subset Min(\Psi_\mu(f)\}$$

où $Min(g)$ dénote l'ensemble des minima de la fonction g.

Les valeurs d'extinction produisent une hiérarchisation des extrema de l'image selon leur persistance en face de l'opérateur $\Psi_\lambda(f)$. Le choix de la famille d'opérateurs $\Psi_\lambda(f)$ détermine ainsi cette hiérarchisation.

Considérons quelques exemples de familles d'opérateurs $\Psi_\lambda(f)$ utilisées.

Les h-reconstructions Considérons la fonction $\Psi_h(f) = \delta_\infty(f - h, f)$, appelée h-reconstruction et qui désigne la reconstruction géodésique de la fonction $f - h$ dans la fonction

f. La figure 6.19 illustre l'effet d'une telle opération. Les maxima les plus profonds par rapport à ses voisins persistent pour des valeurs de h grands. Cette transformation est une opération connexe, croissante et antiextensive, et permet de définir les valeurs d'extinction des maxima de la fonction f. Les valeurs d'extinction des minima s'obtiennent par dualité. Vachier montre dans [98] que ces valeurs correspondent à la dynamique définie par Grimaud [23] comme $\varepsilon(M) = dyn(M) - 1$.

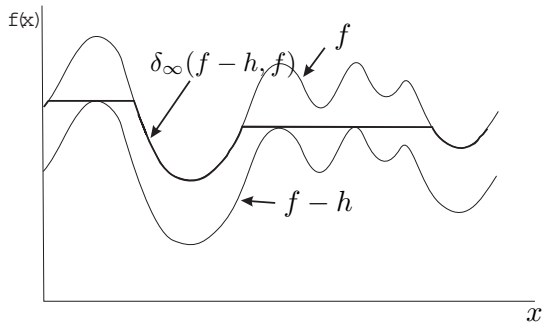


FIG. 6.19 – Exemple de h-reconstruction sur une fonction monodimensionnelle.

Les ouvertures et fermetures aréolaires Elles sont définies à partir de l'ouverture aréolaire binaire :

Définition 6.4 *Considérons un ensemble binaire X. L'ouverture aréolaire binaire de paramètre λ de X se définit comme :*

$$\gamma^a_\lambda(X) = \{x \in X \,|\, Surface\,(C_x(X)) \geq \lambda\} \tag{6.2}$$

où $C_x(X)$ désigne la composante connexe de l'ensemble X contenant le point x.

Les ouvertures et fermetures aréolaires ont été étendues par L. Vincent [100], [101] aux images à niveaux de gris. L'ouverture aréolaire pour les images à niveaux de gris se définit à partir des seuils $X^+_h(f) = \{x | f(x) \geq h\}$ comme :

$$\gamma^a_\lambda(f)(x) = sup\{h \leq f(x) \,|\, x \in \gamma^a_\lambda\left(X^+_h(f)\right)\} \tag{6.3}$$

La fermeture aréolaire se définit par dualité. À la différence des ouvertures et fermetures traditionnelles par reconstruction, les structures de l'image ne sont pas filtrées par rapport à un élément structurant, mais selon leur surface totale. Ainsi, des structures longues et étroites, qui seraient normalement éliminées par les ouvertures et fermetures classiques, seront préservées par un filtre aréolaire.

La figure 6.20 montre l'effet de l'ouverture aréolaire sur une fonction monodimensionnelle. Les maxima sont coupés de manière à ce que leur surface soit au moins égale au paramètre s.

L'arasement volumique L'arasement volumique, introduit par Vachier [98] peut être interprété comme une érosion associée à un élément structurant non plan de volume λ capable d'adapter localement sa forme aux structures de l'image.

Définition 6.5 *Soit f une fonction numérique. L'arasement volumique de taille λ de f noté $a^v_\lambda(f)$ est défini sur les maxima de l'image par :*

$$a^v_\lambda(f)(x) = sup\{s \leq f(x) \,|\, Vol^s_x(f) \geq \lambda\} \tag{6.4}$$

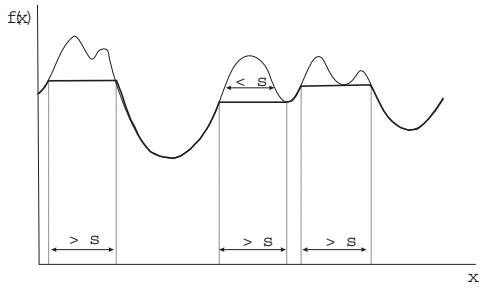


FIG. 6.20 – Exemple de filtre aréolaire sur une fonction monodimensionnelle.

où

$$Vol_x^s(f) = \sum_{y \in C_x(X_s^+(f))} |f(y) - s|$$

La version duale de cet opérateur peut être appliquée aux minima locaux. Dans notre cas, puisque nous travaillons avec des images gradient, nous nous intéressons principalement aux hiérarchies de minima.

Ces opérateurs hiérarchisent les minima de l'image très différemment. Ainsi, les h-reconstructions ne permettent de survivre qu'aux minima suffisamment profonds, tandis que les fermetures surfaciques s'attaquent aux structures de petite surface, indépendamment de leur profondeur. L'arasement volumique est un compromis entre les deux opérateurs précédents, en tenant compte à la fois de la surface et de la profondeur des minima.

Vachier [98] démontre dans sa thèse que les valeurs d'extinction associées aux h-reconstructions, aux fermetures/ouvertures surfaciques et à l'arasement volumique peuvent tous être calculés à partir de simulations d'inondation. Elle montre que la valeur d'extinction associée à un minimum correspond à la mesure de son lac respectif lorsque pendant l'inondation il rencontre un lac plus grand – au sens de la profondeur, la surface ou le volume –. Cette méthode est la même que celle que nous avons utilisée pour calculer les ultramétriques basées sur l'inondation synchrone à partir de l'inondation uniforme. Les ultramétriques définies par inondation synchrone à vitesse de croissance constante correspondent donc aux valeurs d'extinction utilisés pour classer les minima.

6.4 Hiérarchie sur une partition quelconque

L'inondation synchrone permet de construire des hiérarchies très intéressantes avec des critères différents. Ces méthodes obligent cependant à travailler sur une image gradient, dont le nombre de minima détermine le nombre de régions du niveau le plus fin de la hiérarchie. Ce niveau constitue le plus souvent une sursegmentation très importante de l'image. Puisque l'arbre représentant la hiérarchie contient un nœud par région de la partition fine, cette sursegmentation a une influence sur le temps de calcul associé à la manipulation du graphe. Quand les images traitées sont grandes, ou dans le cas des traitements 3D, que nous étudierons dans le chapitre 7, il est nécessaire de réduire le nombre de régions du niveau le plus fin de la hiérarchie. En même temps, la partition fine doit être suffisamment détaillée pour contenir tous les contours significatifs. Ainsi, il serait intéressant de disposer d'une partition fine avec moins de régions que celle obtenue par LPE. Il nous faudrait alors une méthode permettant

de construire des hiérarchies à partir d'une partition fine quelconque, et de qualité semblable à celles obtenues par LPE. Nous présentons ici une méthode basée sur les graphes pour calculer de telles hiérarchies. Cette méthode nous sera d'une grande utilité dans le chapitre 7, où nous étendons le calcul de ces hiérarchies aux séquences vidéo. Mais elle ne le sera que si l'on est capable de construire une partition fine peu sursegmentée et en même temps adaptée au contenu de l'image. Le calcul d'une partition fine avec ces caractéristiques est présenté dans la section 6.4.3.

6.4.1 Graphe de voisinage associé à une partition

On peut associer à une partition quelconque son graphe de voisinage. Chaque région de la partition est représentée par un nœud sur le graphe de voisinage. Deux nœuds sont reliés par une arête si leurs régions respectives sont voisines sur la partition. Les arêtes sont valuées avec une mesure de dissimilarité locale entre les régions. Des exemples de mesures de dissimilarité sont la différence entre les couleurs moyennes des régions ou la valeur moyenne du gradient au long des frontières. Dans notre cas, nous avons trouvé que la distance euclidienne entre les moyennes de couleur des régions dans les espaces RGB ou YUV donne des résultats satisfaisants pourvu que les régions de la partition fine soient homogènes en couleur.

Le graphe de voisinage aux arêtes valuées peut être interprété comme une image gradient synthétique où la hauteur de la frontière séparant deux régions est donnée par la valeur de l'arête sur le graphe. La figure 6.21 illustre cette interprétation. Les hiérarchies obtenues par inondation synchrone que nous avons présentées dans les sections précédentes peuvent être calculées sur cette image gradient synthétique. Cependant, remarquons que le graphe de voisinage contient exactement la même information qu'une telle image, mais qu'elle constitue une représentation plus synthétique. En effet, le graphe contient beaucoup moins de nœuds qu'il n'y a de pixels sur l'image. Ainsi, nous réaliserons les calculs sur le graphe, et ne nous servirons de l'image synthétique qu'à des propos de compréhension.

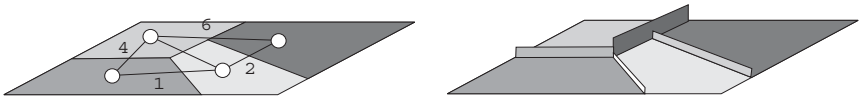


FIG. 6.21 – Graphe de voisinage associé à une partition et interprétation sous forme d'image gradient synthétique.

Dans le cas particulier où la partition de départ correspondrait à la partition obtenue par LPE et où les arêtes du graphe seraient valuées avec la valeur minimale du gradient au long de la frontière entre deux régions, la représentation par gradient synthétique serait équivalente, du point de vue de l'inondation, à l'image gradient originale. En effet, rappelons que les chemins d'inondation sont déterminés par les points de col (valeurs moins fortes) du gradient au long de la frontière entre deux régions. Ainsi, donner une valeur constante à la frontière égale à la hauteur du point de col ne modifie pas les chemins d'inondation.

La représentation sous forme de graphe permet également de développer des algorithmes plus robustes qu'en travaillant directement sur l'image gradient. Le fait que ce soit le chemin de hauteur maximale minimum qui est toujours choisi lors de l'inondation peut amener à ce que des fusions se produisent à travers des passages très étroits sur la frontière lorsqu'on travaille directement sur l'image gradient. Cette situation est illustrée dans la figure 6.22, où les lacs vont fusionner trop tôt et leur mesure (volume, profondeur, surface) sera ainsi

sous-estimée. Souvent, il serait souhaitable de considérer plutôt la valeur moyenne du gradient sur la frontière. La représentation sous forme de graphe permet de valuer les frontières avec différentes mesures de dissimilarité, ce qui ajoute de la flexibilité et de la robustesse aux algorithmes. Il est cependant nécessaire de calculer d'abord une partition fine sur laquelle le graphe est construit.

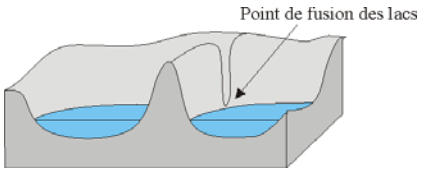


FIG. 6.22 – Passage étroit entre deux bassins versants.

6.4.2 Calcul de l'ultramétrique sur le graphe

Rappelons que les chemins suivis par l'eau lors de l'inondation correspondent aux chemins sur l'arbre de poids minimum. Ainsi, le calcul de l'arbre de poids minimum du graphe de voisinage donne la structure des fusions nécessaires au calcul de la hiérarchie. À ce propos, n'importe quel algorithme de calcul de l'arbre de poids minimum pourrait être appliqué. Cependant, nous avons montré que l'arbre de poids minimum à lui seul ne suffit pas à produire des hiérarchies significatives, et que ce sont les ultramétriques choisies pour valuer ses arêtes qui déterminent la qualité des partitions résultantes. Pour calculer la hiérarchie associée à une partition fine quelconque il est donc nécessaire de calculer ces ultramétriques. Pour le cas des images gradient, nous avons présenté dans la section 6.3.5 un algorithme basé sur l'inondation uniforme. Pour les graphes, il est possible d'utiliser une méthode de calcul similaire, que nous appellerons *inondation du graphe*.

Pour visualiser cet algorithme, il suffit de passer par l'image gradient synthétique équivalente (figure 6.21). En effet, l'algorithme présenté dans la section 6.3.5 peut être appliqué directement à cette image. Cet algorithme réalise une inondation de l'image et calcule l'ultramétrique associée à chaque fois qu'une fusion entre deux lacs se produit. Ces fusions se produisent pour une hauteur d'eau égale à la hauteur de la frontière séparant deux régions. La hauteur de la frontière correspond également à la valeur de l'arête sur le graphe. Ainsi, il est possible de réaliser l'inondation directement sur le graphe, de la manière suivante :

- ordonner les arêtes du graphe de voisinage par ordre croissant de valeurs ;
- pour chaque arête du graphe (traitée par ordre croissant de valeurs), si l'arête n'introduit pas de cycle sur l'arbre :
  - calculer sur le graphe de voisinage une mesure géométrique pour chaque composante connexe (région) ;
  - ajouter cette arête à l'arbre, valuée avec la mesure la plus petite des deux calculées dans l'étape précédente.

L'arbre aux arêtes valuées obtenu à la fin de l'algorithme donne l'ultramétrique définissant la hiérarchie.

Dans cet algorithme, la valeur de l'arête du graphe traitée à tout moment donne la hauteur de l'eau si l'inondation se réalisait sur l'image gradient synthétique équivalente. La mesure géométrique choisie peut être la surface ou le volume. La surface est calculée en faisant la somme des surfaces des régions de la partition fine associées à la composante connexe. Le

volume est calculé en multipliant la surface par la hauteur de l'eau, c'est-à-dire, par la valeur de l'arête du graphe de voisinage active. La dynamique coïncide avec la valeur de l'arête, puisque tous les lacs ont la même valeur de minimum (zéro). Ainsi, pour la dynamique, la hiérarchie est donnée par l'arbre de poids minimum du graphe de voisinage, sans besoin de réévaluer les arêtes.

6.4.3 Calcul de la partition fine

Nous venons de présenter une méthode de calcul de hiérarchies sur une partition fine quelconque. L'intention principale de cette approche était d'éliminer la sursegmentation importante produite par la LPE. Nous proposons ici une technique permettant de calculer une partition fine beaucoup moins sursegmentée que celle donnée par la LPE.

Les zones λ-plates couleur que nous avons vues dans le chapitre 5 permettent de détecter les régions homogènes dans une image couleur. Rappelons qu'en l'absence de textures importantes, la détection des zones λ-plates couleur produit d'une part des grandes régions pour les zones de l'image relativement uniformes en couleur et d'autre part des régions très petites correspondant aux zones de transition. Les contours des objets se trouvent quelque part dans les zones de transition, et chaque objet est marqué par une grande zone plate. En établissant un seuil sur la taille des zones plates et en ne gardant que celles de plus grande surface nous obtenons une image de marqueurs indiquant la position des régions homogènes. Cette image est utilisée comme marqueur pour une ligne de partage des eaux calculée sur l'image gradient. Dans tous les cas, un pré-filtrage est souhaitable afin d'éliminer le bruit ainsi que des textures. La figure 6.23 montre le schéma de calcul d'une telle partition, que nous détaillons ensuite.

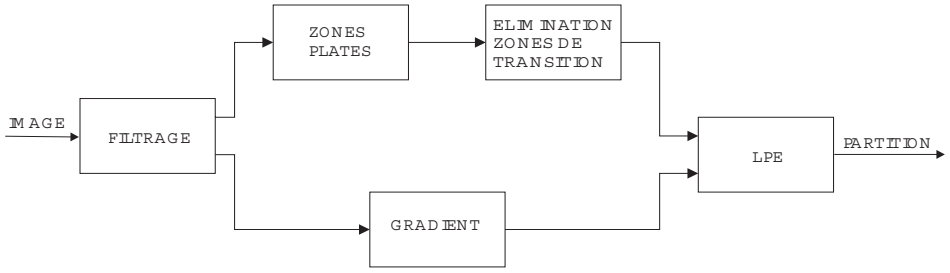


FIG. 6.23 – Calcul d'une partition moyennant la détection des zones quasi-plates.

Préfiltrage Pour cette étape nous utilisons des nivellements. Le filtrage ne doit pas être trop important au risque d'éliminer des structures significatives.

Détection des zones λ-plates Les zones λ-plates couleur sont détectées, ce qui produit d'une part des grandes régions pour les zones de l'image relativement uniformes, et d'autre part des régions très petites correspondant aux zones de transition (figure 6.24-(d)). Les contours des objets se trouvent à l'intérieur des zones de transition, et chaque objet est marqué par une zone plate de taille importante. Les zones plates de taille relativement importante sont considérées comme marquant un objet et sont ainsi retenues pour être utilisées comme marqueurs d'une LPE. Deux seuils doivent être déterminés lors de cette étape : la valeur de λ et le seuil de taille des zones plates retenues. Le seuil de taille peut être déterminé en fonction de la taille de l'image. Nous avons trouvé qu'une valeur d'entre 3 et 5 pixels donne de bons

résultats pour l'ensemble des images de taille QCIF testées. La valeur de λ a des répercussions plus importantes sur la qualité de la segmentation obtenue et doit être déterminée avec plus d'attention. En utilisant une valeur de λ fixe pour toutes les images, les résultats obtenus sont très différents et il est difficile de choisir une valeur de λ qui donne de bons résultats dans tous les cas. Pour fixer la valeur de λ individuellement pour chaque image, nous avons donc utilisé un critère basé sur le pourcentage de l'image couvert par ce que nous appelons les zones de transition (le seuil de taille détermine les régions qui sont considérées comme appartenant à des zones de transition). Nous avons obtenu de bons résultats pour des pourcentages d'entre 10 % et 30 % de couverture de l'image par des zones de transition, ce qui correspond à des valeurs de λ très différentes selon les images.

Calcul du gradient Le gradient utilisé pour la LPE est un gradient couleur, calculé comme :

$$grad(x) = \max_{y \in B(x)} d(x, y) \tag{6.5}$$

où d est une distance couleur (en général la même que celle utilisée pour la détection des zones plates) et $B(x)$ est l'élément structurant de taille 1 définissant le voisinage du pixel x. Dans le même esprit que pour les zones λ-plates, où une différence entre pixels voisins de moins de λ est négligée, il est également possible d'éliminer de telles variations aussi dans le gradient. Ainsi, le λ-gradient est défini comme :

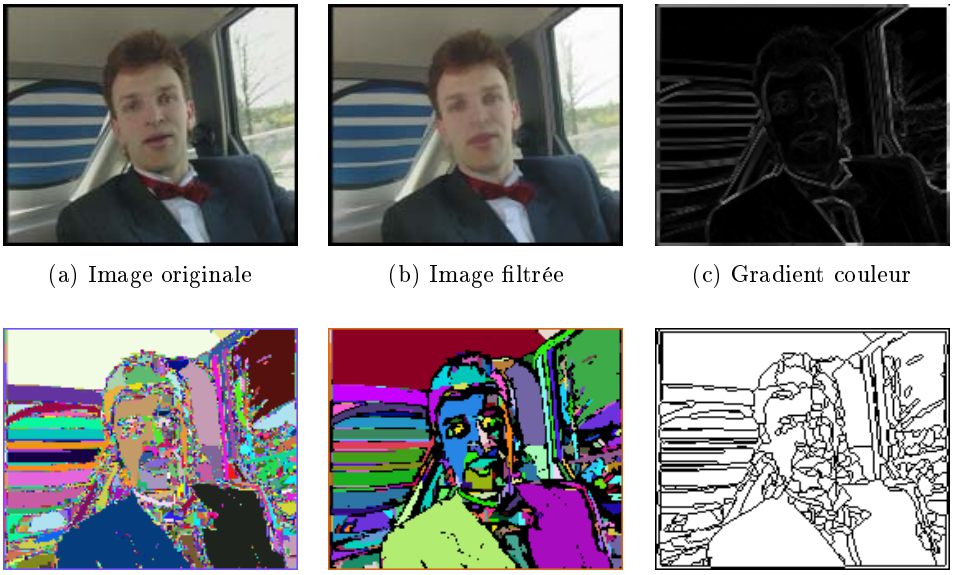
$$\lambda\text{-grad(x)} = \begin{cases} grad(x) & \text{si } grad(x) > \lambda, \\ 0 & \text{autrement.} \end{cases} \tag{6.6}$$

L'utilisation du λ-gradient n'apporte pas d'avantage par rapport à l'utilisation du gradient couleur dans le cas d'inondation avec les zones λ-plates comme marqueurs, car de toute manière les zones λ-plates couvrent déjà les parties du gradient dont le niveau de gris est inférieur à λ. Ce gradient peut cependant être intéressant dans des applications où les zones λ-plates ne sont pas utilisées comme marqueurs, comme par exemple le cas de la construction de la hiérarchie directement à partir de la LPE du gradient.

Ligne de partage des eaux Les zones λ-plates – une fois supprimées les dites zones de transition – sont utilisées comme marqueurs pour une LPE sur l'image gradient pour obtenir la partition fine.

La figure 6.24 illustre la procédure avec une image de la séquence « Carphone ». Nous travaillons dans l'espace couleur RGB. L'image est d'abord filtrée par nivellement. Ensuite, le gradient couleur de cette image est calculé. Parallèlement, les zones plates couleur sont détectées et les zones plates trop petites éliminées (régions en noir dans l'image 6.24-(e)). Cette image est utilisée comme marqueur pour une LPE sur le gradient. La partition résultante est illustrée dans la figure 6.24-(f). La partition résultante peut être comparée avec celle obtenue par détection des bassins versants du gradient, illustrée dans la figure 6.25-(a). Les bassins versants du λ-gradient (figure 6.25-(b)) donnent une partition un peu plus adaptée au contenu de l'image, mais encore moins bonne que celle obtenue avec la méthode que nous proposons.

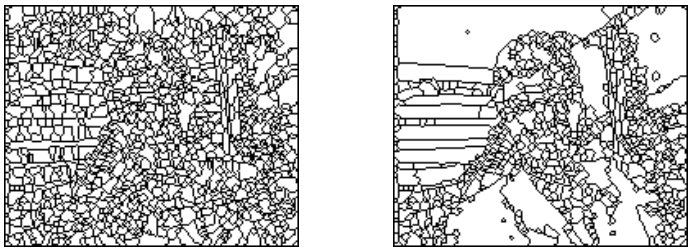
Cette méthode de calcul de la partition fine combine la bonne détection des zones couleur obtenue par les λ-zones plates avec la précision des contours extraits par la LPE. La partition fine ainsi obtenue élimine la sursegmentation produite par la LPE tout en constituant une représentation suffisamment détaillée de l'image.



(a) Image originale (b) Image filtrée (c) Gradient couleur

(d) Toutes les zones plates ($\lambda = 1,5$) (e) Zones plates > 5 pixels (f) Partition résultante

FIG. 6.24 – Obtention d'une partition fine par détection des zones plates couleur.



(a) Bassins versants du gradient (b) Bassins versants du λ-gradient

FIG. 6.25 – Partitions obtenues par inondation des bassins versants du gradient.

6.4.4 Résultats

Les figures 6.26 et 6.27 comparent les résultats obtenus avec la hiérarchie basée sur les régions de la LPE avec ceux obtenus avec la hiérarchie construite sur une partition fine calculée selon la méthode que nous venons de présenter. La partition fine calculée selon la méthode de la section 6.4.3 a beaucoup moins de régions que celle correspondant aux régions de la LPE. Les partitions de la hiérarchie construite sur notre partition fine sont également de meilleure qualité (moins de fuites) que celles de la hiérarchie construite sur les régions de la LPE.

La méthode de construction de la hiérarchie sur le graphe a plusieurs avantages par rapport à la construction directe par LPE :

- Réduction du nombre de niveaux de la hiérarchie grâce à l'utilisation d'une partition fine moins sursegmentée.
- Les structures très étroites peuvent exister dans la partition si elles ont une surface suffisante, ce qui n'est pas le cas en travaillant directement sur le gradient, comme l'illustre la figure 6.28, où la fonction de départ est formée d'une structure de seulement deux pixels d'épaisseur. On considère ici comme voisinage les deux pixels adjacents à chaque pixel. Le gradient n'arrive pas à placer un minimum à l'intérieur de la structure, ce qui la ferait disparaître après LPE, tandis que les zones plates arrivent à la détecter. Cette caractéristique est intéressante lorsqu'il y a dans l'image des objets composés de structures fines que l'on voudrait détecter.
- Grâce à la meilleure estimation de la dissimilarité entre régions sur des frontières plus longues, la qualité des résultats obtenus est aussi meilleure.

6.5 Des variantes plus robustes

Nous avons présenté une technique pour calculer des hiérarchies de partitions emboîtées adaptées au contenu de l'image même pour des niveaux de la hiérarchie avec très peu de régions. Ces hiérarchies, et en particulier celle basée sur le volume, donnent des bons résultats pour des images tout venant, mais on peut encore chercher à les améliorer. Nous présentons ici des variantes de la technique proposée, qui peuvent conduire à des meilleurs résultats dans certains cas.

Nous présentons d'abord une construction récursive de la hiérarchie. Ensuite, nous explorons la combinaison de plusieurs hiérarchies à l'aide d'algèbres d'ultramétriques.

6.5.1 Construction récursive de la hiérarchie

La hiérarchie associée à l'arbre de poids minimum choisit toujours le chemin de valeur maximale plus petite pour relier deux régions. Cette approche présente l'inconvénient de privilégier les passages de petite hauteur, quelle que soit leur dimension. Ainsi, deux régions séparées par une frontière très contrastée en moyenne mais avec une petite brèche étroite seront fusionnées tôt dans la hiérarchie, et la mesure du volume de ces régions sera faussée. Ce problème est moins important lorsque la hiérarchie est construite sur un graphe de voisinage et que la partition fine n'est pas très sursegmentée. La raison est que d'une part le graphe permet de valuer les frontières avec une moyenne du contraste, et d'autre part le fait que les régions de la partition fine soient plus grandes rend cette moyenne plus représentative du contraste réel.

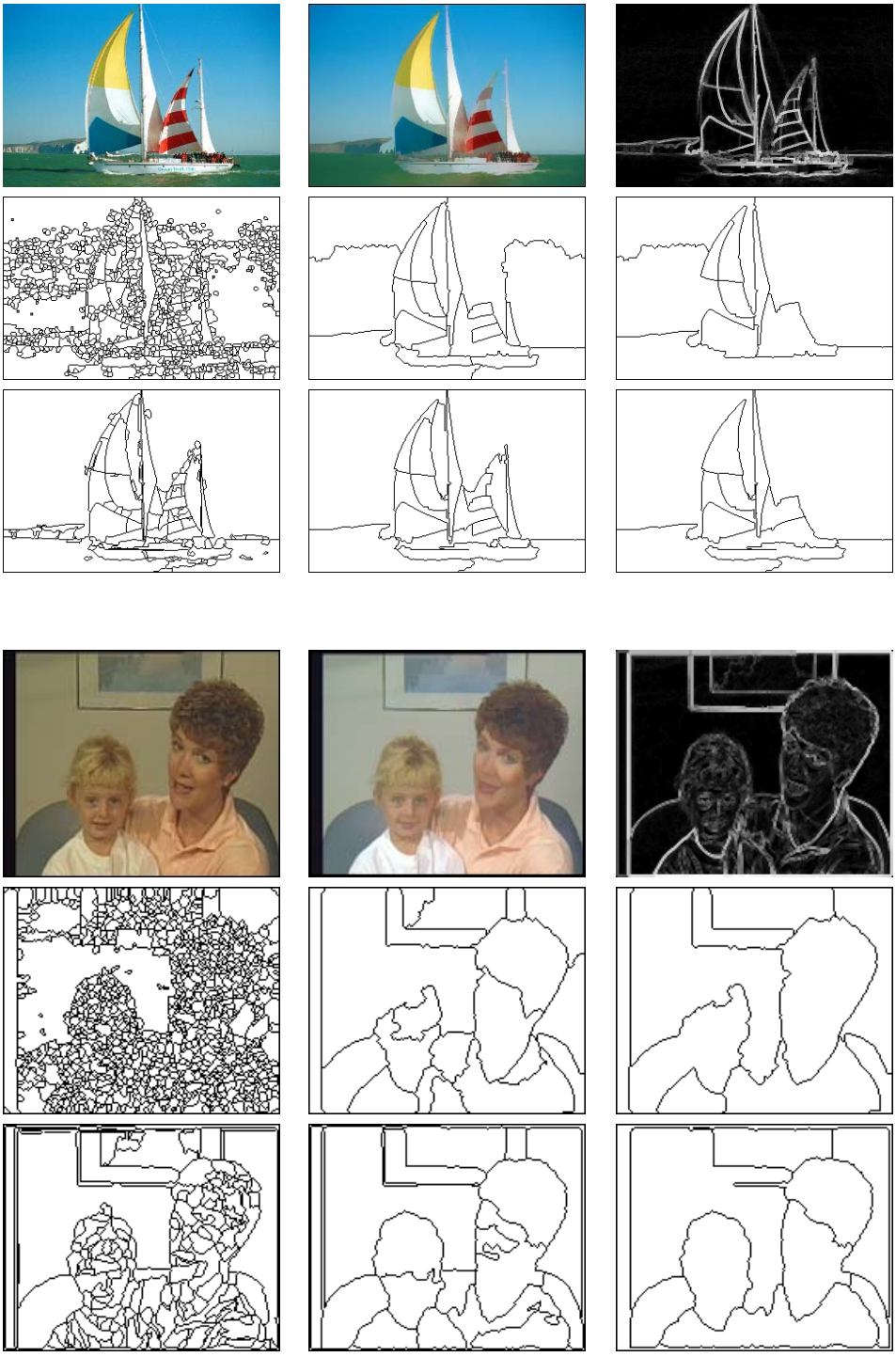


FIG. 6.26 – Résultats obtenus avec la hiérarchie calculée sur les régions de la LPE et avec celle calculée sur les régions de la partition fine proposée dans la section 6.4.3. Ligne d'en haut, de gauche à droite : image originale, image filtrée et gradient. Ligne du milieu : partition fine, segmentation volumique en 20 régions et segmentation volumique en 10 régions pour la hiérarchie calculée sur les régions de la LPE. Ligne d'en bas : idem pour la hiérarchie calculée sur la partition fine proposée dans 6.4.3.

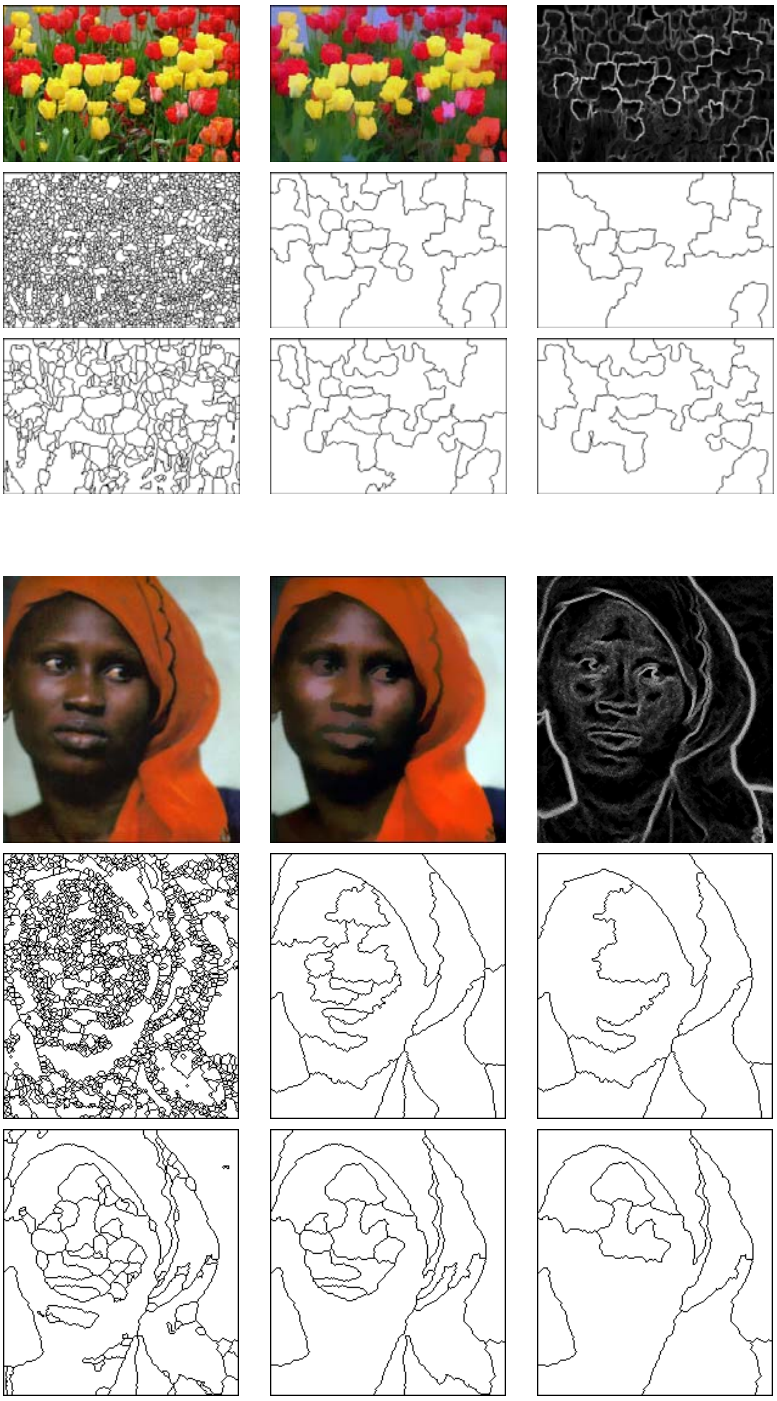
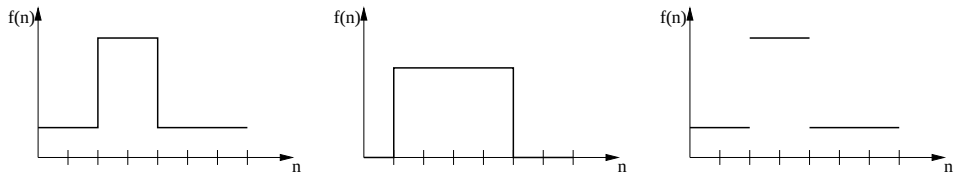


FIG. 6.27 – Résultats obtenus avec la hiérarchie calculée sur les régions de la LPE et avec celle calculée sur les régions de la partition fine proposée dans la section 6.4.3. Ligne d'en haut, de gauche à droite : image originale, image filtrée et gradient. Ligne du milieu : partition fine, segmentation volumique en 20 régions et segmentation volumique en 10 régions pour la hiérarchie calculée sur les régions de la LPE. Ligne d'en bas : idem pour la hiérarchie calculée sur la partition fine proposée dans 6.4.3.



(a) Fonction de départ (b) Gradient (c) Zones plates

FIG. 6.28 – Problème de résolution du gradient pour les structures fines, résolu en utilisant des zones plates comme marqueurs.

La robustesse de la méthode peut cependant être encore améliorée en faisant des réévaluations systématiques du contraste des frontières des régions au fur et à mesure que la construction de la hiérarchie progresse, selon l'esprit de ce qui est fait dans [60], [43]. L'idée à la base de ces réévaluations est le fait que les frontières se font de plus en plus longues au fur et à mesure que des fusions entre régions se produisent. Ainsi, pour des niveaux élevés de la hiérarchie les informations de contraste dont on dispose sont plus fiables et devraient empêcher des fusions à travers des passages étroits.

Tandis que pour la construction non récursive les mesures de dissimilarité locale entre régions voisines sont constantes tout au long du processus, pour la construction récursive il est nécessaire de réaliser des réévaluations de ces mesures à chaque fusion. Ainsi, pour l'approche récursive, à chaque fois qu'une fusion entre deux régions se produit, il est nécessaire d'actualiser les valeurs de toutes les frontières qui ont changé, c'est-à-dire, les frontières entre la nouvelle région fusionnée et toutes ses voisines. Cette actualisation ajoute un coût d'actualisation au calcul de la hiérarchie.

L'algorithme de calcul est le suivant :

1. Construction du graphe de voisinage associé à la partition fine. Sur ce graphe chaque arête doit contenir deux informations : le contraste local avec la région voisine et la longueur de la frontière représentée par l'arête. L'information sur la longueur servira à l'actualisation.
2. Ajout de l'arête de valeur plus faible à l'arbre, seulement si elle n'introduit pas de cycle. Comme pour la construction non récursive, le volume (ou mesure géométrique choisie) est calculé et le plus petit est attribué à l'arête correspondante sur l'arbre.
3. L'ajout d'une arête produit la fusion de deux composantes connexes du graphe, ce qui oblige à une actualisation des mesures de dissimilarité. Toutes les arêtes reliant un nœud appartenant à l'une des composantes connexes fusionnées et un nœud n'y appartenant pas doivent être actualisées. Pour cela, il est nécessaire de recalculer la mesure de dissimilarité, ce qui est fait en fonction du contraste local et de la longueur des arêtes.
4. Puisque les valeurs des arêtes ont changé, après chaque fusion une nouvelle classification des arêtes est nécessaire pour déterminer celle de valeur plus faible, qui sera la prochaine à ajouter. Retour au point 2 jusqu'à ce que toutes les arêtes aient été traitées.

La classification des arêtes à chaque fusion peut être évitée en utilisant une file d'attente hiérarchique. Les arêtes du graphe sont initialement introduites dans la file d'attente avec priorité donnée par la valeur d'arête. Quand une arête est réactualisée, elle est réintroduite dans la file d'attente correspondante à sa nouvelle priorité. Ainsi, une même arête peut sortir plusieurs fois de la file d'attente. Pour savoir si elle correspond à l'état actuel du graphe, il

suffit de comparer la priorité en cours de la file avec la valeur de l'arête sur le graphe. Si ces deux valeurs correspondent, l'arête est traitée. Autrement, elle est ignorée.

Ce type de construction de la hiérarchie pose cependant le problème suivant : une région est toujours fusionnée à travers le passage le plus bas au long de sa frontière. C'est-à-dire que c'est la hauteur du passage le plus bas qui détermine – par rapport à la valeur des autres frontières – l'ordre de fusion de la région. Quand une région est fusionnée avec une région voisine, ses frontières sont réévaluées avec la moyenne des valeurs des frontières entourant la nouvelle région. De cette manière, la hauteur du passage le plus bas au long de sa frontière augmente (puisqu'on fait la moyenne avec des valeurs plus fortes). Cette augmentation implique un changement dans l'ordre de fusion des régions : notre région sera en effet fusionnée plus tard, et d'autres régions passeront à être plus prioritaires. On voit bien que les frontières qui n'ont pas encore été réévaluées auront plus de chances d'être fusionnées, pour des valeurs similaires des frontières. En particulier, des régions reliées par des passages étroits (qui ne peuvent donc pas être réévalués, puisque leur frontière est constituée d'un unique morceaux) sont fusionnées avant les régions séparées par de longues frontières pour lesquelles une réévaluation a été possible. Or cet effet est contraire à ce que nous recherchons ! Une solution consiste à pénaliser les frontières courtes de façon à compenser cet effet. La pénalisation est faite en ajoutant à la valeur de la frontière (dissimilarité couleur) un terme proportionnel à $longueur\_totale/longueur\_frontière$, où $longueur\_frontière$ est la longueur de la frontière séparant les deux régions et $longueur\_totale$ est le périmètre des deux régions après fusion.

La figure 6.29 compare les résultats obtenus pour l'image « Lena » pour la hiérarchie sans construction récursive, avec construction récursive sans pénalisation des frontières courtes et avec pénalisation des passages étroits. On remarque que la construction récursive sans pénalisation des passages étroits a tendance à produire des régions aux formes imbriquées (région du chapeau). Ce problème est résolu par la construction avec pénalisation des frontières courtes, avec laquelle les régions obtenues ont des formes plus régulières. Par rapport à la construction non récursive, les résultats obtenus sont plus adaptés au contenu de l'image.

6.5.2 Hiérarchies obtenues par algèbre d'ultramétriques

Dans le chapitre 3, nous avons vu que différentes ultramétriques définies sur un même ensemble de données peuvent être combinées pour donner lieu à de nouvelles ultramétriques. En particulier nous avons parlé du sup et de l'inf d'ultramétriques. Nous avons vu que le sup d'ultramétriques donne lieu à une nouvelle ultramétrique, tandis que l'inf ne l'est malheureusement pas, mais il est tout de même possible de construire à partir de l'indice de distance donné par l'inf l'ultramétrique inférieure maxima, appelée aussi ultramétrique sous-dominante.

Considérons une série de hiérarchies H_i définies sur un même ensemble de données, c'est-à-dire, sur une même partition fine. Ces hiérarchies sont représentées par des arbres aux arêtes valuées T_i. Ces arbres sont différents et n'ont pas nécessairement la même structure, mais ils ont tous le même nombre de régions et le même nombre d'arêtes. Par simplicité, nous supposons que les arêtes de l'arbre ont toutes des valuations différentes, ce qui équivaut à dire que seulement deux régions fusionnent pour chaque niveau de la hiérarchie. Dans ce cas, le nombre de niveaux de toutes les hiérarchies est exactement le même. Les ultramétriques données par les valeurs des arêtes sont des fonctions croissantes de l'ordre de fusion des régions. Afin d'obtenir des résultats des opérations intéressantes, il est nécessaire de mettre les différentes ultramétriques à l'échelle. En effet, si l'une d'entre elles était trop au dessus des autres, le sup donnerait cette même ultramétrique. Pour réaliser ce changement d'échelle il suffit de

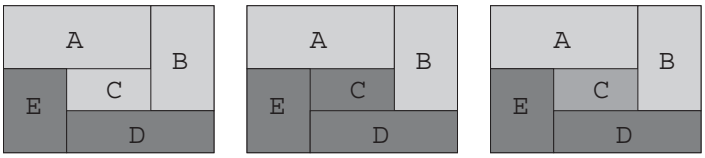
FIG. 6.29 – Hiérarchies obtenues avec et sans construction récursive de la hiérarchie. De haut en bas, première ligne : image originale, image filtrée et partition fine ; deuxième ligne : niveaux de la hiérarchie sans construction récursive avec 30, 20 et 10 régions ; troisième ligne : mêmes trois niveaux de la hiérarchie avec construction récursive, quatrième ligne : mêmes trois niveaux de la hiérarchie pour la construction récursive de la hiérarchie avec pénalisation des passages étroits.

réévaluer consécutivement les arêtes des arbres T_i par ordre croissant.

6.5.2.1 Partitions associées

Une fois les différentes hiérarchies mises à l'échelle, regardons comment le sup et l'inf agissent du point de vue des partitions concernées. Le sup considère que la distance entre toute paire d'éléments (régions) est donnée par le maximum des distances entre ces deux éléments dans toutes les hiérarchies. Considérons le niveau de chacune des hiérarchies contenant seulement deux régions. A chacune des hiérarchies correspond une partition différente. Puisque les hiérarchies H_i ont été mises à l'échelle, pour chacune d'entre elles la distance entre deux nœuds appartenant à chacune des régions différentes de la partition est $N-1$, où N est le nombre des niveaux des hiérarchies H_i. La distance entre deux nœuds contenus à l'intérieur de chacune des régions est plus petite que $N-1$. En faisant le sup des distances, on obtient que la distance entre deux nœuds est de $N-1$ si elle l'était sur une des hiérarchies. En conséquence, la partition correspondante au niveau $N-1$ de la hiérarchie est constituée de l'intersection des partitions correspondant à ce niveau pour chacune des hiérarchies. Ceci n'est pas seulement vrai pour le niveau $N-1$ de la hiérarchie, mais pour tous les niveaux.

La figure 6.30 l'illustre avec un exemple, où l'on dispose de deux hiérarchies H_1 et H_2 définies sur une même partition fine, formée de 5 régions. Si l'on suppose qu'à chaque niveau de la hiérarchie seulement deux régions peuvent fusionner, alors les hiérarchies H_1 et H_2 ont toutes les deux 5 niveaux, le niveau 0 correspondant à la partition fine, le niveau 4 correspondant à une seule région contenant toute l'image. Les images (a) et (b) de la figure 6.30 montrent les partitions associées au niveau 3 des hiérarchies H_1 et H_2. Puisque la distance ultramétrique correspond à l'indice de stratification sur la hiérarchie, nous connaissons pour les hiérarchies H_1 et H_2 les régions qui sont à distance 3 entre elles, données par les couples de régions qui sont séparées pour le niveau 3 de la hiérarchie. Pour les régions qui n'ont pas été séparées, on sait que la distance entre elles est plus petite que 3. La table 6.1 résume ces informations. Le sup élément à élément des tables correspondantes aux hiérarchies H_1 et H_2 donne la partition correspondante de la nouvelle hiérarchie. On remarque que la région C est maintenant séparée pour ce niveau, puisqu'elle se trouve à distance 3 de toutes les autres régions. Les régions A et B se trouvent encore à distance inférieure à 3 entre elles, de même que les régions D et E.



(a) H_1 (b) H_2 (c) $Sup(H_1, H_2)$

FIG. 6.30 – Intersection de partitions associée au sup de distances ultramétriques.

La construction de l'ultramétrique sous-dominante associée à l'indice de distance donné par l'inf a l'effet contraire. Si l'on considère un niveau avec n régions sur chacune des hiérarchies, la distance entre les régions appartenant à des classes différentes est égale à $N-n+1$, et toutes les autres distances sont plus petites. En prenant l'inf des distances, deux régions ne seront à distance $N-n+1$ que si elles le sont pour toutes les hiérarchies H_i. Ainsi, la partition associée au niveau n résulte en union des régions des partitions, comme le montre la figure 6.31.

	A	B	C	D	E
A	0	< 3	< 3	3	3
B	< 3	0	< 3	3	3
C	< 3	< 3	0	3	3
D	3	3	3	0	< 3
E	3	3	3	< 3	0

(a) H_1

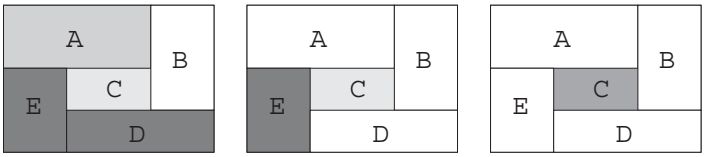
	A	B	C	D	E
A	0	<3	3	3	3
B	<3	0	3	3	3
C	3	3	0	<3	<3
D	3	3	<3	0	<3
E	3	3	<3	<3	0

(b) H_2

	A	B	C	D	E
A	0	<3	3	3	3
B	<3	0	3	3	3
C	3	3	0	3	3
D	3	3	3	0	<3
E	3	3	3	<3	0

(c) $Sup(H_1, H_2)$

TAB. 6.1 – Distances ultramétriques associées aux partitions de la figure 6.30.



(a) H_1 (b) H_2 (c) $Inf(H_1, h_2)$

FIG. 6.31 – Union de partitions associée à l'inf de distances ultramétriques.

6.5.2.2 Algorithme de calcul

Le calcul de nouvelles hiérarchies en utilisant des algèbres passe par la construction d'une table contenant les nouvelles distances entre paires d'éléments. Une table de distances a une structure d'arbre si elle satisfait la condition ultramétrique : $\forall a, b, c\ d(a, c) \leq Max(d(a, b), d(b, c))$. Le sup de distances ultramétriques satisfait cette condition. Dans ce cas, il est possible de trouver l'arbre pour lequel les distances entre éléments correspondent exactement aux distances exprimées par la table de distances. Cependant, dans un cas plus général, et en particulier dans le cas de l'inf, la table de distances résultante ne satisfait pas la condition ultramétrique. Dans ce cas, il est nécessaire de construire une ultramétrique à partir des indices de distance donnés par la table, et il existe diverses manières de la réaliser. La minimisation de l'erreur entre les distances exprimées par la table et celles exprimées par l'arbre semble être la solution optimale, mais elle constitue un problème complexe. Hartigan [28] propose une méthode heuristique pour en obtenir une approximation moyennant des opérations locales sur l'arbre. De notre côté, nous avons décidé d'utiliser l'ultramétrique sous dominante, car elle est simple à calculer et s'adapte bien au cas de l'inf.

Étant données deux hiérarchies représentées par les arbres aux arêtes valuées T_1 et T_2, la génération d'une nouvelle hiérarchie à partir de T_1 et T_2 se fait selon les étapes suivantes :

- création des tables de distances associées aux arbres T_1 et T_2 ;
- combinaison des valeurs des tables (par sup, inf ou autre), élément à élément ;
- création de l'arbre exprimant la nouvelle hiérarchie.

Création de la table de distances associée à un arbre Pour créer la table de distances associée à l'arbre, on se base sur le fait que la distance ultramétrique définie par l'arbre entre toute paire de nœuds est donnée par la valeur de l'arête la plus forte sur le chemin reliant les deux nœuds. Les arêtes sont initialement supprimées et ordonnées par ordre croissant de valeurs. Ensuite, les arêtes sont ajoutées à l'arbre une à une. Chaque arête ajoutée produit une fusion entre deux composantes connexes. Puisque les arêtes ont été ajoutées par ordre croissant de valeurs, on connaît à ce moment la distance ultramétrique (valeur de l'arête la plus forte) entre chacun des nœuds appartenant à l'une et à l'autre composante connexe. Ces

distances sont inscrites sur la table en parcourant les deux composantes connexes moyennant un algorithme d'étiquetage. L'algorithme 6.1 donne une description détaillée.

Algorithme 6.1 Algorithme de génération de la table de distances associée à un arbre.

```
T : Arbre aux arêtes valuées
table : Table de distances entre éléments
Supprimer toutes les arêtes de l'arbre T.
pile ← Ordonner les arêtes par ordre croissant de valeurs.
while NoVide(pile) do
  e_ij ← ExtraireArête(pile)
  valeur ← Valeur(e_ij)
  v_i, v_j ← TrouverExtremitésArête(T, e_ij)
  for v_k appartenant à la même composante connexe que v_i do
    for v_l appartenant à la même composante connexe que v_j do
      table[v_k][v_l] ← valeur
    end for
  end for
  AjouterArête(T, e_ij)
end while
```

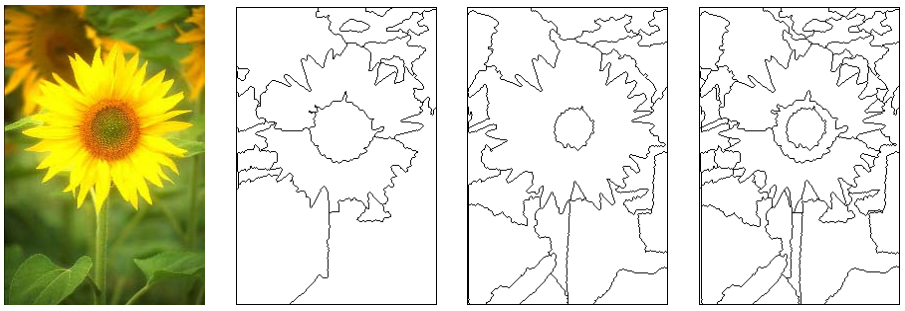
Création de l'arbre de poids minimum associé à une table de distances Cette opération revient à trouver l'ultramétrique sous-dominante et consiste à calculer l'arbre de poids minimum associé à un ensemble de données pour lesquelles on connaît les distances entre toute paire d'éléments. Les algorithmes de calcul de l'arbre de poids minimum permettent de le faire. Néanmoins, dans notre cas nous avons une contrainte additionnelle à respecter, donnée par les relations de voisinage sur la partition fine. Pour les respecter, au lieu de calculer l'arbre à partir de la table de distances, nous ajoutons une étape intermédiaire consistant à réévaluer les arêtes du graphe de voisinage avec les distances données par cette table. Les distances correspondant à des nœuds n'étant pas reliés par une arête sont ignorées. Puis, l'arbre de poids minimum du graphe de voisinage est calculé, en suivant un des nombreux algorithmes existants [21].

6.5.2.3 Exemple d'application et résultats

Cette méthode peut être appliquée quand différentes hiérarchies sont disponibles pour une même image. Ici, nous l'avons appliquée à la construction de hiérarchies associées à des images couleur. Des hiérarchies volumiques ont été calculées sur les composantes H et V de l'image dans l'espace HSV. Ensuite, les deux hiérarchies ont été combinées par sup et inf. Afin de pouvoir combiner les hiérarchies, il est nécessaire de disposer d'une partition fine commune à toutes les hiérarchies. Le calcul de cette partition est fait en suivant la méthode basée sur les zones plates, décrite dans la section 6.4.3. Les graphes de voisinage associés à la partition

fine sont calculés pour chacune des composantes H et V. Les arêtes des graphes sont valuées avec la différence des moyennes des valeurs des régions. Ainsi, on obtient deux graphes ayant la même structure mais avec des arêtes valuées différemment. Les hiérarchies volumiques sont alors calculées sur chacun des graphes. Ces hiérarchies sont combinées selon la méthode que nous venons de décrire.

Les images (b) et (c) de la figure 6.32 montrent le niveau des hiérarchies sur H et V avec 20 régions. Le niveau correspondant sur le sup de hiérarchies est illustré dans la figure 6.32-(d) et correspond à l'intersection des deux partitions précédentes. Cette partition contient 58 régions. Néanmoins, il est plus intéressant de comparer des partitions contenant le même nombre de régions, et nous montrons pour cette raison dans la figure 6.32-(e) le niveau sur le sup des hiérarchies contenant seulement 20 régions, ainsi que les niveaux des hiérarchies sur H et V avec 58 régions (images (f) et (g) de la figure 6.32). La figure 6.32-(h) montre le niveau correspondant aux partitions (b) et (c) sur l'inf des hiérarchies, qui contient 10 régions.



(a) Image (b) H - 20 rég. (c) V - 20 rég. (d) Sup - 58 rég.

(e) Sup - 20 rég. (f) H - 58 rég. (g) V - 58 rég. (h) Inf - 10 rég.

FIG. 6.32 – Résultats obtenus par sup et inf de hiérarchies sur les composantes couleur H et V dans l'espace HSV.

Les résultats obtenus pour le sup des hiérarchies sont intéressants pour la partition en 58 régions. En effet, en comparant avec les hiérarchies obtenues sur H et V, on obtient une meilleure segmentation de la fleur - moins de régions et plus significatives –. Mais ce n'est pas le cas quand on compare les niveaux des hiérarchies avec 20 régions, où la fleur n'est même pas séparée du fond. On trouve cependant des petites régions sans importance, qui ont été classées aux niveaux élevés de la hiérarchie. Ce problème se produit de manière générale et est causé par l'existence dans la partition fine de petites régions qui sont classées différemment par les deux hiérarchies. Pour expliquer ce phénomène, considérons la construction de deux hiérarchies surfaciques H_1 et H_2 sur la partition de la figure 6.33. La hiérarchie H_1 décide de

fusionner la région C avec la région B. Comme C a une petite surface, on a $d(C,B)$ petit. Mais puisque $B \cup C$ a une grande surface, et que A a aussi une grande surface, on a, pour H_1, $d(C,A) = d(B,A)$ grand. Maintenant, si H_2 considère que C doit être fusionnée avec A, on a, en suivant le même raisonnement, $d(A,C)$ petit et $d(C,B)$ grand. En faisant le sup des distances, on a $d(C,A)$ grand et $d(C,B)$ grand, et en conséquence C est éloignée de tous ses voisins, et est classée aux niveaux élevés de la hiérarchie donnée par le sup. Dans le cas de l'inf, on a précisément l'effet inverse, et la hiérarchie résultante donnera $d(C,A)$ petit et $d(C,B)$ petit. Ainsi, on crée à travers la région C un passage à faible coût entre deux régions, A et B, qui devraient être séparées pour des niveaux élevés de la hiérarchie.

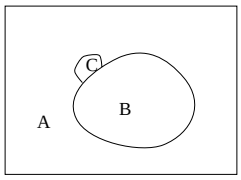
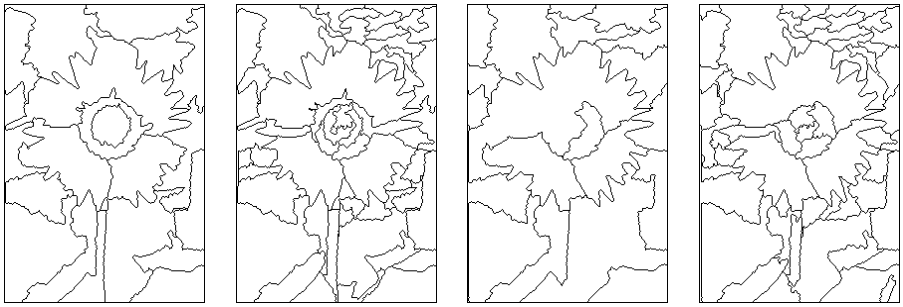


FIG. 6.33 – Partition donnant lieu à de possibles problèmes.

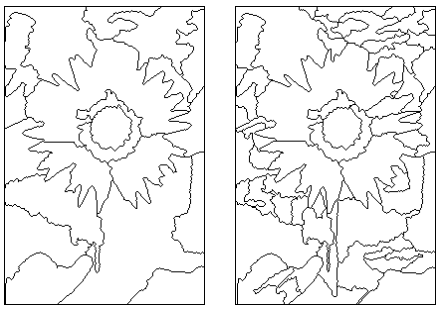
Une solution à ces problèmes consiste à reconstruire, sur l'arbre représentant la nouvelle distance ultramétrique, une hiérarchie volumique. Pour cela, cet arbre est maintenant interprété comme s'il s'agissait d'un graphe de voisinage et les valeurs des arêtes sont interprétées comme la mesure de dissimilarité. Un nouveau arbre est alors calculé par inondation, de la manière décrite dans la section 6.4.2. Les ultramétriques entre régions sont ainsi modifiées pour tenir compte de la taille des régions, ce qui évite que des petites régions apparaissent aux niveaux élevés de la hiérarchie. Les résultats sont présentés dans la figure 6.34. On remarque la meilleure qualité de la segmentation avec 20 régions obtenue pour la hiérarchie volumique calculée sur le sup, par rapport à celles obtenues pour les composantes H et V (figures 6.32-(b) et (c)). La fleur est mieux segmentée dans ses régions significatives, tandis que le fond est moins sursegmenté. Les résultats obtenus avec l'inf s'améliorent aussi après calcul de la hiérarchie volumique, mais ils ne sont pas meilleurs que ceux donnés par les hiérarchies sur les composantes H et V. En effet, l'inf prenant toujours la distance la plus petite entre régions donnée par les différentes hiérarchies, si seulement l'une d'entre elles produit une erreur et classe la région comme étant peu contrastée, cette région sera aussi classée comme étant peu importante dans la hiérarchie finale. Afin de comparer les résultats, dans les figures 6.34-(e) et (f) nous présentons les résultats obtenus si la hiérarchie est calculée comme nous l'avons décrit dans la section 6.4, où les trois composantes sont combinées en utilisant la distance euclidienne dans l'espace couleur RGB. Les résultats sont très similaires à ceux obtenus pour le sup, mais on remarque cependant la meilleure extraction de la tige de la fleur pour le cas du sup.

6.6 Introduction d'information *a priori*

Contrairement aux méthodes classiques de segmentation morphologique avec marqueurs, la technique de construction d'une hiérarchie de partitions emboîtées que nous avons proposée dans les sections précédentes est générale et ne fait pas de suppositions sur le contenu des images. Le seul critère utilisé pour déterminer l'importance des régions est leur rapport entre taille et contraste (leur volume), ce qui peut être considéré comme une évaluation de leur



(a) Sup - 20 rég. (b) Sup - 58 rég. (c) Inf - 20 rég. (d) Inf - 58 rég.



(e) Vol - 20 rég. (f) Vol - 58 rég.

FIG. 6.34 – Résultats obtenus en calculant la hiérarchie volumique sur les hiérarchies par sup et inf.

importance visuelle. Pour des applications où les objets d'intérêt ne correspondent pas à des régions ayant un grand volume, de tels objets ne se trouveront pas sur les niveaux élevés de la hiérarchie, mais il sera toutefois possible de les extraire en descendant dans la hiérarchie. Cela se fera en échange d'un peu plus de travail de la part de l'utilisateur.

Dans les cas où l'on dispose d'informations sur les caractéristiques des images, il est intéressant de pouvoir les utiliser pour réduire la quantité d'interaction nécessaire tout en gardant une approche hiérarchique. Ceci est fait en plaçant les objets d'intérêt en haut de la hiérarchie. Dans cette section nous présentons deux manières d'introduire des informations *a priori* sur les images. La première consiste à combiner plusieurs hiérarchies en une seule selon les caractéristiques des objets à détecter (d'une manière un peu différente à celle présentée dans la section 6.5.2). La deuxième méthode, basée sur la définition de marqueurs qu'on appelle « flous », offre un cadre plus général permettant de placer sélectivement en haut de la hiérarchie certaines régions.

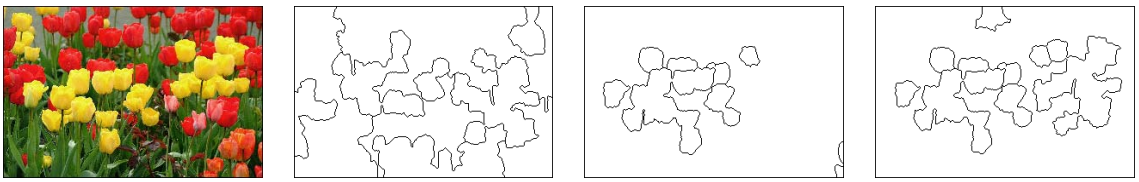
6.6.1 Hiérarchies mixtes moyennant des distances lexicographiques

Nous avons présenté dans la section 6.3.2.2 quelques ultramétriques basées sur l'inondation synchrone permettant d'obtenir des hiérarchies significatives. Parmi elles, celle basée sur le volume donne les meilleurs résultats dans un cadre général, puisqu'elle représente un compromis entre taille et contraste des régions choisies. C'est un critère particulièrement intéressant lorsque l'on cherche à détecter des objets sur le premier plan de l'image. Dans le même esprit, la dynamique est bien adaptée aux images où les objets à détecter sont petits et contrastés.

Il existe cependant des images pour lesquelles un seul de ces critères ne suffit pas à produire une bonne représentation de l'image, et il est intéressant dans ces cas de pouvoir combiner différents critères. Nous considérons ici la combinaison lexicographique de distances, que nous avons présentée dans le chapitre 3, section 3.4.4.

Remarquons que pour que l'ordre lexicographique ait une utilité pratique, il est nécessaire que les arbres des hiérarchies utilisées comme chiffres plus significatifs de la lexicographie aient plusieurs arêtes avec la même valeur. Autrement dit, les premières hiérarchies doivent être plus grossières que les suivantes pour que l'ordre lexicographique ait une influence sur le résultat. Ici nous considérons un ensemble de hiérarchies H_n de plus en plus fines, c'est-à-dire avec de plus en plus de niveaux. De la même manière que pour le sup et inf d'ultramétriques, une table de distances entre éléments peut être construite pour chacune des hiérarchies. À partir de cette table, une nouvelle table est construite, contenant pour chaque couple d'éléments un vecteur, composé des distances ultramétriques entre ces éléments dans les différentes hiérarchies. Ces vecteurs sont alors utilisés pour valuer les arêtes du graphe de voisinage représentant la partition fine, et l'arbre de poids minimum est construit à partir de l'ordre lexicographique. Rappelons en effet que les lexicographies d'ultramétriques ne constituent pas une ultramétrique, mais qu'en prenant la lexicographie comme indice de distance, il est possible de la transformer en ultramétrique. En particulier, en calculant l'arbre de poids minimum à partir de l'indice lexicographique nous obtenons l'ultramétrique sous-dominante.

La figure 6.35 compare les résultats obtenus avec les critères de volume et de contraste local à ceux obtenus en utilisant une distance lexicographique combinant ces deux critères. Pour construire la distance lexicographique, le logarithme de la distance volumique a été divisé en 5 tranches uniformes, de façon à réduire la résolution de cette hiérarchie, et utilisée comme premier chiffre de la distance lexicographique. Le deuxième chiffre est donné par la distance basée sur le contraste. On voit dans la figure 6.35-(b) que la hiérarchie volumique essaie de diviser l'image en régions de grande surface, tandis que les objets d'intérêt – les fleurs – sont plutôt petits et contrastés. En même temps, le contraste ne prenant pas en compte la taille des régions, la partition résultante a trop tendance à privilégier les petites régions contrastées. On obtient de meilleurs résultats avec la hiérarchie lexicographique qui, elle, choisit les régions les plus contrastées parmi celles ayant un volume suffisant.

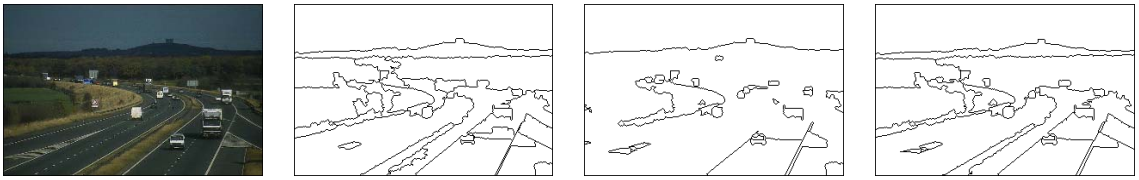


(a) Image (b) Volume (c) Contraste (d) Lexicographique

FIG. 6.35 – Résultats obtenus pour 10 régions avec les hiérarchies basées sur le contraste et le volume, et en utilisant des distances lexicographiques.

Un cas typique où ce type de distance peut être particulièrement utile c'est l'analyse de scènes routières. En effet, l'analyse de ce type d'images implique la détection des voitures, qui constituent des régions assez contrastées mais de petite taille. La hiérarchie volumique n'est donc pas bien adaptée puisqu'elle classera les voitures dans les niveaux les plus bas de la hiérarchie. Cependant, la hiérarchie basée sur le contraste n'est pas non plus bien adaptée,

parce qu'elle ne saura pas détecter la chaussée convenablement. La distance lexicographique permet donc de combiner les deux afin d'obtenir des résultats mieux adaptés à la scène. La figure 6.36 présente un exemple. La hiérarchie volumique, divisée en 3 tranches, a été utilisée comme chiffre plus significatif. Comme deuxième chiffre nous avons pris la hiérarchie donnée par le contraste. On constate que la hiérarchie lexicographique arrive à détecter plus de voitures que la hiérarchie volumique, tout en donnant un bon aperçu de la composition de la scène, ce qui n'est pas le cas de la hiérarchie basée sur le contraste.



(a) Image originale (b) Volume (c) Contraste (d) Lexicographique

FIG. 6.36 – Résultats obtenus pour 30 régions avec les hiérarchies basées sur le contraste et le volume, et en utilisant des distances lexicographiques.

6.6.2 Les marqueurs flous

La segmentation morphologique traditionnelle à partir de marqueurs a démontré sa puissance dans des nombreuses applications où des informations *a priori* sont disponibles sur les images à traiter. Or, dans la méthode traditionnelle, une segmentation par marqueurs donne une segmentation unique. Si cette segmentation n'est pas totalement satisfaisante, il faut introduire de nouveaux marqueurs et recommencer le processus. De plus, la segmentation à partir de marqueurs contient exactement autant de régions que de marqueurs. Une hiérarchie de partitions, au contraire, permet d'expérimenter avec différents niveaux de résolution et de composer des partitions différentes à partir d'une seule inondation, ce qui la rend très intéressante pour la segmentation interactive.

Les marqueurs flous [57] combinent les avantages des marqueurs morphologiques avec ceux des hiérarchies. Ils permettent en effet de désigner certaines régions comme étant plus importantes sans préjudice au calcul de la hiérarchie complète.

6.6.2.1 Définition

Considérons l'inondation synchrone, telle que nous l'avons présentée dans la section 6.3.2. La mesure des temps pris par les lacs pour se remplir produit une hiérarchisation des minima, et en conséquence de la partition fine résultante après inondation. Considérons maintenant que chaque minimum est marqué avec une valeur entre 0 et 1. Nous appellerons ces valeurs *marqueurs flous*, l'adjectif « flou » venant du fait que ces marqueurs ne détermineront pas exactement la position des régions dans la hiérarchie, mais donnent seulement une indication de l'importance du minimum par rapport aux autres. Ces marqueurs sont utilisés pour modifier la vitesse de croissance des lacs comme $nouvelle\_vitesse = vitesse\_initiale \cdot valeur\_marqueur\_flou$. Ainsi, les lacs ayant un marqueur plus petit seront ralentis par rapport à ceux ayant un marqueur plus grand et se rempliront plus tard. Il seront donc classés à des niveaux plus élevés de la hiérarchie. Les cas extrêmes correspondent aux

marqueurs de valeurs 1 et 0. Pour une valeur du marqueur flou de 1, la vitesse de croissance n'est pas altérée. Pour une valeur de 0, le lac n'arrive jamais à se remplir, et reste vide. Entretemps, il absorbe ses lacs voisins. Tous les lacs ayant une valeur de marqueur 0 seront ainsi classés au niveau plus haut de la hiérarchie, car le temps nécessaire pour leur remplissage est infini. On appelle les marqueurs avec une valeur 0 des *marqueurs forts*. Pour cette raison, deux lacs ayant un marqueur 0 ne peuvent jamais se rencontrer, et le niveau le plus haut de la hiérarchie ne contiendra plus une unique région, mais autant de régions qu'il y a des marqueurs avec une valeur 0. La valeur 0 correspond, pour le niveau plus haut de la hiérarchie, à la situation traditionnelle de segmentation morphologique à partir de marqueurs, avec la différence que chacune des régions résultantes n'est que le sommet d'un arbre représentant une hiérarchie complète correspondant à cette région. Entre les valeurs de 0 et 1 il y a un continuum de marqueurs possibles permettant d'ajuster avec précision la nouvelle hiérarchie.

6.6.2.2 Algorithme de calcul

Algorithme de base L'algorithme de calcul pour produire la hiérarchie avec les marqueurs flous peut être obtenu en modifiant légèrement celui utilisé pour calculer la hiérarchie donnée par l'inondation synchrone à partir de l'inondation uniforme (section 6.3.5). En effet, pour le calcul de la hiérarchie donnée par l'inondation synchrone à partir de l'inondation uniforme, une arête est ajoutée à l'arbre à chaque fois que, pendant l'inondation uniforme, une rencontre entre deux lacs se produit. La valeur de l'arête doit être proportionnelle au temps nécessaire pour que le plus petit lac se remplisse pendant l'inondation synchrone. Nous avons vu dans la section 6.3.5 que le temps s'exprime comme $temps = mesure/vitesse$, où $mesure$ est la mesure du lac le plus petit et $vitesse$ est la vitesse de croissance des lacs (constante pour tous les lacs). Comme la $vitesse$ est une valeur constante, il suffit de valuer l'arête avec la $mesure$ du lac le plus petit au moment de la rencontre. Le calcul de la hiérarchie pour le cas des marqueurs flous est similaire. Quand deux lacs se rejoignent, une arête est ajoutée à l'arbre, mais la valeur de l'arête doit maintenant être calculée différemment, car la vitesse des lacs n'est pas la même pour tous les lacs. Elle dépend en effet de la valeur du marqueur flou. Ainsi, la nouvelle vitesse associée à chaque lac est : $nouvelle\_vitesse = vitesse\_initiale \cdot valeur\_marqueur\_flou$, où la $vitesse\_initiale$ correspond à la vitesse de croissance des lacs quand il n'y a pas de marqueurs flous. Maintenant, le temps nécessaire pour le remplissage d'un lac s'exprime comme :

$$temps = \frac{mesure}{nouvelle\_vitesse} = \frac{mesure}{vitesse\_initiale \cdot valeur\_marqueur\_flou}$$

Comme la $vitesse\_initiale$ est la même pour tous les lacs, la valeur qu'il faut comparer comme étant proportionnelle au temps est $mesure/valeur\_marqueur\_flou$.

Il est également intéressant de pouvoir appliquer cet algorithme aux hiérarchies construites sur une partition fine quelconque. Nous avons vu qu'il est possible de construire des hiérarchies moyennant une inondation du graphe où les mesures géométriques sont estimées à partir de la surface des régions représentées par les nœuds et aux valeurs des arêtes. Pour tenir compte des marqueurs flous, il suffit alors, comme pour l'inondation sur le gradient, de diviser cette mesure par la valeur du marqueur.

Algorithme modifié L'algorithme que nous venons d'introduire permet de faire monter dans la hiérarchie certaines régions selon la valeur du marqueur flou qui leur est attribuée. Cet

algorithme fonctionne bien dans les cas où un seul marqueur flou est associé à chaque région d'intérêt, et où ce marqueur est placé avec précision, c'est-à-dire sur le minimum d'intérêt. Or dans certains cas, on ne dispose pas d'un pixel unique marquant le minimum, mais plutôt d'un ensemble de pixels englobant une partie de l'image que l'on aimerait placer en haut de la hiérarchie. En général, une telle région contiendra beaucoup de minima locaux. Si l'algorithme de calcul de la hiérarchie associée aux marqueurs flous est appliqué tel que nous venons de le présenter – en marquant tous les minima de la région comme étant importants –, la hiérarchie résultante aura tendance à placer *chacun des minima* en haut de la hiérarchie. La figure 6.37 illustre pourquoi cela arrive. Dans cette figure, chaque minimum M_i a un marqueur associé. Trois de ces minima, M_2, M_3 et M_4, correspondent à une région que l'on voudrait placer en haut de la hiérarchie. On leur a associé un marqueur de valeur $k < 1$. Le minimum M_1 a un marqueur de valeur 1, ce qui revient à dire qu'il n'a pas de marqueur (la vitesse de croissance n'est pas modifiée). Les deux premiers lacs à se rejoindre sont M_3 et M_4. Le temps nécessaire à leur remplissage est calculé comme nous l'avons expliqué, et celui dont le temps est inférieur donne sa valeur à l'arête. Ici, les temps sont V_3/k et V_4/k, où V_i est le volume du lac i. Dans ce cas, la valeur de l'arête est V_4/k, car $V_4 < V_3$. Le processus continue jusqu'à ce que l'inondation ait été complétée. Si nous examinons maintenant l'arbre résultant, et assumons que k a une faible valeur (afin de placer la région en haut de la hiérarchie), on voit que les arêtes reliant des nœuds à l'intérieur de la région marquée ont de très fortes valeurs (puisque les volumes ont été divisés par k, qui est petit) par rapport aux autres arêtes de l'arbre. Cela fait que ce sont ces arêtes qui seront éliminées lors de l'accès aux niveaux élevés de la hiérarchie. La conséquence sera une atomisation de la région que nous voulions placer en haut de la hiérarchie en de petites régions.

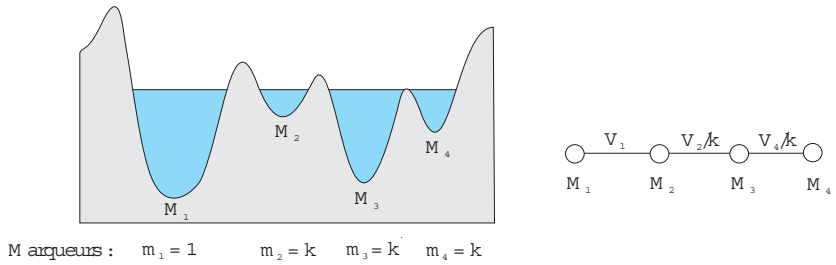


FIG. 6.37 – Introduction de marqueurs flous dans la construction de la hiérarchie.

Afin de résoudre ce problème, nous proposons une version modifiée de l'algorithme permettant de marquer des zones de l'image plus ou moins larges, avec la finalité de les placer haut dans la hiérarchie, mais en évitant l'effet d'atomisation des régions. Dans cette version, chaque marqueur a une valeur associée, ainsi qu'une étiquette permettant de l'identifier. Quand une région de l'image est marquée, toute la région obtient la même étiquette. Maintenant, quand deux lacs se rencontrent, on ne regardera pas uniquement la valeur du marqueur flou, mais aussi l'étiquette. Si deux lacs appartenant à un même marqueur flou (même étiquette) se rencontrent, la valeur du marqueur flou est ignorée (seulement les volumes sont comparés). Cependant, si les lacs ont deux étiquettes différentes, les volumes des lacs sont divisés par la valeur du marqueur flou. En appliquant cet algorithme à l'exemple de la figure 6.37, les valeurs des arêtes entre les nœuds M_2-M_3 et M_3-M_4, qui appartiennent au même marqueur, ne seraient pas divisées par la valeur du marqueur, k, alors que M_1-M_2 tiendrait compte de cette valeur. On éviterait ainsi que les arêtes internes à la région aient de trop fortes valeurs

conduisant à une atomisation de la région.

Remarquons que cet algorithme n'a pas d'équivalence en termes d'inondation synchrone. En effet, si l'inondation synchrone était utilisée, on ne pourrait pas modifier la vitesse de croissance d'un lac selon l'étiquette du marqueur de ses voisins, puisque les voisins d'un lac ne sont pas connus au début de l'inondation.

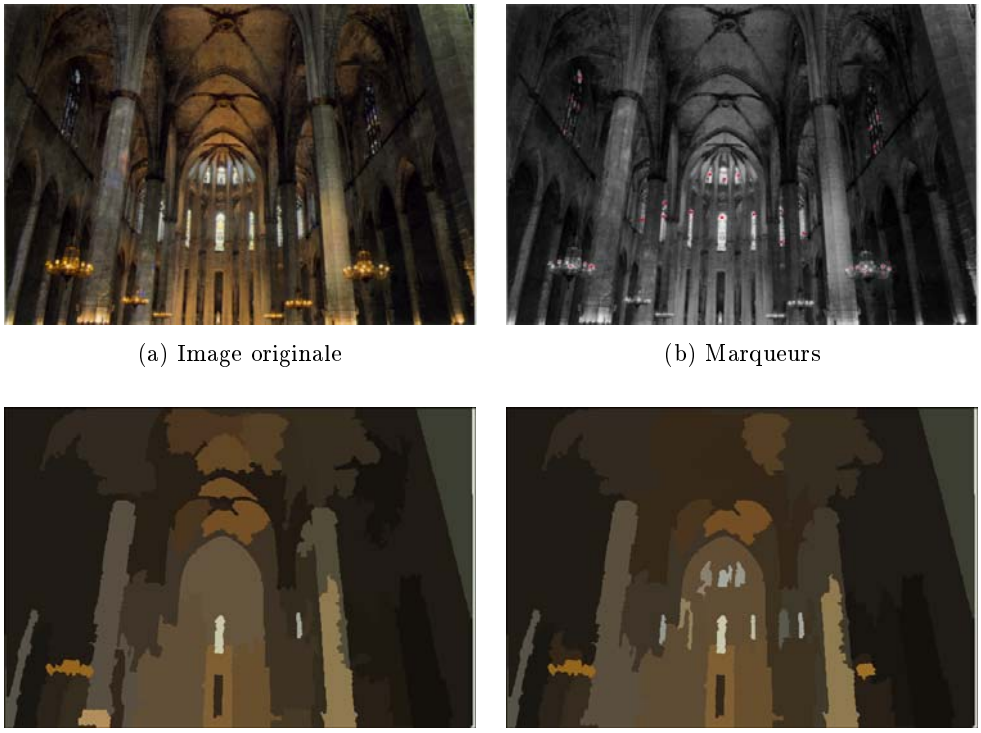
Nous présentons maintenant plusieurs exemples d'application des marqueurs flous. Le premier correspond à l'algorithme de base car un seul point est utilisé pour marquer chaque région d'intérêt. Le deuxième exemple correspond au cas où une zone de l'image est marquée contenant plusieurs minima. Pour cet exemple nous avons utilisé l'algorithme modifié.

6.6.2.3 Des exemples d'application

Les marqueurs flous constituent un outil puissant pour l'introduction d'information dans la construction de la hiérarchie. Les stratégies pour leur obtention dépendent de chaque application. Nous en présentons ici quelques exemples.

Sélection des marqueurs par dynamique Dans certaines applications, on ne cherche pas à extraire les objets sémantiques, mais à obtenir des représentations approchées des images en représentant les régions par leur couleur moyenne ou autre modèle de texture. L'intention est d'avoir une bonne représentation visuelle du contenu de l'image avec peu de régions. Des exemples de ce type d'application seraient le codage et la compression. Dans ces applications, les petits détails contrastés jouent un rôle important dans la qualité visuelle de la segmentation. L'utilisation du critère de volume, qui donne de bons résultats quand l'on cherche à extraire de grands objets à signification sémantique, n'est pas particulièrement approprié à ce type d'application. En utilisant les marqueurs flous pour aider les minima et les maxima de l'image à monter dans la hiérarchie on obtient des résultats beaucoup mieux adaptés, comme l'illustre la figure 6.38. Cette figure montre une photographie de l'intérieur de l'église « Santa Maria del Mar » à Barcelone. Sur cette image, les points de lumière correspondant aux fenêtres et aux lampes sont très importants du point de vue perceptuel. La figure 6.38-(c) montre le niveau de la hiérarchie avec 25 régions en utilisant l'ultramétrique basée sur le volume. La figure 6.38-(d) présente le résultat obtenu avec le même nombre de régions, en utilisant comme marqueurs flous les maxima de l'image avec plus forte dynamique. On remarque la meilleure qualité visuelle de la segmentation, grâce à l'inclusion de certains objets contrastés tels que les fenêtres. La valeur des marqueurs est inversement proportionnelle à leur dynamique : la dynamique 0 correspond à une valeur du marqueur de 1 (pas de marqueur), la dynamique maximale correspond à une valeur de marqueur de zéro (marqueur fort). La figure 6.38-(b) montre les marqueurs choisis (points rouges), superposés à la luminance de l'image.

Sélection des marqueurs par couleur Dans des applications où l'ont connaît la couleur des objets que l'on cherche à segmenter, il est possible d'utiliser ces informations pour extraire des marqueurs permettant de placer plus haut dans la hiérarchie les parties de l'image contenant ces couleurs. Dans l'exemple de la figure 6.39, nous avons extrait les marqueurs en détectant les pixels ayant une couleur contenue dans une région de l'espace couleur déterminée, suivi d'un filtrage pour éliminer les composantes connexes trop petites. Ensuite, chaque composante connexe constitue un marqueur, identifié avec une étiquette, et d'une valeur proche de zéro (grande influence du marqueur). La figure présente la segmentation en 4 régions sans et



(a) Image originale (b) Marqueurs

(c) Segmentation sans marqueurs flous (d) Segmentation avec marqueurs flous

FIG. 6.38 – Segmentation en 25 régions obtenue sans et avec marqueurs flous.

avec l'utilisation des marqueurs flous. Avec les marqueurs flous, la hiérarchie sépare la voiture du fond plus tôt que sans les marqueurs.

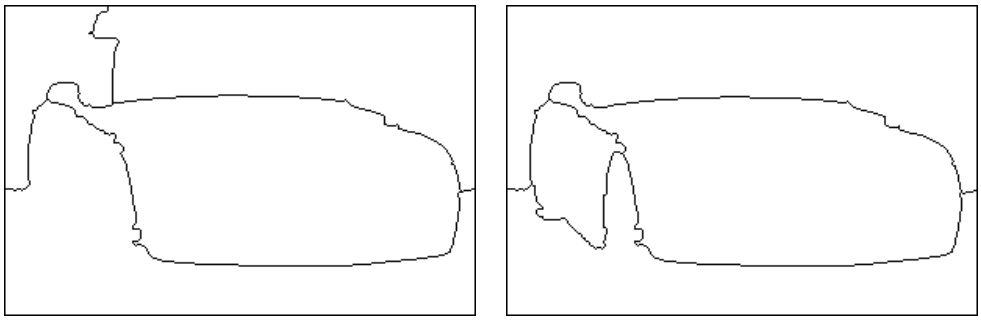
6.7 Mécanismes d'interaction

Nous avons vu dans le chapitre 4 différentes possibilités d'interaction avec une famille de partitions emboîtées du point de vue des fonctionnalités offertes à l'utilisateur. Nous décrivons ici la mise en place de ces systèmes d'un point de vue technique. L'idée est de mettre à la disposition de l'utilisateur un ensemble d'outils basés sur la hiérarchie. Comme la hiérarchie est construite une fois pour toutes, les calculs nécessaires pour répondre à l'interaction sont des manipulations simples de l'arbre qui s'exécutent très rapidement. Ceci permet à l'utilisateur d'expérimenter avec les différents outils et de les combiner sans que des calculs lourds ne soient nécessaires.

6.7.1 Accès à un niveau de la hiérarchie

Notre hiérarchie de partitions emboîtées est représentée par un arbre aux arêtes valuées. Les arêtes avec forte valeur séparent les régions ayant une plus grande dissimilarité. Comme nous l'avons introduit dans la section 6.3.4.2, l'accès aux différents niveaux de la hiérarchie se fait par élimination d'arêtes. Chaque composante connexe de la forêt résultante correspond à une région de la partition résultante. Ainsi, pour obtenir le niveau de la hiérarchie avec n régions, on élimine les $n-1$ arêtes de plus forte valeur. Pour obtenir les boules maximales

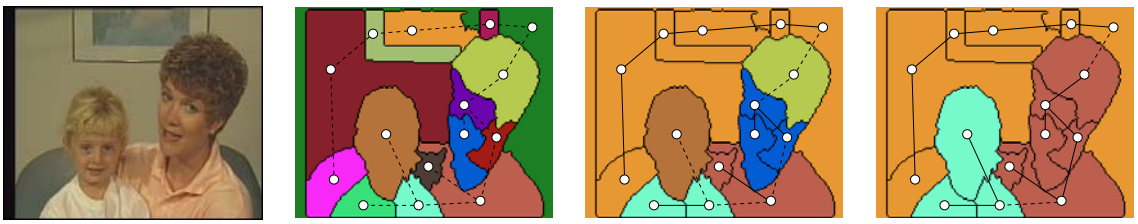
(a) Image originale (b) Marqueurs



(c) Segmentation sans marqueurs (d) Segmentation avec marqueurs

FIG. 6.39 – Segmentation en 4 régions obtenue sans et avec marqueurs flous.

de l'ultramétrique de rayon plus petit ou égal à r, il suffit de faire un seuillage des arêtes de façon à ce que toutes les arêtes de valeur plus forte que r soient éliminées. Nous parlons ici symboliquement d'élimination d'arêtes, or ceci entraînerait une perte d'information. Dans la pratique, les arêtes sont simplement activées ou désactivées. La figure 6.40 montre l'accès aux différents niveaux de la hiérarchie par élimination/désactivation d'arêtes. Les arêtes en pointillé ont été désactivées. Quand toutes les arêtes sont désactivées on a la partition fine (figure 6.40-(b)). En activant des arêtes par ordre croissant de valeur on accède à des niveaux de plus en plus grossiers (figures 6.40-(c) et (d)).



(a) Image originale (b) Partition fine (c) 5 régions (d) 3 régions

FIG. 6.40 – Accès aux différents niveaux de la hiérarchie par désactivation d'arêtes (les arêtes en pointillé sont désactivées).

En général les arêtes sont valuées avec le volume des régions, ce qui fait que l'ensemble des valeurs des arêtes présente une variance très importante. Il est parfois plus simple de

travailler avec des valeurs plus petites. Remarquons que réévaluer les arêtes consécutivement par ordre croissant de valeurs ne modifie pas la hiérarchie et peut simplifier dans certains cas sa manipulation.

La sélection des arêtes à éliminer se fait en parcourant l'arbre pour trouver les $n-1$ de plus forte valeur. Ces arêtes sont ensuite éliminées. Si les arêtes ont été réévaluées consécutivement à partir de 1, il suffit de faire un seuillage, en ne gardant que celles dont la valeur est plus petite ou égale à $maxval - n + 1$, où $maxval$ est le nombre d'arêtes du graphe.

L'élimination d'arêtes sur l'arbre produit une forêt. Chaque arbre, ou composante connexe, de la forêt correspond à une région de la partition. Le passage de la forêt à l'image se fait en deux étapes :

- Étiquetage des nœuds de la forêt de manière à ce que chaque composante connexe obtienne une étiquette. Pour étiqueter une composante de l'arbre on part d'un nœud quelconque et on reconstruit la composante connexe à l'aide d'une file d'attente.
- « Look up table » sur la partition fine : chaque pixel de l'image appartient à une région de la partition fine, et chaque région de la partition fine correspond à un nœud de la forêt. On peut de cette façon associer un nœud de l'arbre à chaque pixel de l'image. La valeur du pixel de l'image est alors remplacée par l'étiquette du nœud correspondant.

Outils associés Ce type d'opération correspond aux outils *barre de défilement* et *pinceau* que nous avons présenté dans le chapitre 4.

La barre de défilement permet à l'utilisateur d'accéder directement et rapidement à n'importe quel niveau de la hiérarchie. La représentation sous forme d'arbre est très simplifiée et permet la visualisation immédiate de la partition associée au niveau de résolution sélectionné par l'utilisateur.

La figure 6.41 montre la mise en œuvre pratique de cet outil. Deux fenêtres permettent de visualiser la partition associée au niveau courant de la hiérarchie, sous forme de contours (à gauche de la fenêtre) et sous forme de régions étiquetées (à droite). La représentation sous forme de contours permet de mieux évaluer la précision des contours par rapport à l'image originale. Moyennant la barre de défilement, l'utilisateur peut visualiser directement n'importe quel niveau de la hiérarchie.

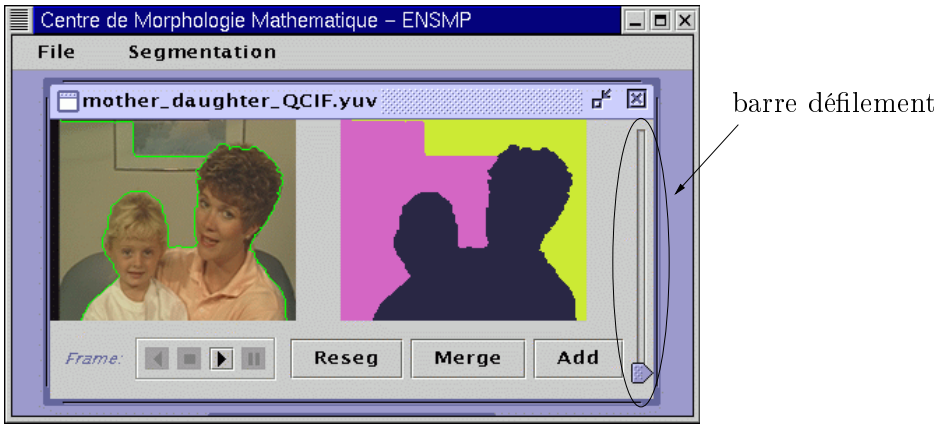


FIG. 6.41 – La barre de défilement permet d'accéder rapidement aux différents niveaux de résolution de la hiérarchie.

Pour la mise en œuvre de l'outil pinceau, un niveau de la hiérarchie est choisi au préalable. L'utilisateur *peint* l'objet en déplaçant la souris sur l'image. À tout moment, la forme du pinceau correspond à la boule de l'ultramétrique centrée sur la position de la souris. L'utilisateur peut augmenter ou réduire la taille du pinceau. La réduction de la taille du pinceau se fait en choisissant un niveau plus fin de la hiérarchie, tandis que l'augmentation se fait en accédant à un niveau plus grossier.

6.7.2 Actions locales sur des régions

Souvent, l'utilisateur souhaite segmenter certaines parties de l'image avec plus de détail. Accéder à un niveau plus fin de la hiérarchie selon le mécanisme précédent donnera une segmentation plus fine, non seulement de la région souhaitée, mais de toute l'image. Il est intéressant de pouvoir, en partant d'un niveau donné de la hiérarchie, segmenter avec plus ou moins de détail certaines régions, mais *sans toucher au reste de l'image*. Nous avons vu dans le chapitre 4 que ceci correspond à la modification du diamètre de seulement certaines boules de l'ultramétrique.

La représentation de la hiérarchie sous forme d'arbre aux arêtes valuées permet une implantation très simple et intuitive de cette opération. Nous partons ici d'une forêt représentant une partition P_i de la famille de partitions contenant n régions. Sur cette forêt, $n-1$ arêtes ont été désactivées, et chaque composante connexe correspond à une boule de l'ultramétrique. Nous supposons aussi qu'une des boules a été sélectionnée.

6.7.2.1 Réduction du rayon d'une boule

La réduction du rayon d'une boule équivaut à resegmenter cette boule en régions plus petites. Pour ce faire, il suffit de parcourir la boule, en ordonnant les arêtes par ordre croissant de valeurs. Ensuite, les arêtes de plus forte valeur seront éliminées. La différence avec l'accès global à un niveau de la hiérarchie est qu'ici seulement les arêtes *appartenant à la boule sélectionnée* sont considérées. En éliminant les $k-1$ arêtes de plus forte valeur, la boule sélectionnée est divisée en k boules plus petites.

La figure 6.42 présente un exemple d'application. La figure 6.41 correspond au niveau de la hiérarchie avec 3 régions. Supposons que l'utilisateur souhaite plus de détail à l'intérieur des personnages, mais pas sur le fond. L'accès à un niveau plus fin de la hiérarchie resegmenterait sans doute les personnages, mais aussi le fond. La réduction du rayon d'une boule donnée permet de resegmenter cette région sans modifier le reste de l'image. À partir du niveau avec 3 régions de la figure 6.41 l'utilisateur a cliqué sur la région contenant les personnages et a demandé une resegmentation de cette région en 5 régions. Le résultat est illustré dans la figure 6.42.

6.7.2.2 Augmentation du rayon d'une boule

L'augmentation du rayon d'une boule équivaut à la fusionner avec ses régions voisines les plus similaires. Ainsi, il est nécessaire de parcourir la boule en cherchant les arêtes désactivées – c'est-à-dire les arêtes joignant un nœud appartenant à la boule avec un nœud ne lui appartenant pas – et de les ordonner par ordre décroissant de valeurs. En activant les k arêtes désactivées de valeur plus petite on augmente le rayon de la boule en la fusionnant avec les boules voisines les plus similaires.

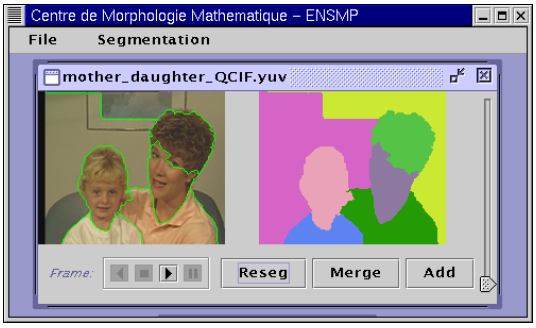


FIG. 6.42 – Réduction du rayon d'une boule.

À partir de la figure 6.42, supposons maintenant que l'utilisateur souhaite une segmentation plus grossière du fond (une seule région). L'utilisateur clique alors sur une des régions du fond et demande de l'agrandir. La région la plus proche d'après la hiérarchie est fusionnée avec celle-ci. La figure 6.43 montre le résultat.

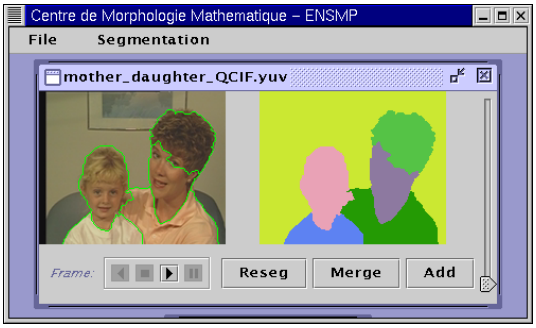


FIG. 6.43 – Augmentation du rayon d'une boule donnée.

6.7.2.3 Amélioration en introduisant un critère de distance

L'utilisation pratique dans un logiciel de segmentation des opérations de réduction et d'augmentation du rayon des boules a mis en évidence un problème lié à la manière dont ces opérations sont perçues par l'utilisateur. Du point de vue de l'utilisateur, ces opérations sont des resegmentations et des fusions de régions. Dans le cas de la resegmentation, quand la région à resegmenter est grande, l'utilisateur a tendance à cliquer avec la souris *près du point où il souhaite voir le contour apparaître*. Or tel que nous avons présenté cette interaction, les résultats ne tiennent pas compte du point exact où l'utilisateur a cliqué. Si le nouveau contour qui apparaît est éloigné du point sélectionné, l'utilisateur est relativement déçu du résultat. Pour cette raison nous avons modifié le mécanisme de réduction du rayon des boules afin de tenir compte du point où l'utilisateur clique. L'idée est de pénaliser les contours trop éloignés du point sélectionné. Pour cela, les valeurs des arêtes sont modifiées de la manière suivante :

$$\begin{aligned} nouvelle\_valeur(nœud1, nœud2) = {} & ancienne\_valeur(nœud1, nœud2) + \\ & distance\_normalisée(nœud\_cliqué, nœud2) \end{aligned}$$

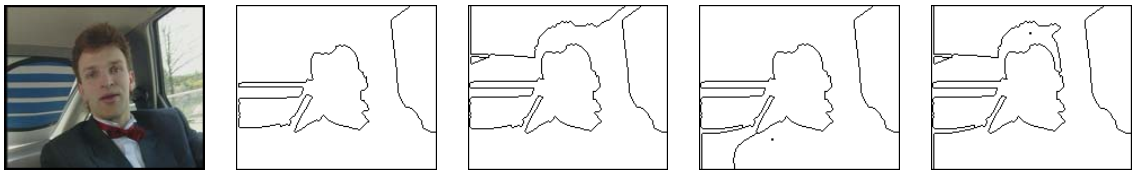
où

$$distance\_normalisée(nœud\_cliqué, nœud2) = distance(nœud\_cliqué, nœud2) \cdot facteur$$

$$\text{avec } facteur = \frac{valeurmaxarête}{distance\_maximale}$$

$nœud2$ est celui des deux nœuds qui est le plus éloigné du $nœud\_cliqué$, et $distance(nœud\_cliqué, nœud2)$ est la distance euclidienne entre le premier pixel des régions associées aux nœuds.

La figure 6.44 montre les résultats obtenus pour l'image « Carphone ». Le point de départ est une partition très grossière (3 régions) de la hiérarchie. Pour la méthode de resegmentation de la section 6.7.2.1, si l'utilisateur clique à l'intérieur de la région correspondant aux corps afin de resegmenter cette région, la partition obtenue est celle de la figure 6.44-(c), indépendamment du point exact choisi. En introduisant le critère de distance, la partition résultante dépend du point sélectionné. Le résultat est illustré dans les figures 6.44-(d) et (e), où le point cliqué est marqué par un point noir.



(a) (b) (c) (d) (e)

FIG. 6.44 – Resegmentation en tenant compte du point choisi. (a)-Image originale, (b)-niveau de la hiérarchie avec 3 régions, (c)-resegmentation du personnage en 2 régions, (d)-resegmentation en 2 régions en tenant compte du point choisi, (e)-resegmentation en 2 régions en tenant compte du point choisi.

6.7.3 Segmentation avec marqueurs

6.7.3.1 Segmentation traditionnelle avec marqueurs

La segmentation morphologique traditionnelle avec des marqueurs peut se réaliser sur la hiérarchie représentée par l'arbre aux arêtes valuées. Il s'agit de trouver la partition telle que chaque région contienne un seul marqueur et que la distance entre régions soit maximisée.

Ici, nous supposons que c'est l'utilisateur qui dessine les marqueurs sur l'image pour désigner les objets d'intérêt. Les marqueurs sont ensuite associés aux nœuds de l'arbre. La correspondance entre marqueur et nœud de l'arbre se fait en passant par la partition fine : tous les nœuds de l'arbre dont la région de la partition fine contient un marqueur sont marqués. Si une région de la partition fine contient plusieurs marqueurs, il est nécessaire, en théorie, de couper cette région, ce qui peut être fait avec une ligne de partage des eaux à partir de ces marqueurs et restreinte à la région concernée. Il est alors nécessaire de reconstruire la hiérarchie avec la nouvelle partition fine. En pratique, cette opération est seulement nécessaire si la région concernée est suffisamment grande (des centaines de pixels). Si elle est petite, un des deux marqueurs peut être choisi de manière aléatoire, puisque cela aura très peu d'influence sur le résultat final. C'est le cas quand la hiérarchie a été construite sur les régions de la LPE (cf.

section 6.2.3), par opposition au cas où elle est basée sur les zones plates couleur, car dans ce dernier cas les régions de la partition fine sont beaucoup plus grandes.

Une fois que certains nœuds ont été marqués sur l'arbre, l'algorithme 6.2 permet de les propager à travers les arêtes de plus faible valeur jusqu'à ce que chaque nœud obtienne une étiquette [53]. Les arêtes ayant des extrémités sur des régions appartenant à des marqueurs différents sont désactivées créant ainsi une forêt dans laquelle chaque arbre (composante connexe) contient un unique marqueur.

Algorithme 6.2 Algorithme de segmentation à partir de marqueurs.

T : Arbre aux arêtes valuées

$FileAttenteH \leftarrow$ Mettre dans la file d'attente hiérarchique les nœuds voisins des nœuds marqués, avec priorité donnée par les valeurs d'arête.

while NoVide($FileAttenteH$) **do**

$v_i \leftarrow$ ExtraireNœud($FileAttenteH$)

if $etiquette(v_i) == \varnothing$ **then**

Trouver voisins de v_i.

Étiqueter v_i avec l'étiquette du voisin pour lequel la valeur d'arête correspond à la priorité actuelle.

$FileAttenteH \leftarrow$ voisins non étiquetés de v_i, avec priorité donnée par les valeurs d'arête les reliant à v_i

end if

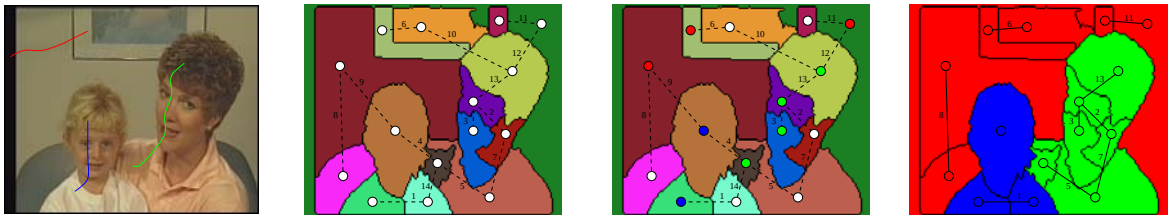
end while

Lorsque chaque marqueur est attribué à un nœud unique de l'arbre, l'algorithme 6.2 produit une seule composante connexe par marqueur. Cette situation est souhaitable, car il est intéressant de pouvoir identifier une composante connexe sur l'image avec une composante sur l'arbre. En effet, cela simplifie l'exécution ultérieure d'autres interactions telles que les opérations de resegmentation/fusion d'une région, puisqu'on est sûr, en parcourant la composante connexe sur l'arbre, d'avoir traité toute la région concernée. En pratique l'utilisateur n'introduit pas des marqueurs composés d'un point unique, mais d'un tracé à la forme compliquée (figure 6.45). Plusieurs problèmes peuvent apparaître :

- si l'utilisateur introduit des marqueurs qui se croisent, le résultat ne sera connexe ni sur l'image, ni sur l'arbre ;
- si les marqueurs sont connexes sur l'image, le résultat sera aussi connexe sur l'image, mais chaque composante connexe sur l'image peut être composée de plusieurs composantes connexes sur l'arbre. Ceci arrive quand le chemin sur l'arbre entre deux nœuds appartenant au même marqueur m_1 rencontre un marqueur différent m_2. Dans cette situation il n'y a pas moyen d'obtenir une seule composante connexe sur l'arbre pour le marqueur m_1 (même si sur l'image le résultat est connexe).

Le premier problème est causé par une mauvaise utilisation de l'outil, et il peut être résolu, une fois la segmentation associée aux mauvais marqueurs obtenue, par un réétiquetage des régions non connexes. On se ramène de cette manière à la situation du deuxième problème, où les régions sont connexes sur l'image, mais pas sur l'arbre.

La figure 6.45 illustre la manière comme plusieurs composantes connexes de la forêt peuvent



(a) Marqueurs (b) Arbre et partition fine (c) Marqueurs sur l'arbre (d) Résultat

FIG. 6.45 – Segmentation à partir de marqueurs sur l'arbre. Sur la segmentation résultante, chaque composante connexe sur l'image peut être composée de plusieurs composantes connexes de la forêt.

être associées à une unique région de la segmentation résultante. L'utilisateur introduit trois tracés pour marquer les trois régions d'intérêt (figure 6.45-(a)). Ensuite, les nœuds de l'arbre correspondant à une région de la partition fine touchée par un marqueur acquièrent l'étiquette du marqueur (figure 6.45-(c)). Les nœuds restants ne contiennent pas d'étiquette (nœuds en blanc sur le dessin). L'algorithme 6.2 est alors appliqué pour propager les marqueurs à tous les nœuds de l'arbre (figure 6.45-(d)). Dans le résultat, chaque région est composée d'une seule composante connexe sur l'image. Or chaque composante connexe sur l'image ne correspond pas nécessairement à une unique composante connexe (arbre) de la forêt, ce qui serait souhaitable.

À cause de ce problème l'algorithme 6.2 n'est pas applicable – du moins pas directement – au cas où la segmentation se fait à partir de marqueurs dessinés par l'utilisateur. Une nouvelle approche est nécessaire. Notre solution consiste à travailler sur le graphe de voisinage, sur lequel nous appliquerons un algorithme de calcul de l'arbre similaire à l'algorithme de construction de la hiérarchie, mais avec des contraintes additionnelles, ce qui nous permettra de trouver un arbre compatible avec les marqueurs donnés. L'algorithme 6.3 détaille la méthode de construction d'une hiérarchie compatible avec un ensemble de marqueurs. Sur cet algorithme, les fonctions Surface(v_k) et Volume(v_k) donnent la surface et le volume des composantes connexes de l'arbre contenant le nœud v_k. Les marqueurs sont imposés sur le graphe de voisinage. Ainsi, sur ce graphe quelques nœuds sont marqués, et d'autres ne le sont pas. Les différents nœuds appartenant à un même marqueur sont identifiés par une même étiquette. Une simulation d'inondation a lieu, avec construction simultanée de la hiérarchie. Si deux régions se retrouvent, et que l'une d'elles a un marqueur et pas l'autre, celle n'ayant pas de marqueur est marquée avec l'étiquette de l'autre. Si les deux régions sont déjà marquées avec des marqueurs différents, on ignorera cette rencontre. Avec cette méthode, l'arbre représentant la hiérarchie est en réalité une forêt avec autant de composantes connexes qu'il y avait de marqueurs connexes sur l'image. Ceci ne pose pas un problème, car en général le nombre de marqueurs est petit, et ces régions peuvent être fusionnées manuellement par l'utilisateur si cela s'avère nécessaire. Les arêtes à l'intérieur d'une composante connexe sont valuées avec leur valeur d'extinction.

Cette méthode de construction permet d'enchaîner la segmentation par marqueurs avec des opérations de resegmentation locale des régions. Ceci est particulièrement utile lorsque les résultats obtenus à partir des marqueurs ne correspondent pas exactement aux résultats recherchés. S'il n'était pas possible de continuer à interagir avec la segmentation résultante, l'utilisateur serait obligé d'introduire de nouveaux marqueurs jusqu'à l'obtention du résultat souhaité.

La figure 6.46 présente les résultats de deux segmentations à partir de marqueurs, suivis d'une resegmentation d'une des régions obtenues. La resegmentation est possible parce que la hiérarchie est construite en tenant compte des marqueurs. De cette manière à chaque composante connexe sur l'image correspond une composante connexe sur l'arbre.

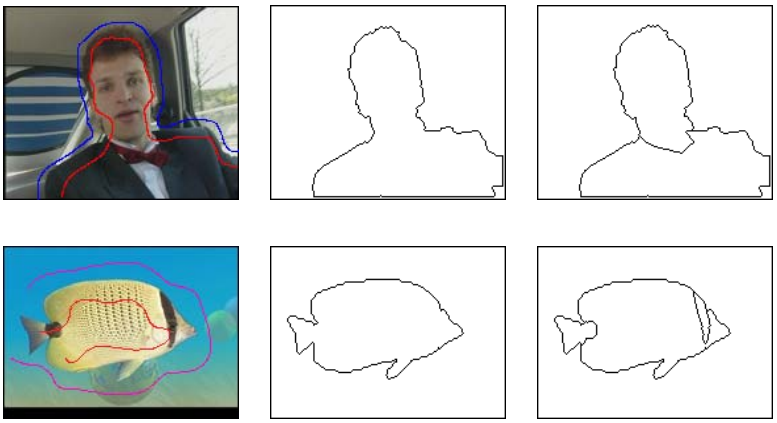


FIG. 6.46 – Segmentation à partir de marqueurs avec construction de la hiérarchie et resegmentation de l'une des régions.

6.7.3.2 Segmentation avec un seul marqueur

La représentation par dendrogramme de la hiérarchie inspire la segmentation à partir d'un marqueur unique [81], [80]. À partir de plusieurs nœuds marqués sur le dendrogramme, le sommet commun aux nœuds marqués est choisi comme résultat de la segmentation (figure 6.47).

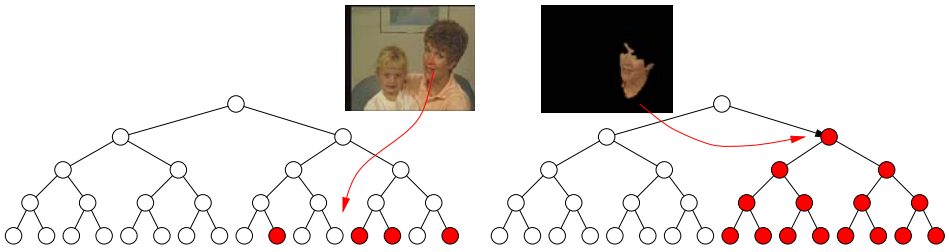


FIG. 6.47 – Segmentation sur la hiérarchie à partir d'un seul marqueur.

L'arbre aux arêtes valuées contenant plus d'information que le dendrogramme (cf. chapitre 3), il est aussi possible de mettre en place ce type d'interaction sur l'arbre aux arêtes valuées. Il s'agit de trouver le niveau de la hiérarchie plus petit pour lequel tous les noeuds marqués sont contenus dans une même région, et de prendre cette région comme résultat. Pour cela, il suffit d'ajouter des arêtes par ordre croissant de valeurs jusqu'à ce que tous les noeuds marqués soient contenus dans une même composante connexe. La figure 6.48 montre un exemple.

Remarquons toutefois que, bien que cette méthode soit plus intuitive que la segmentation traditionnelle par marqueurs où il est nécessaire de marquer aussi le fond, elle est moins robuste. En effet, si la hiérarchie a commis quelques erreurs de hiérarchisation des régions –

Algorithme 6.3 Algorithme de construction de la hiérarchie à partir de marqueurs.

G : Graphe de voisinage avec arêtes valuées avec une mesure de dissimilarité locale, et nœuds valués avec la surface des régions

T : Arbre aux arêtes valuées (initialement vide)

v_i : Nœuds de l'arbre (et du graphe)

e_{ij} : Arête reliant les nœuds v_i et v_j

pile ← Mettre dans la pile les arêtes de G, ordonnées par ordre croissant de valeurs.

while NoVide(*pile*) **do**
  e_{ij} ← ExtraireArête(*pile*)
  v_i, v_j ← TrouverExtrémités(e_{ij})
  if arête e_{ij} ne forme pas un cycle dans T **then**
    if (marqueur(v_i) == ∅) ou (marqueur(v_j) == ∅)
    ou (marqueur(v_i) == marqueur(v_j)) **then**
      if (marqueur(v_i) == ∅) et (marqueur(v_j) != ∅) **then**
        for v_k appartenant à la même composante connexe que v_i **do**
          marqueur(v_k) ← marqueur(v_j)
        end for
      else if (marqueur(v_j) == ∅) et (marqueur(v_i) != ∅) **then**
        for v_l appartenant à la même composante connexe que v_j **do**
          marqueur(v_l) ← marqueur(v_i)
        end for
      end if
      if Volume(v_i) > Volume(v_j) **then**
        Surface(v_i) ← Surface(v_i) + Surface(v_j)
        AjouterArête(T, e_{ij})
        ValuerArête(e_{ij}, Volume(v_j))
      else
        Surface(v_j) ← Surface(v_i) + Surface(v_j)
        AjouterArête(T, e_{ij})
        ValuerArête(e_{ij}, Volume(v_i))
      end if
    end if
  end if
end while

par exemple par manque de contraste entre des objets différents – elle peut vite conduire à fusionner beaucoup trop de régions. La méthode traditionnelle – un marqueur pour chaque objet plus un marqueur pour le fond –, avec l'algorithme que nous avons proposé (alg. 6.3), construit la hiérarchie en tenant compte des marqueurs, ce qui évite des erreurs.

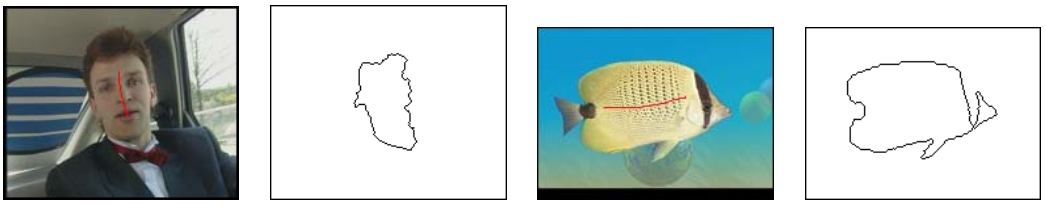


FIG. 6.48 – Segmentation sur la hiérarchie à partir d'un seul marqueur.

6.7.4 La baguette magique

Nous avons décrit dans le chapitre 4 le fonctionnement de la baguette magique telle qu'on la trouve dans des logiciels commerciaux. Sa fonction est de détecter des régions homogènes en couleur, la notion d'homogénéïté étant donnée par une tolérance sur la couleur que l'utilisateur peut contrôler. L'algorithme qu'on trouve dans ces logiciels est une simple croissance de régions pixel à pixel avec un seuil de tolérance couleur. Les petites régions contrastées à l'intérieur des régions ne sont pas sélectionnées, ce qui souvent n'est pas souhaité.

La hiérarchie permet de mettre en place une fonction baguette magique beaucoup moins sensible au bruit et mieux adaptée au contenu de l'image. Pour cela, on prend la boule maximale de l'ultramétrique centrée sur le point choisi par l'utilisateur et telle que l'écart entre la couleur moyenne de la boule et la couleur du pixel choisi soit plus petit que la tolérance. L'algorithme de calcul ajoute à partir du pixel choisi des arêtes de l'arbre par ordre croissant de valeur, tant que la couleur moyenne de la région ainsi obtenue est à une distance couleur du pixel choisi inférieure à la tolérance.

La figure 6.49 montre quelques résultats obtenus avec cet outil, qui peut être utilisé quand les régions que l'on cherche à détecter ont une couleur homogène.

6.7.5 Le lasso

Le lasso est un outil permettant de trouver la forme de l'objet à partir d'un contour extérieur approximé introduit par l'utilisateur.

Dans l'outil que nous proposons, nous permettons à la hiérarchie de choisir les meilleures régions contenues à l'intérieur du contour introduit par l'utilisateur, avec une contrainte de surface. C'est-à-dire, la surface de la région résultante doit être au moins égale à un certain pourcentage de celle contenue à l'intérieur du contour introduit par l'utilisateur, ce pourcentage étant un paramètre contrôlé par l'utilisateur.

L'approche que nous avons adoptée consiste à prendre l'ensemble des boules maximales de l'ultramétrique complètement contenues à l'intérieur du contour dessiné par l'utilisateur, et à choisir celle de surface maximale. Si la condition sur la surface est vérifiée, cette région est retenue comme solution. Si non, l'arête de valeur plus faible joignant deux des boules intérieures au contour est ajoutée. Ce procédé est itéré jusqu'à ce que la condition soit vérifiée, ou jusqu'à ce que toutes les arêtes intérieurs au contour aient été ajoutées. Remarquons que, étant donnée

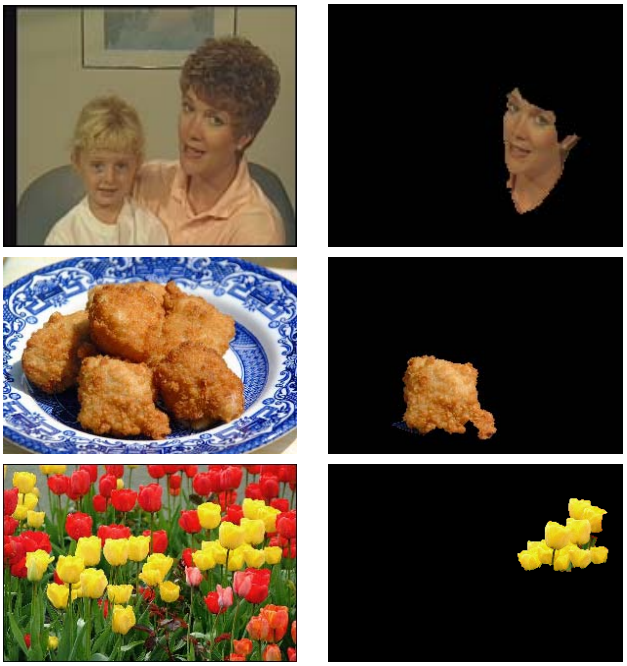


FIG. 6.49 – Régions obtenues avec l'outil baguette magique.

la structure d'arbre utilisée pour représenter la hiérarchie, cet outil a une utilité plutôt limitée. L'arbre ne contenant qu'un seul chemin entre toute paire de nœuds, dans le cas où l'arbre sort du contour pour rentrer dans la région, les deux composantes connexes intérieures à la région ne peuvent pas être connectées. Ainsi, cet outil fonctionne bien seulement dans les cas où les objets sont bien contrastés avec le fond. Il serait cependant possible de réaliser cette opération sur un graphe de voisinage où les arêtes auraient été réévaluées avec les ultramétriques données par l'arbre. La figure 6.50 en montre un exemple.

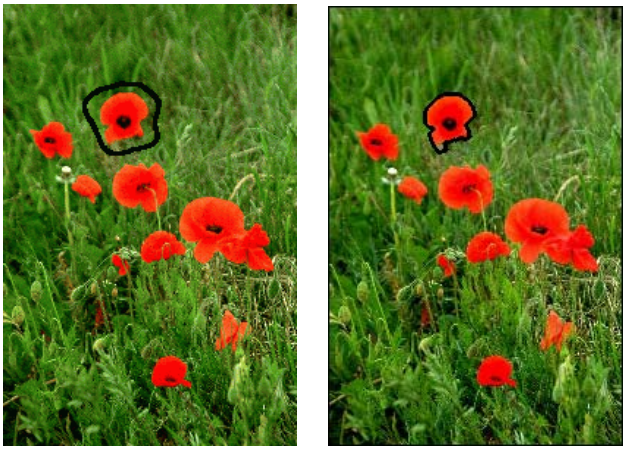


FIG. 6.50 – Région obtenue avec l'outil lasso.

6.8 Conclusions

Au début de ce chapitre, nous avons présenté plusieurs hiérarchies de type « bottom-up » qui découlent naturellement de certains opérateurs morphologiques. Parmi elles, nous avons montré que c'est la LPE qui présente la meilleure performance, pour deux raisons. D'abord, la partition fine associée n'isole pas les petites zones plates contrastées correspondant en général aux zones de transition, comme c'est le cas des nivellements ou des zones plates croissantes. Ensuite, la LPE offre un plus grand potentiel, du point de vue de la *qualité* des hiérarchies obtenues. En effet, au moyen de l'inondation synchrone, nous avons défini des ultramétriques significatives. Les ultramétriques définies par inondation à profondeur, surface et volume constants correspondent aux valeurs d'extinction définies dans [98]. Mais la construction de la hiérarchie directement sur les régions de la LPE présente deux problèmes :

- une sursegmentation importante de la partition fine (niveau le plus bas de la hiérarchie), qui peut pénaliser les temps de calcul pour manipuler et accéder aux différents niveaux de la hiérarchie ;
- des fuites à travers des passages étroits au long de la frontière entre bassins versants, ce qui amène à une hiérarchisation erronée de certaines régions.

Pour résoudre ces problèmes, nous avons étendu le calcul des ultramétriques sur les régions de la LPE à des partitions fines quelconques, ce que nous avons fait à partir d'une formulation du problème sous forme de graphe. Nous avons également proposé une méthode de calcul de la partition fine qui évite la sursegmentation grâce à la détection des zones plates couleur. On réduit de cette manière de plus de 60% le nombre de régions de la partition fine par rapport à celle obtenue par LPE. Bien que des réductions sur le nombre de régions de la partition fine peuvent être obtenues en appliquant des préfiltrages importants à l'image, une réduction aussi importante n'aurait pas pu être obtenue par filtrage sans risque d'éliminer des régions significatives de l'image.

Des ultramétriques que nous avons considérées, c'est le volume qui présente les meilleures performances dans un cadre général. Il est cependant mieux adapté aux scènes où l'objet à segmenter est au premier plan. Dans d'autres cas, les résultats peuvent être améliorés en combinant des hiérarchies différentes construites sur le même ensemble de données. Nous avons proposé deux manières de combiner des hiérarchies : les algèbres d'ultramétriques, et l'utilisation des distances lexicographiques. Nous avons présenté des cas de figure où la combinaison de hiérarchies améliore les résultats par rapport à l'utilisation d'une hiérarchie unique. Les mécanismes que nous avons présentés ouvrent les portes à de multiples possibilités pour la création de nouvelles hiérarchies à partir de celles existantes. Il est également possible de concevoir de nouvelles hiérarchies adaptées à des applications pour lesquelles on dispose de connaissances *a priori* sur les images traitées, moyennant les marqueurs flous. Nous avons également vu une possible amélioration de la robustesse en construisant la hiérarchie de manière récursive.

Finalement, nous avons décrit la mise en œuvre d'une batterie d'outils d'interaction, flexibles et puissants, qui dérivent naturellement de la notion de hiérarchie. Nous avons également introduit des améliorations importantes aux outils de segmentation disponibles sur le marché.

Les avantages du système que nous proposons sont :

- Faible complexité de la méthode de calcul de la hiérarchie. En effet, une inondation de l'image gradient ou du graphe suffisent au calcul de la hiérarchie.
- La représentation très simplifiée de la hiérarchie par arbre aux arêtes valuées permet de la manipuler très rapidement, par simple activation/désactivation d'arêtes. L'utilisateur

perçoit donc la réponse aux interactions comme immédiate, ce qui est une caractéristique très souhaitable dans un algorithme interactif.

- Par rapport à d'autres systèmes utilisant des représentations d'arbre similaires [62], l'utilisation d'une partition fine adaptée permet de réduire énormément la complexité. En effet, dans notre application un nœud de l'arbre représente une région homogène de l'image, et non pas un pixel, comme c'est le cas dans d'autres applications (pour une image QCIF, la partition fine contient typiquement moins de 300 nœuds). Mais l'amélioration la plus importante par rapport à d'autres méthodes est l'utilisation des ultramétriques obtenues par inondation synchrone. Ces ultramétriques, et en particulier celle basée sur le volume, offrent une manière de hiérarchiser les régions beaucoup mieux adaptée à la perception humaine que les mesures basées uniquement sur des similarités locales entre régions. De plus, la possibilité de combiner ces ultramétriques de diverses manières permet de construire facilement des distances adaptées à des contextes spécifiques.
- Les outils d'interaction sont intuitifs et ne requièrent pas de connaissances des algorithmes sous-jacents de la part de l'utilisateur. Le système dépend de peu de paramètres, qui peuvent être fixés pour toutes les images. Ces paramètres sont taille du pré-filtrage et la taille des régions considérées comme appartenant aux zones de transition dans le calcul de la partition fine. Ainsi, un utilisateur non expérimenté n'a pas besoin d'ajuster des paramètres. Cependant, il est possible de permettre le réglage de ces paramètres aux utilisateurs avancés.

Chapitre 7

Segmentation de séquences vidéo

7.1 Introduction

Nous avons présenté dans le chapitre 6 une technique pour calculer une hiérarchie de partitions emboîtées associée à une image fixe. Nous avons également présenté une batterie d'outils permettant à l'utilisateur de définir facilement et rapidement les objets d'intérêt. Nous abordons ici le cas des séquences vidéo. Dans certains contextes d'application, comme c'est le cas de MPEG4, les objets d'intérêt doivent être définis, non pas pour une seule image, mais pour toute une séquence vidéo. La solution adoptée est en général l'une des suivantes :

- segmenter interactivement les images de la séquence une à une ;
- segmenter interactivement une des images de la séquence, et ensuite appliquer un algorithme automatique de suivi.

La première approche n'est faisable que si la séquence est relativement courte et exige beaucoup de travail, car elle n'exploite pas la redondance entre les images successives. L'approche par suivi est celle que l'on trouve le plus souvent dans la littérature [44], [72], [25], [111], [19], [26], [38].

Contrairement à la plupart des systèmes existants, nous ne proposons pas une approche de suivi des objets, mais une méthode basée sur la construction d'une hiérarchie associée à la séquence vidéo complète.

Nous détaillons dans la section 7.2 les problèmes que les approches par suivi rencontrent dans une application interactive. Dans la section 7.3 nous montrons les avantages de disposer d'une hiérarchie de partitions emboîtées associée à la séquence entière et définissons les caractéristiques que cette hiérarchie doit avoir. La section 7.4 illustre l'extension aux images tridimensionnelles de l'algorithme que nous avons proposé pour les images 2D, et décrit les problèmes qui apparaissent quand cette approche est appliquée aux séquences vidéo. Dans la section 7.5 nous proposons une solution mixte 3D-récursive permettant de construire une hiérarchie de partitions emboîtées associée à une séquence vidéo. Finalement, la section 7.6 présente des résultats obtenus en utilisant la méthode proposée. Dans la section 7.7, nous discutons avec plus de détails les caractéristiques de notre approche.

7.2 L'approche par suivi

7.2.1 Description

Dans un contexte de segmentation interactive, l'approche par suivi demande à l'utilisateur de définir interactivement les objets d'intérêt sur l'une des images de la séquence. Ensuite, à partir de cette définition, les objets sont automatiquement suivis pour toute la séquence. Cette approche comporte deux étapes clairement identifiées :

- la définition interactive de l'objet d'intérêt sur une image de la séquence ;
- le suivi de l'objet au long de la séquence.

Il est important de remarquer que dans cette approche le traitement de la séquence ne peut commencer qu'une fois l'objet d'intérêt défini, car cette définition sert à initialiser l'algorithme de suivi. Ainsi, la dépense principale en temps de calcul se fait *après* que l'utilisateur ait réalisé son travail. Si l'utilisateur veut modifier la définition de l'objet, il est nécessaire de recommencer tout le processus depuis le début et de refaire les calculs.

7.2.2 Problèmes

Par ailleurs, l'utilisation d'un algorithme de suivi dans un contexte de segmentation interactive présente quelques problèmes, que nous décrivons ensuite.

7.2.2.1 Difficulté à corriger des erreurs

Lorsque tout se passe idéalement, l'algorithme de suivi minimise le temps de travail de l'utilisateur, car une fois l'objet d'intérêt défini, l'algorithme s'exécute de manière automatique, sans requérir plus d'intervention. Mais souvent la réalité est différente. Comme pour toute technique, les algorithmes de suivi sont susceptibles de produire des erreurs à un moment donné. Ces erreurs peuvent survenir pour des raisons très différentes, qui dépendent des caractéristiques et de la conception de l'algorithme. Les causes d'erreur peuvent être le manque de contraste de l'objet, des déformations, le fort mouvement de la caméra, etc. Quand une erreur se produit sur une image, elle peut facilement se propager au reste de la séquence. Ainsi, l'utilisateur, qui a laissé l'algorithme s'exécuter sans supervision, retrouve à son retour des erreurs à partir d'une image donnée et qui se sont souvent propagées jusqu'à la fin de la séquence. Il peut alors adopter une des deux stratégies :

- corriger manuellement les erreurs image par image ;
- corriger la première image erronée et relancer ensuite l'algorithme à partir de cette image.

La correction manuelle n'est envisageable que quand peu d'images ont été affectées. Quand l'erreur atteint un grand nombre d'images, il ne reste comme solution que de relancer l'algorithme à partir de la première image où l'erreur s'est produite. Mais cela ne garantit pas que les erreurs ne vont pas se reproduire, à moins que des techniques d'apprentissage aient été spécifiquement intégrées. Si les erreurs sont survenues à cause de caractéristiques de la séquence la rendant particulièrement difficile à segmenter, il est probable que ces erreurs se reproduiront. L'utilisateur devra ainsi répéter le processus autant de fois que nécessaire jusqu'à obtenir les résultats souhaités pour toute la séquence. La quantité d'interaction nécessaire pour obtenir la segmentation peut ainsi augmenter de façon considérable. Mais surtout, c'est le temps d'exécution qui augmente. En effet, à chaque fois que l'utilisateur relance l'algorithme tous les calculs sont refaits à partir du nouveau masque, les calculs réalisés pendant les itérations

précédentes n'étant en général pas réutilisés. Or si la séquence est longue, ces calculs peuvent durer plusieurs dizaines de minutes, voire des heures, et chaque correction génère une perte de temps importante.

7.2.2.2 Manque de flexibilité

Les algorithmes de suivi ont besoin en entrée d'un masque définissant les objets d'intérêt, et ce n'est qu'après ce choix initial que l'algorithme peut démarrer. Par la suite, il est nécessaire de refaire les calculs si l'utilisateur souhaite faire des modifications à la définition de l'objet. Pour de longues séquences, ceci génère encore une perte de temps importante.

Ce problème est beaucoup moins important dans l'approche proposée dans [46], où ce n'est pas seulement le masque correspondant à l'objet d'intérêt qui est suivi, mais une partition de texture. La définition de l'objet n'est qu'une « look up table » sur la partition de texture, indiquant quelles régions de cette partition appartiennent à chacun des objets. Cette approche permet en effet de réaliser des modifications sur la définition de l'objet avec une simple modification de la « look up table », ce qui évite de refaire la plupart des calculs. Mais il reste encore une étape de cet algorithme qui doit être recalculée à chaque fois. Cette étape consiste à décider, pour les régions qui n'existaient pas dans la partition fine de la première image et qui apparaissent plus tard dans la séquence, à quel objet elles appartiennent. Cette étape doit donc être recalculée pour chaque nouvelle définition d'objet.

7.3 Hiérarchie associée à une séquence vidéo

Pour palier aux problèmes posés par l'approche par suivi, nous proposons de construire, de la même manière que nous l'avons fait pour les images fixes, une hiérarchie de partitions emboîtées associée à la séquence vidéo [110]. Nous représenterons cette hiérarchie par un arbre aux arêtes valuées, tel que nous l'avons décrit dans le chapitre 6.

Une séquence vidéo contient beaucoup d'information redondante. Les mêmes objets se répètent image après image. De temps en temps de nouveaux objets apparaissent dans la scène, et d'autres disparaissent. L'idée dans l'approche que nous proposons est d'exploiter cette redondance au maximum afin d'obtenir une représentation de la scène aussi simple que possible.

Pour construire une hiérarchie permettant d'exploiter la redondance, il est nécessaire de :
- identifier les régions qui sont les mêmes tout au long de la séquence ;
- établir un ordre global de fusion de ces régions de manière à construire une famille de partitions emboîtées adaptée au contenu de la séquence.

Les régions ne seront plus traitées comme des ensembles connexes de pixels s'étendant uniquement dans l'espace, mais comme des régions spatio-temporelles. Chacune de ces régions est représentée par *un nœud unique* dans l'arbre. La figure 7.1 illustre ce concept : chaque région s'étend dans l'espace et dans le temps, et un seul nœud est utilisé pour représenter chaque région. Les arêtes de l'arbre représentent des relations de voisinage spatio-temporelles. Ainsi, lorsque des ajouts ou éliminations d'arêtes ont lieu sur l'arbre pour accéder aux différents niveaux de la hiérarchie, les partitions obtenues sont des partitions spatio-temporelles (fig. 7.2).

On voit bien la simplification obtenue en représentant la séquence par un arbre dont les nœuds correspondent à des régions spatio-temporelles : si sur la première image de la séquence il y a n régions homogènes (partition fine en n régions), et on suppose qu'il n'y a pas de nouvelles régions qui apparaissent plus tard dans la séquence, alors l'arbre ne contiendra que

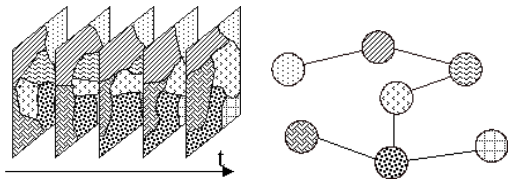


FIG. 7.1 – Les régions spatio-temporelles sont représentées par un seul nœud de l'arbre.

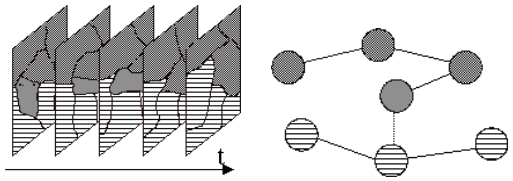


FIG. 7.2 – Les partitions emboîtées sont des partitions spatio-temporelles.

n nœuds, *et ce quelle que soit la longueur de la séquence*. Puisque les interactions de l'utilisateur se traduisent toujours par des manipulations de l'arbre, le faible nombre de nœuds implique que les résultats de l'interaction peuvent être visualisés immédiatement. Dans la réalité, des régions nouvelles apparaîtront au fur et à mesure de l'évolution de la séquence, qui devront être représentées par un nouveau nœud sur l'arbre. De cette manière le nombre de nœuds de l'arbre augmentera par rapport au nombre de nœuds nécessaires pour représenter la première image. Le nombre de nouvelles régions variera selon que l'on traite une séquence avec plus ou moins de mouvement, mais nous verrons dans la section 7.7 qu'en général l'augmentation du nombre de nœuds de l'arbre est suffisamment lente pour ne pas mettre en danger l'utilité de l'algorithme.

La hiérarchie que nous proposons est représentée, de la même manière que pour les images fixes, par :

- une partition fine ;
- un arbre aux arêtes valuées.

La partition fine, comme le montre la figure 7.1, identifie les différentes régions. Deux régions sont considérées comme différentes si elles ont une étiquette distincte sur la partition fine. La partition fine contient l'information sur la *forme des régions*. Elle est stockée sur le disque, car elle est une séquence avec autant d'images que la séquence de départ. L'arbre aux arêtes valuées contient l'information sur la *hiérarchie*, c'est-à-dire l'ordre de fusion des régions. Chaque nœud de l'arbre correspond à une région de la partition fine, et les valeurs des arêtes indiquent l'ordre de fusion des régions.

Cette approche ne présente pas les inconvénients des techniques par suivi, car la définition de l'objet est *postérieure* au calcul de la hiérarchie, qui n'est ainsi réalisé qu'une seule fois. La définition de l'objet, ainsi que toute correction ou modification, se fait par manipulation de l'arbre (ajout/suppression d'arêtes). La représentation en forme d'arbre étant très simple, ces opérations sont réalisées avec un très faible coût en temps de calcul. Les nœuds de l'arbre représentant des régions spatio-temporelles, toute modification faite par l'utilisateur sur une image se traduit par un changement de l'état de l'arbre et affecte automatiquement la séquence complète.

Nous développons dans les sections suivantes les aspects relatifs à la construction de cette

hiérarchie.

7.4 L'approche 3D

Rappelons d'abord l'algorithme de construction de la hiérarchie que nous avons proposé pour les images fixes :

1. pré-filtrage ;
2. calcul de la partition fine :
   - détection des zones λ-plates couleur ;
   - élimination des petites zones plates constituant les régions de transition ;
   - calcul du gradient ;
   - LPE sur le gradient en prenant les zones plates comme marqueurs ;
3. construction du graphe de voisinage associé à la partition fine ;
4. inondation du graphe avec construction de l'arbre aux arêtes valuées.

Voyons maintenant comment il serait possible d'appliquer cet algorithme à des images 3D (obtenues par tronçons d'un volume tridimensionnel). Puisque toutes les opérations que nous utilisons sont des opérations basées sur des relations de voisinage, il est possible d'étendre l'algorithme que nous avons développé pour les images 2D aux images tridimensionnelles avec un simple changement de la grille définissant la connexité. Ainsi, pour le cas tridimensionnel, chaque pixel a des voisins sur la même image, mais aussi sur les plans voisins.

Les figures 7.3 à 7.6 présentent trois niveaux de la hiérarchie pour une image 3D correspondant à des plans d'un cerveau humain. Ici, au lieu de passer par la construction de la partition fine avec inondation du graphe, nous avons simplement appliqué l'algorithme de calcul de hiérarchies par inondation de l'image gradient décrit dans la section 6.3.5 du chapitre 6 en connexité 3D.

En considérant la dimension temporelle comme une troisième dimension spatiale, il est possible de traiter également des séquences vidéo. Des approches 3D de traitement de séquences vidéo ont été proposées dans la littérature [83], [75], [37], permettant de calculer une segmentation associée à une séquence.

L'application de l'approche 3D aux séquences vidéo pose deux problèmes principaux :

- manque de résolution du gradient 3D dans les zones de mouvement ;
- les séquences traitées sont souvent trop longues pour être traitées comme un seul volume 3D.

Regardons ces problèmes un peu plus en détail.

Le problème du manque de résolution temporelle Un opérateur gradient détecte des changements de couleur ou de niveau de gris dans le voisinage d'un pixel. Des valeurs élevées du gradient indiquent donc une frontière. En connexité 2D, les valeurs fortes du gradient indiquent une frontière spatiale. En 3D, cependant, une valeur forte peut indiquer soit une frontière spatiale, soit une frontière temporelle (déplacement d'un objet). Le résultat, ce sont des frontières très larges aux endroits où il y a du mouvement, comme l'illustre la figure 7.7. Quand la LPE est calculée, les contours seront positionnés arbitrairement à l'intérieur de ces larges frontières. Or ce que l'on cherche à détecter ce sont les contours donnant la position et la forme exactes de chaque objet à tout moment, et l'incertitude dans le positionnement des contours réduit considérablement la qualité des résultats.

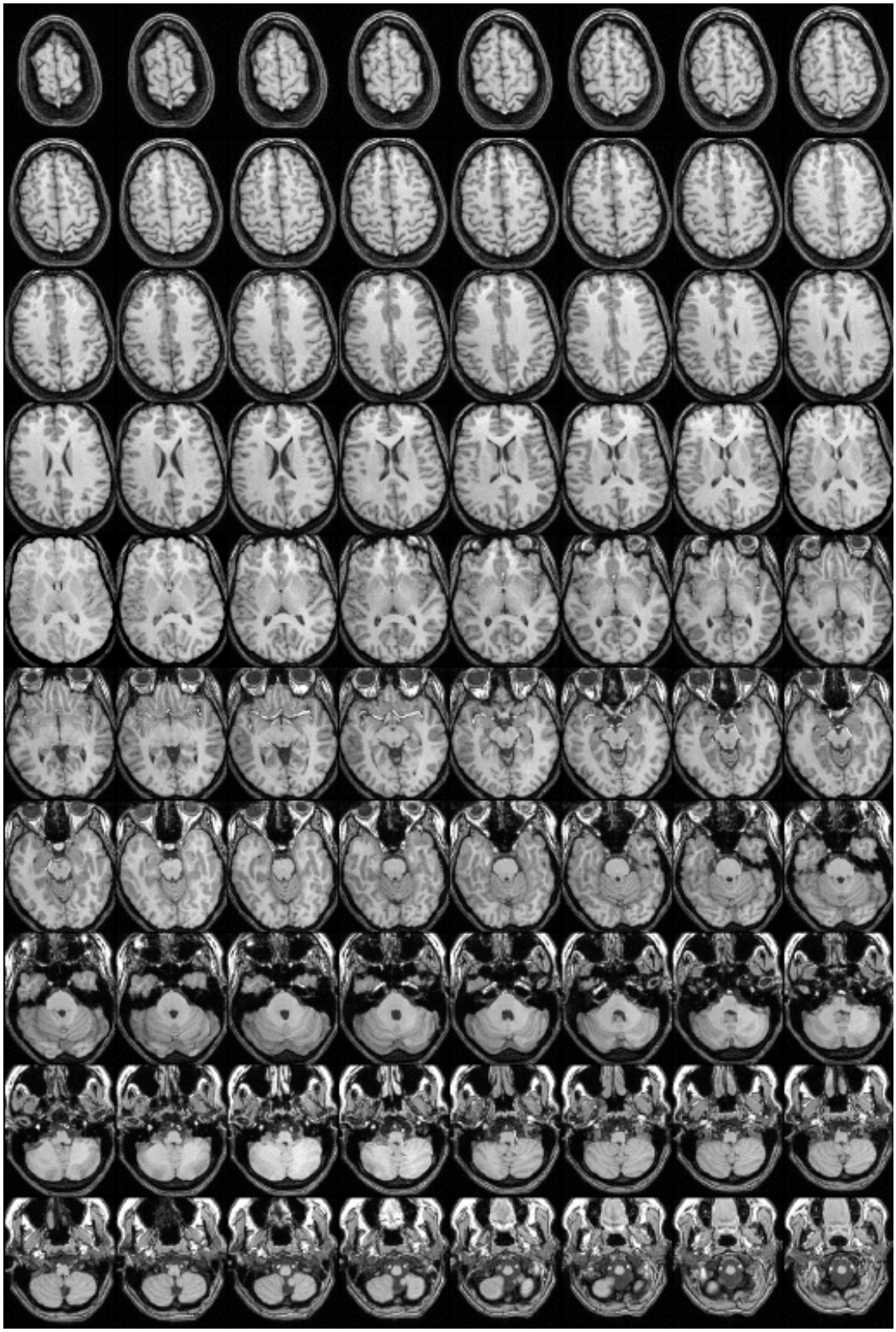


FIG. 7.3 – Série de coupes d'un cerveau humain.

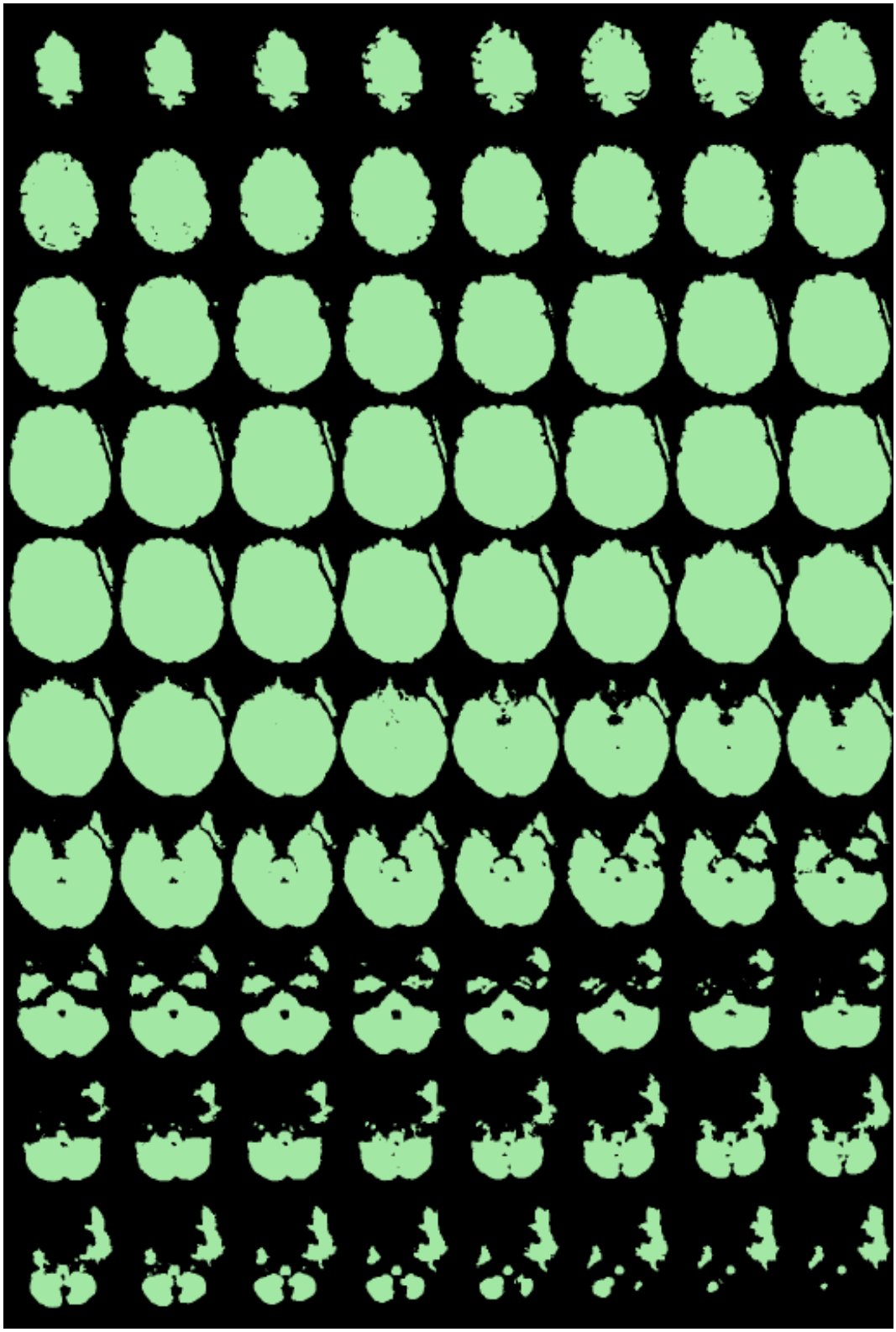


FIG. 7.4 – Segmentation en 2 régions de la hiérarchie de l'image de la figure 7.3.

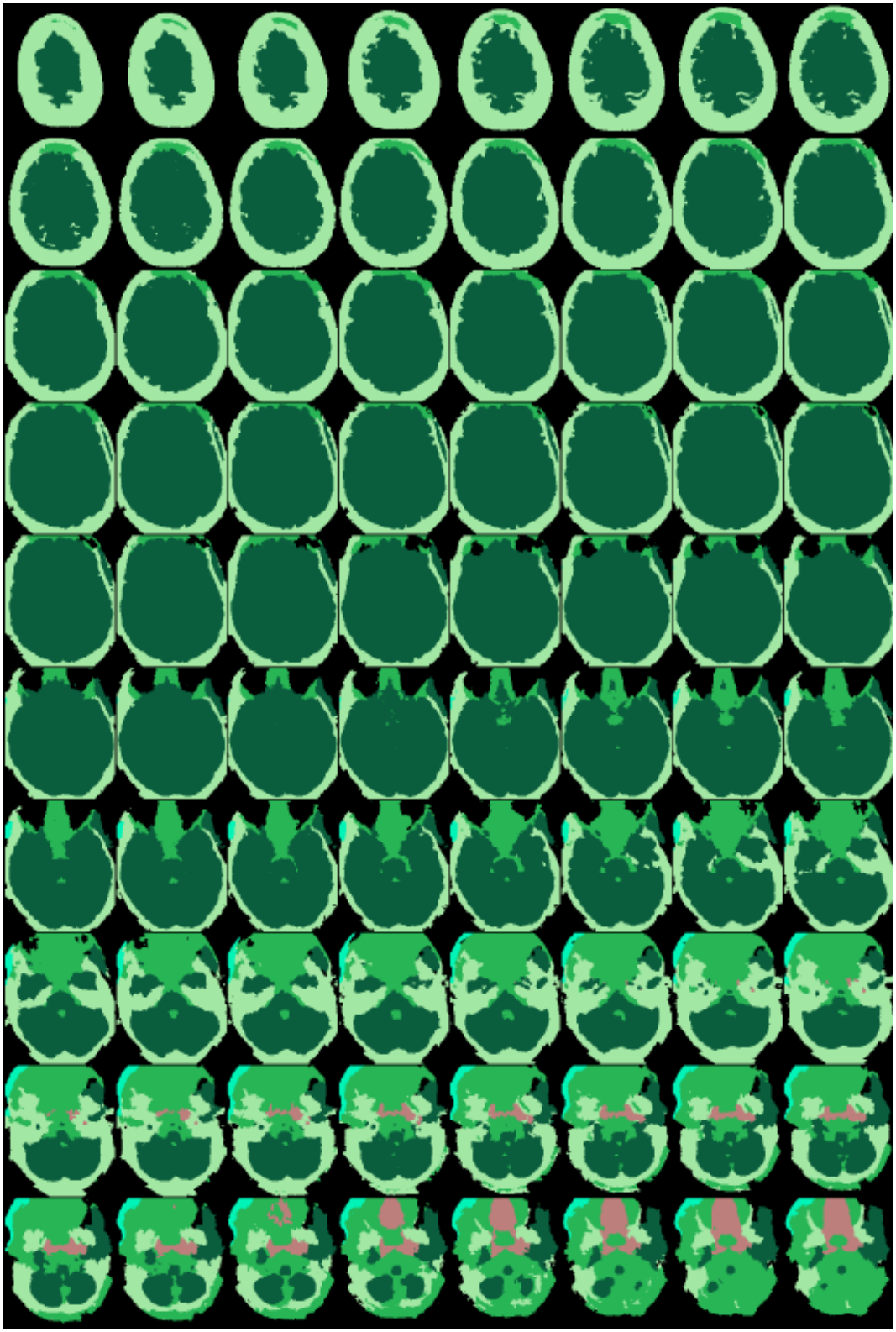


FIG. 7.5 – Segmentation en 10 régions de la hiérarchie de l'image de la figure 7.3.

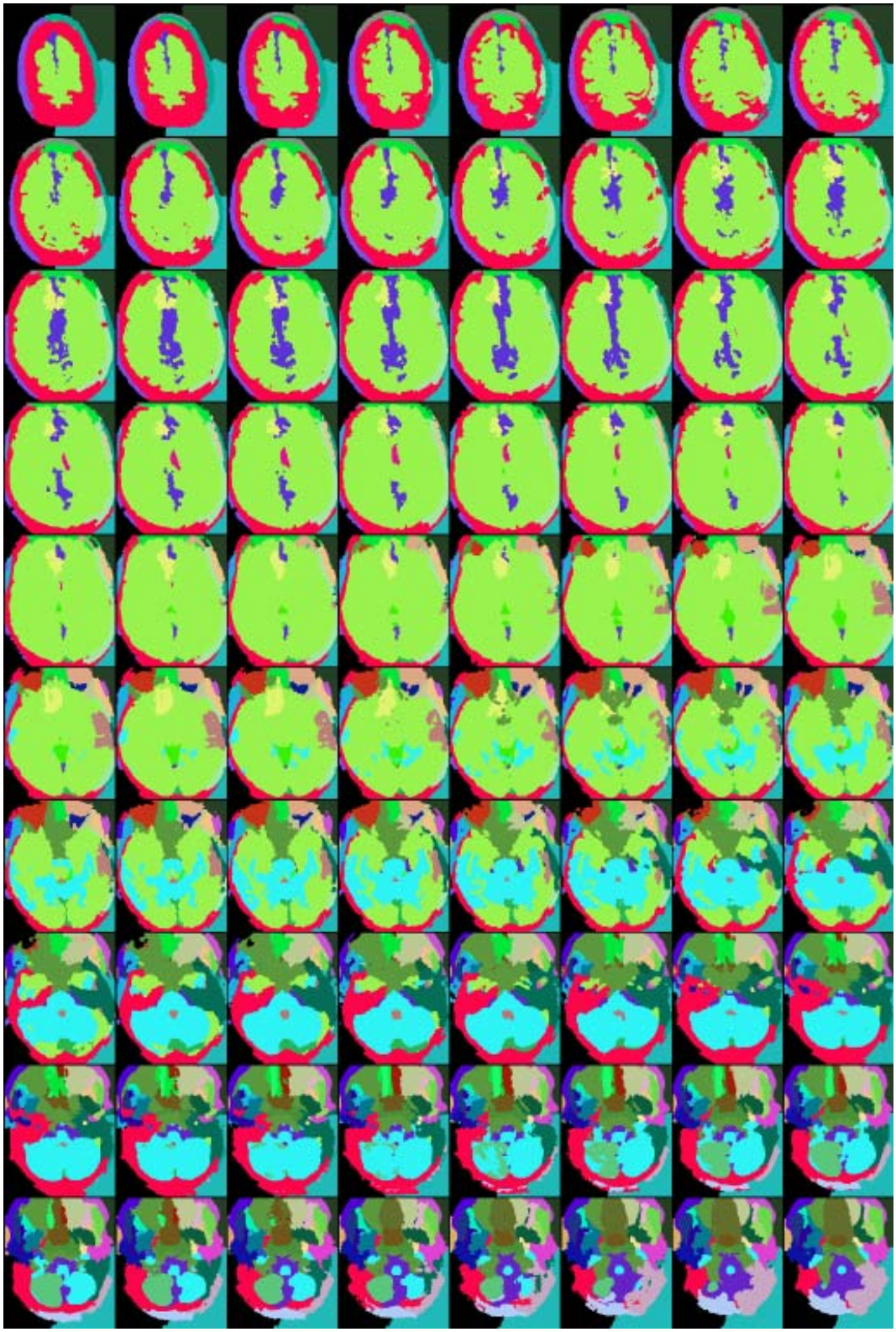
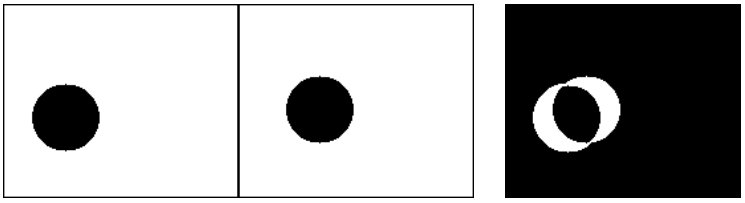


FIG. 7.6 – Segmentation en 50 régions de la hiérarchie de l'image de la figure 7.3.



(a) Image originale (b) Gradient

FIG. 7.7 – Gradient en connexité 3D.

Une solution qui est proposée dans [75] consiste à sur-échantillonner la séquence en espace et en temps par un facteur 2. Avec cette méthode, les frontières spatiales et temporelles du gradient 3D se trouvent sur des images différentes, ce qui permet de les différencier. Cette technique présente cependant l'inconvénient d'augmenter de manière importante le temps de calcul ainsi que les besoins en mémoire vive.

On pourrait également considérer l'utilisation d'un gradient construit en connexité 2D. Mais une telle stratégie peut donner lieu à des *fuites de mouvement* quand une LPE 3D est utilisée pour calculer la partition fine. Ce problème est illustré dans la figure 7.8. Le gradient 2D détectant seulement les frontières spatiales, les déplacement des objets ne sont pas détectés. De cette manière, à cause du déplacement, des pixels à l'intérieur de l'objet où la valeur du gradient est faible sont voisins temporels d'un nombre de pixels appartenant au fond, qui présente également une valeur faible du gradient. Les étiquettes à l'intérieur de l'objet peuvent alors se propager au fond et à l'inverse. Ainsi, dans la figure 7.8, la LPE en 3D ne détecte qu'une seule région, quand deux régions auraient dû être trouvées. En conséquence, l'utilisation de cette approche n'est pas envisageable car elle rend très fréquentes les propagations d'un objet vers un objet différent.

(a) Gradient en connexité 2D (b) Partition par LPE en connexité 3D

FIG. 7.8 – Erreur de propagation causée par l'utilisation d'un gradient calculé en connexité 2D.

Une autre solution consiste à travailler, plutôt que sur une image gradient, sur l'image originale en utilisant une technique de croissance de régions [75], [45], [52]. Cette solution a comme inconvénient de produire des contours plus irréguliers que la LPE.

Nous avons décidé d'avoir recours à une approche mixte combinant la robustesse de la croissance de régions avec la qualité des contours obtenus par LPE que nous détaillons dans la section 7.5. Notre technique est basée sur la détection des zones λ-plates couleur sur l'image originale (filtrée) suivie d'une LPE sur le gradient en prenant les zones plates comme marqueurs. Elle permet d'utiliser un gradient 2D tout en minimisant les fuites. En effet, de telles fuites ne se produisent pas dans le cas des zones plates, car les zones plates ne se propagent que dans des régions de même couleur. Dans notre approche, une fois les zones plates détectées et

les zones plates trop petites éliminées, une grande partie de l'image est déjà couverte – environ 80-90 % –. Ensuite, la LPE n'est utilisée que pour décider de la position exacte des contours dans les 20-10 % restants de l'image. Dans ce cas, la probabilité que ces fuites de mouvement se produisent se voit réduite de façon très importante. De plus, il est possible d'éliminer totalement la possibilité d'avoir des fuites en calculant la LPE plan par plan, c'est-à-dire, en connexité en 2D. Ceci est parfaitement possible parce que ce sont les contours spatiaux que nous cherchons à détecter, les zones plates ayant déjà exploité la redondance temporelle, mais il peut y avoir tout de même une certaine perte de cohérence entre les contours des objets des images successives.

Découpage des séquences Les séquences vidéo sont en général trop longues pour être traitées comme un seul volume 3D, à cause des limitations en mémoire vive des ordinateurs. Dans un contexte pratique nous ne pouvons pas ignorer ce problème. Il est nécessaire de couper la séquence en blocs qui, eux, peuvent être traités comme des images 3D. Mais le découpage d'une séquence fait apparaître un nouveau problème : celui de la mise en correspondance des informations de deux blocs consécutifs. Les informations à mettre en correspondance sont la partition fine et la hiérarchie. Le problème de la mise en correspondance des partitions fines appartenant à deux blocs successifs a été traité par le passé [75], [43]. Or mettre en correspondance des hiérarchies dans le but d'obtenir une hiérarchie unique pour toute la séquence est un problème pour lequel aucune solution n'a, à notre connaissance, été proposée.

7.5 Solution proposée

Nous décrivons ici une technique mixte 3D-récursive pour calculer une hiérarchie associée à une séquence vidéo complète. La séquence est divisée en blocs, et chaque bloc traité comme une image 3D. Nous proposons des mécanismes permettant de mettre en correspondance la partition fine de deux blocs consécutifs et en produire une hiérarchie commune.

7.5.1 Description

Pour le traitement de la séquence, une fenêtre glissante est définie. Cette fenêtre a une longueur de M images et à chaque étape se déplace de $M-1$ images. Ainsi, deux blocs consécutifs ont une image commune (fig. 7.9).

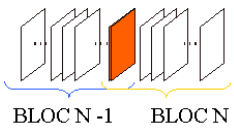


FIG. 7.9 – Deux blocs consécutifs ont une image commune.

Nous imposerons deux contraintes de continuité, qui doivent être vérifiées *sur l'image commune* aux deux blocs :

- la partition fine doit être la même ;
- l'ordre de fusion des régions existantes sur cette image doit être le même pour les deux blocs.

Ces contraintes nous semblent raisonnables, car l'image étant la même pour les deux blocs il est naturel que la partition fine et la hiérarchie coïncident. Remarquons que ces conditions ne doivent être vérifiées que pour l'image commune à deux blocs consécutifs. Pour les autres images, il y aura des régions communes aux deux blocs ainsi que des régions appartenant à seulement l'un d'entre eux, qui correspondent à des régions qui apparaissent/disparaissent au fur et à mesure de l'évolution de la séquence.

Le premier bloc de la séquence n'est pas soumis à ces contraintes, et devra être traité différemment. Nous appellerons le traitement du premier bloc *étape d'initialisation*. Ensuite, le traitement des blocs suivants se fera dans ce que nous appellerons *étape de projection*.

7.5.2 L'étape d'initialisation

Cette étape consiste à calculer la hiérarchie associée au premier bloc de la séquence. Nous avons vu dans la section 7.4 que l'algorithme utilisé pour les images fixes peut être appliqué directement aux images 3D simplement en changeant la connexité, et étendu aux séquences vidéo avec quelques modifications additionnelles.

L'algorithme modifié suit les étapes suivantes :

1. calcul de la partition fine :
   - détection des zones λ-plates couleur en connexité 3D ;
   - élimination des petites zones plates constituant les régions de transition ;
   - calcul du gradient en connexité 2D ;
   - LPE sur le gradient en prenant les zones plates comme marqueurs ;
2. construction du graphe de voisinage associé à la partition fine ;
3. inondation du graphe avec construction de l'arbre aux arêtes valuées.

Comme pour les images fixes, la valeur de λ est calculée de façon à ce que les zones λ-plates résultantes de surface plus grande qu'un certain seuil couvrent un pourcentage de l'image fixé à l'avance et de l'ordre de 80-90 %.

L'étape LPE peut se faire en connexité 2D ou 3D. La portion de l'image à couvrir par LPE étant très petite, le choix de la connexité n'a que peu d'influence sur le résultat. On prendra la connexité 2D si l'on veut un risque zéro de fuites par mouvement, et la connexité 3D si l'on cherche à obtenir des contours très stables dans le temps.

7.5.3 L'étape de projection

L'étape de projection se divise en deux parties :

- d'abord la partition fine du bloc courant est calculée en utilisant les régions du bloc précédent et en détectant de nouvelles régions, ce que nous appellerons *projection de la partition fine* ;
- ensuite, la hiérarchie correspondant à la séquence entière jusqu'au bloc courant est calculée par extension de la hiérarchie obtenue pour le bloc précédent (*projection de la hiérarchie*).

7.5.3.1 Projection de la partition fine

Il s'agit de trouver la partition fine du bloc courant N à partir des informations obtenues pour le bloc précédent $N-1$, et de façon à ce que la partition fine soit la même sur l'image commune aux deux blocs. Cela est fait en plusieurs étapes :

Propagation temporelle des zones plates Dans une première étape, nous prenons les régions de la partition fine de la dernière image du bloc précédent $N-1$. Cette image correspond à l'image commune aux deux blocs, et est la première image du bloc actuel N. Ces régions sont propagées sur les zones λ-plates du bloc N *exclusivement dans la direction temporelle*. La valeur de λ utilisée est celle trouvée dans l'étape d'initialisation. Cette étape met en correspondance les régions du bloc précédent avec les régions du bloc actuel et en même temps impose une certaine stabilité des régions grâce à la propagation temporelle. Remarquons à ce point que, si à cause du fort mouvement des régions se sont temporellement déconnectées, on n'arrivera pas à les détecter. Ces régions apparaîtront comme nouvelles régions dans l'étape suivante. Une étude des régions qui apparaissent et disparaissent pour chaque image pourrait plus tard mettre ces régions en correspondance.

Détection de nouvelles régions Une fois que les régions du bloc $N-1$ ont été propagées dans les zones plates couleur du bloc N, on obtient une image dont une certaine partie est couverte par des régions étiquetées. La détection de nouvelles régions se fait dans la partie de l'image qui n'a pas été couverte lors de l'étape précédente. On détecte les zones λ-plates *complètement incluses* dans la partie non couverte et dont la surface est plus grande qu'un certain seuil. La valeur de λ ainsi que la valeur du seuil de taille sont les mêmes que celles utilisées dans l'étape d'initialisation. Ces régions sont alors identifiées par une nouvelle étiquette. Dans cette étape, les régions temporellement déconnectées apparaissent en tant que nouvelles régions.

Inondation L'ensemble des zones plates obtenues est utilisé comme marqueur pour une LPE sur le gradient. Le gradient, comme pour l'étape d'initialisation et pour les raisons que nous avons expliquées dans la section 7.4, est calculé en connexité 2D. Cette étape permet de trouver les contours exacts des objets.

La figure 7.10 présente un exemple pas à pas, pour un bloc de 5 images. Le bloc d'images est d'abord filtré par nivellement en connexité 3D. Ensuite, les régions de la partition, que l'on connaît pour la première image du bloc, sont propagées en détectant les λ-zones plates dans la direction temporelle, avec la valeur de λ calculée dans l'étape d'initialisation. Remarquons que s'il n'y a pas beaucoup de mouvement, les zones plates ainsi détectées arrivent à couvrir un grand pourcentage de la surface. Les nouvelles régions sont alors détectées aux endroits qui n'ont pas été recouverts par l'étape de propagation. Dans cet exemple, seulement deux nouvelles régions ont été détectées. Ce faible nombre de nouvelles régions détectées démontre la bonne stabilité de l'algorithme, car bien que perceptuellement il n'y a pas de nouveaux objets, les variations de couleur des pixels peuvent être importantes d'une image à l'autre. Finalement, les régions obtenues par propagation temporelle et par détection de nouvelles régions sont utilisées comme marqueurs pour la LPE.

7.5.3.2 Projection de la hiérarchie

Une fois que la partition fine du bloc N a été calculée à partir de celle du bloc $N-1$, il est nécessaire d'étendre au bloc N la hiérarchie existante jusqu'au bloc $N-1$ pour inclure les informations correspondantes au bloc actuel N. Remarquons à ce point qu'il ne s'agit pas de calculer une hiérarchie pour chaque bloc, mais *une hiérarchie globale pour toute la séquence*. L'idée d'une hiérarchie globale est nécessaire au traitement interactif : il n'est pas possible

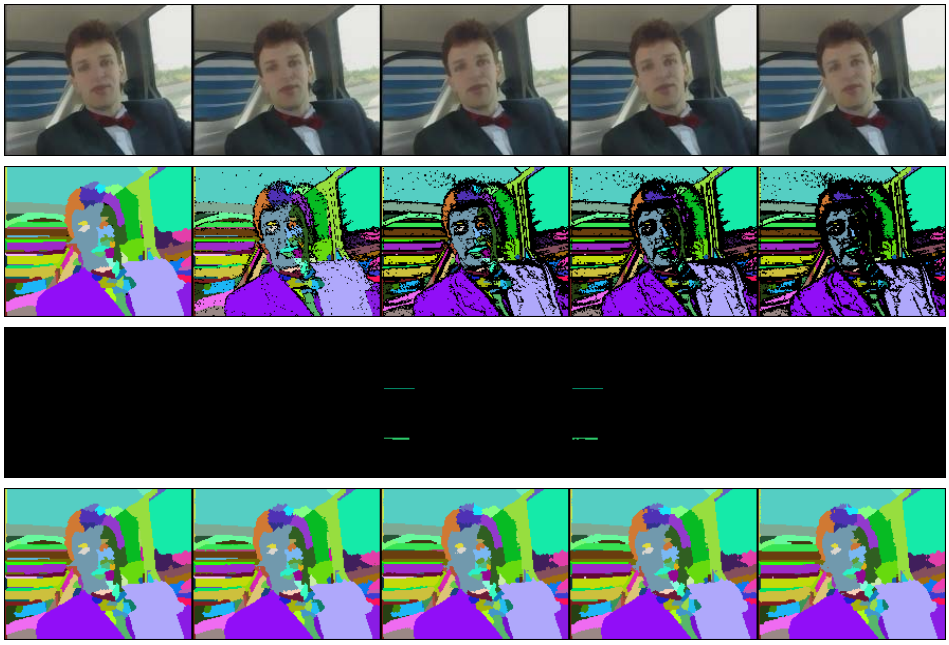


FIG. 7.10 – Projection de la partition fine pas à pas. De haut en bas : bloc d'images filtré, résultat de la propagation temporelle des régions, détection de nouvelles régions et partition fine résultante.

d'oublier des informations passées, car à tout moment l'utilisateur peut vouloir revenir en arrière dans la séquence. Ainsi, et même si dans le bloc actuel il y a des régions qui ont disparu par rapport au premier bloc de la séquence, il est nécessaire que ces régions soient présentes sur l'arbre représentant la hiérarchie.

La hiérarchie jusqu'au bloc $N-1$ est représentée par l'arbre aux arêtes valuées T_{N-1}. Les nœuds de cet arbre représentent la totalité des régions spatio-temporelles qui sont apparues depuis le début de la séquence et jusqu'au bloc $N-1$. Ces régions sont hiérarchisées par les valeurs des arêtes de T_{N-1}.

Étant donnés T_{N-1} et la partition fine associée au bloc courant P_N, l'objectif est de trouver T_N, qui représentera la hiérarchie de la séquence jusqu'au bloc N. Rappelons que nous nous sommes imposés la contrainte suivante (cf. section 7.5.1) : pour les régions de l'image commune aux blocs $N-1$ et N, l'ordre de fusion exprimé par les valeurs des arêtes des arbres T_{N-1} et T_N doit être le même. Remarquons que cette contrainte ne dit rien sur les régions qui n'existent pas sur cette image.

Notre solution pour le calcul de T_N à partir de T_{N-1} consiste à garder la structure et les valeurs d'arête de l'arbre T_{N-1}. Ensuite, T_{N-1} est actualisé avec les nouvelles régions, apparues au bloc N, pour donner T_N. Cette option satisfait la contrainte que nous nous sommes imposés puisque les régions sur l'image commune existent sur T_{N-1}, et pour ces régions nous gardons précisément la hiérarchie donnée par cet arbre. Ainsi, pour calculer T_N nous devons ajouter à T_{N-1} des nœuds correspondant aux nouvelles régions. Pour ce faire, nous devons déterminer :

- les chemins de connexion des nouveaux nœuds aux nœuds de T_{N-1} ;
- les valeurs des arêtes reliant les nouveaux nœuds.

La figure 7.11 présente de manière schématique la projection de la hiérarchie, où les nœuds

ombragés représentent des régions qui sont communes au bloc précédent et au bloc actuel, les nœuds en noir correspondent à de nouvelles régions, et les nœuds en blanc représentent des régions qui ont disparu pour le bloc courant N. L'arbre T_N est obtenu en connectant les régions apparues au bloc N à l'arbre T_{N-1}. Pour cela, il y a plusieurs chemins possibles, marqués en pointillé dans la figure, entre lesquels il faut choisir.

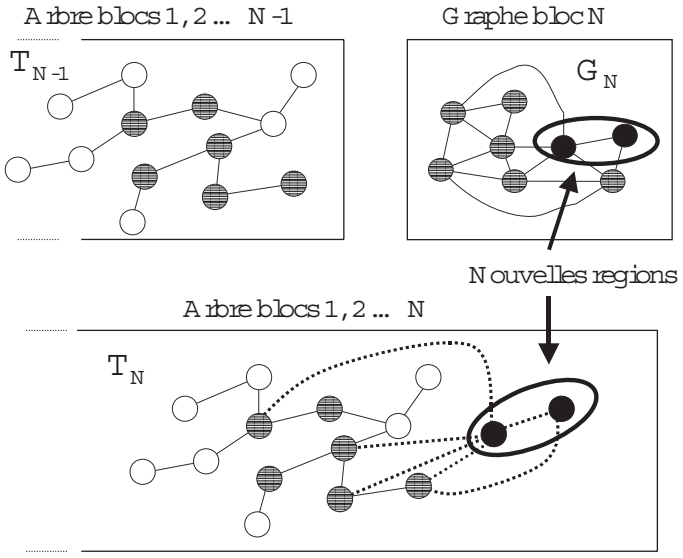


FIG. 7.11 – Projection de la hiérarchie.

Les valeurs des arêtes sont très importantes, car elles détermineront la position des nouvelles régions dans la hiérarchie. Ces valeurs doivent refléter l'importance visuelle des régions. Des valeurs trop élevées placeraient des petites régions en haut de la hiérarchie, au détriment de régions plus importantes. Des valeurs trop petites conduiraient à sous-estimer des régions visuellement importantes.

Afin de proposer une stratégie de calcul de ces paramètres, remarquons les faits suivants à propos de la hiérarchie sur les images fixes :

- les chemins de connexion entre les nœuds sont déterminés par les chemins minimaux ;
- les valeurs des arêtes sont données par le volume – ou autre mesure géométrique – des régions au moment où elles fusionnent avec une région voisine à travers le chemin minimum.

Nous avons montré (cf. chapitre 6, section 6.4.2), que ces calculs peuvent se réaliser sur le graphe de voisinage de la partition fine, moyennant un processus d'inondation du graphe : les arêtes sont ajoutées par ordre croissant de valeurs et, à chaque fois qu'une arête est ajoutée, le volume des deux composantes connexes (ou régions) est calculé sur le graphe et le plus petit donne la valeur de l'arête. L'ajout d'arêtes par ordre croissant de valeur permet de trouver les chemins minimaux entre les nœuds. La mesure géométrique au moment de la rencontre permet d'établir une hiérarchie significative.

Pour obtenir les chemins minimaux – ainsi que les valeurs des arêtes – pour les nouvelles régions, nous utilisons un algorithme similaire, que nous appellerons *inondation partielle* du graphe. À la différence de l'algorithme d'inondation du graphe du chapitre 6, ici nous ne traitons pas toutes les arêtes du graphe de voisinage, mais seules celles ayant comme extrémité

au moins un nœud correspondant à une nouvelle région. Ceci équivaut à inonder seulement les parties du graphe correspondant à de nouvelles régions.

Initialement, nous disposons de la partition fine du bloc actuel, P_N, ainsi que de l'arbre T_{N-1}. Sur P_N il y a des régions qui sont déjà représentées par un nœud sur T_{N-1}, que nous appellerons *anciennes régions* et d'autres qui ne le sont pas, auxquelles nous nous référerons en tant que *nouvelles régions*. Nous désignerons leurs nœuds respectifs comme *anciens nœuds* et *nouveaux nœuds*. L'algorithme 7.1 décrit la projection. Le graphe de voisinage G_N associé à la partition fine P_N du bloc actuel est d'abord créé. L'arbre T_{N-1} est actualisé en lui ajoutant les nouveaux nœuds de G_N, mais sans les connecter. Le graphe ainsi obtenu constitue une version « en construction » de l'arbre final T_N. Par souci de simplicité nous appelons ce graphe T_N. Ensuite, les arêtes de G_N ayant comme extrémité l'un des nouveaux nœuds sont traitées par ordre croissant de valeur. Remarquons qu'il n'est pas nécessaire de traiter les arêtes reliant des anciens nœuds, car pour ces nœuds nous avons décidé de garder les connexions et les valeurs d'arête de l'arbre T_{N-1}. À chaque fois qu'une arête ne formant pas de cycle est extraite, elle est ajoutée à T_N. Il faut alors lui donner une valeur. Si les composantes connexes des deux extrémités de l'arête sont formées exclusivement par de nouveaux nœuds, c'est-à-dire, si aucun des nœuds n'a encore été relié à l'arbre principal, alors leurs volumes sont comparés et le plus petit est choisi comme valeur d'arête. Autrement, c'est le volume de la composante connexe formée exclusivement de nouveaux nœuds qui est utilisé pour valuer l'arête.

La figure 7.12 présente un exemple pas à pas. La partition P_N contient des régions qui existaient déjà dans P_{N-1}, plus trois nouvelles régions. Ces régions sont indiquées dans la figure sur le graphe de voisinage G_N associé à la partition P_N. D'autres régions qui existaient sur P_{N-1} ont disparu dans la partition P_N, mais remarquons que ces régions seront encore présentes dans l'arbre T_N. L'arbre T_{N-1} représente la hiérarchie jusqu'au bloc $N-1$. Les valeurs de ses arêtes donnent l'ordre de fusion des différentes régions. Le graphe G_N donne les relations de voisinage entre les régions de P_N. Ses arêtes ont été valuées avec une mesure de dissimilarité entre régions voisines, typiquement la différence entre les moyennes de couleur des régions. Les arêtes en gras sur G_N sont celles ayant au moins un nouveau nœud comme extrémité, et sont celles qui sont introduites dans la file d'attente, par ordre croissant de valeur. Les nouveaux nœuds ont une mesure associée correspondant à leur surface. Cette mesure sera nécessaire à l'estimation du volume des régions. L'arbre T_N est construit à partir de T_{N-1} en lui ajoutant dans un premier temps les nouveaux nœuds, mais sans les connecter. Ensuite, l'arête de valeur plus faible (valeur 1) est extraite de la file d'attente. Comme elle n'introduit pas de cycle, elle est ajoutée à T_N. Puisque cette arête relie deux nouveaux nœuds, sa valeur sera donnée par le volume le plus petit des composantes connexes des deux côtés de l'arête (dans ce cas chaque composante connexe contient un seul nœud). Les deux composantes connexes ont une surface de 10 et 5 respectivement, et la valeur de l'arête est de 1. Ainsi, l'arête obtient une valeur de $5 \cdot 1 = 5$. Cette arête est supprimée de la file. Pour souci de clarté, nous avons marqué, sur la file d'attente, les arêtes ajoutées à l'arbre avec un A, et celles ignorées avec un I. La prochaine arête à sortir de la file est celle à valeur 2. Cette arête n'ajoute pas de cycle, et elle est donc ajoutée. Comme elle relie une composante connexe contenant seulement des nouveaux nœuds avec la partie contenant des anciens nœuds, elle est valuée avec le volume de la composante connexe contenant seulement des nouveaux nœuds. Ce volume est calculé comme la surface de la nouvelle région, $(10+5)$, multipliée par la valeur de l'arête sur G_N : $(10+5) \cdot 2 = 30$. L'arête est supprimée de la file d'attente, et les arêtes suivantes sont traitées. Les arêtes de valeurs 3 à 6 ne peuvent pas être ajoutées, car elles introduiraient un cycle sur l'arbre, et elles sont donc ignorées. L'arête de valeur 7 n'introduit pas de cycle, et elle est

Algorithme 7.1 Algorithme de projection de la hiérarchie.

T_{N-1} : arbre représentant la hiérarchie jusqu'au bloc $N-1$

P_N : partition fine associée au bloc N

Créer le graphe de voisinage G_N associé à la partition P_N

$T_N \leftarrow$ ajouter à T_{N-1} les nouveaux nœuds de G_N, sans les connecter

pile $\leftarrow$ arêtes de G_N ayant au moins un nouveau nœud par extrémité, ordonnées par ordre croissant de valeur

while NoVide(Pile) **do**

  $e_{ij} \leftarrow$ ExtraireArête(Pile)

  $v_i, v_j \leftarrow$ TrouverExtrémités(e_{ij})

  if arête e_{ij} ne forme pas un cycle dans T_N **then**

    if ComposanteConnexe(v_i) et ComposanteConnexe(v_j) sont formées exclusivement de nouveaux nœuds **then**

      AjouterArête(T_N, e_{ij})

      ValuerArête(e_{ij}, MIN(Volume(v_i), Volume(v_j)))

    else {Seulement une des deux composantes connexes est formée exclusivement de nouveaux nœuds}

      AjouterArête(T_N, e_{ij})

      if ComposanteConnexe(v_i) est formée de nouveaux nœuds **then**

        ValuerArête(e_{ij}, Volume(v_i))

      else {C'est l'autre composante connexe}

        ValuerArête(e_{ij}, Volume(v_j))

      end if

    end if

  end if

end while

ajoutée à l'arbre. Cette arête relie un nouveau nœud avec un ancien nœud. Le volume du nouveau nœud est maintenant $7 \cdot 3 = 21$, et c'est cette valeur qui est associée à l'arête de l'arbre. La dernière arête à sortir de la file d'attente est celle de valeur 8, mais elle est ignorée car elle introduit un cycle. La file est maintenant vide, et la construction de l'arbre T_N est terminée. L'arbre T_N représente maintenant la hiérarchie du début de la séquence jusqu'au bloc courant N.

7.6 Résultats

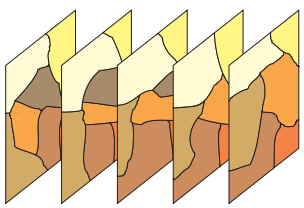
Les figures 7.13 à 7.20 présentent des résultats obtenus avec la méthode proposée pour les séquences « Mother & Daughter », « Carphone » et « Car ». Afin de limiter l'espace, nous ne montrons qu'une image sur dix, mais toutes les images de la séquence ont été traitées. Chaque ligne montre l'image originale, la partition fine et trois niveaux, de plus en plus grossiers, de la hiérarchie. Ces segmentations correspondent à des niveaux de la hiérarchie et ont été donc obtenues de façon complètement automatique et sans aucun type d'interaction. Nous évaluons maintenant la qualité de la hiérarchie, sans interaction. Des résultats avec interaction seront présentés dans la section 7.8. La séquence « Mother & Daughter » a un mouvement léger, tandis que les séquences « Carphone » et « Car » présentent un mouvement beaucoup plus important. Remarquons dans tous les cas l'absence de fuites significatives dans la partition fine. La bonne qualité de la partition fine est fondamentale, car si des fuites apparaissent à ce niveau, il ne sera pas possible de les corriger moyennant interaction. Un autre aspect à noter est la bonne hiérarchisation globale des régions. En effet, lorsqu'il y a du mouvement dans la séquence, de nouvelles régions apparaissent dans la partition fine, tandis que d'autres disparaissent. Ces régions sont correctement hiérarchisées, car ces changements ne sont reflétés dans les niveaux plus grossiers que quand les nouvelles régions sont suffisamment importantes en termes de taille et de contraste.

Temps de calcul Nous donnons ici quelques temps de calcul de la hiérarchie afin de donner une idée approximative de sa complexité. Ce temps de calcul doit être considéré comme une valeur maximale, car nous n'avons appliqué aucun type d'optimisation aux algorithmes. Une optimisation appropriée devrait permettre de le réduire considérablement.

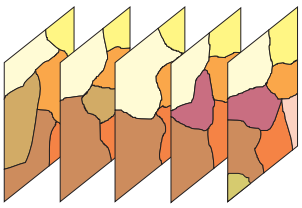
Nous avons traité des séquences de 300 images QCIF (176 x 144 pixels), sur un Pentium III à 800MHz. Avec cette configuration, nous avons obtenu un temps de traitement moyen de 1.43 secondes par image. Un exemple de répartition des temps de traitement est le suivant (pour un bloc de 5 images) :

- pré-filtrage : 2.69 secondes ;
- calcul du gradient : 0.16 secondes ;
- projection de la partition fine : 1.60 secondes ;
- construction du graphe associée à la partition fine : 0.77 secondes ;
- projection du graphe : 0.05 secondes ;

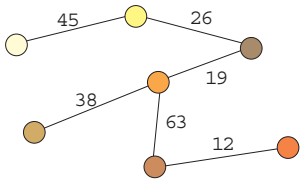
Cette répartition permet de désigner l'opération consommatrice de plus de la moitié du temps de calcul : le pré-filtrage. Dans cette opération, c'est l'obtention du marqueur qui prend le plus de temps, car il est obtenu en appliquant un filtre alterné séquentiel, ce qui oblige à itérer l'opération plusieurs fois. On peut alors se demander si cette étape vaut vraiment sa consommation en temps de calcul. Afin de le déterminer, nous avons appliqué l'algorithme avec et sans préfiltrage à plusieurs images et nous avons comparé les résultats. Le filtre utilisé



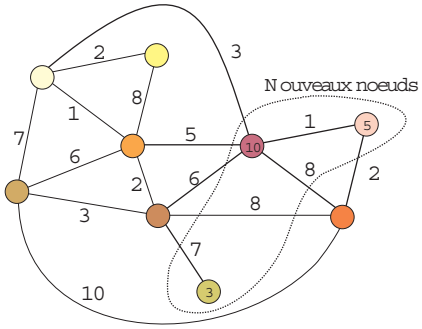
Partition P_{N-1}



Partition P_N



Arbre T_{N-1}



Graphe G_N

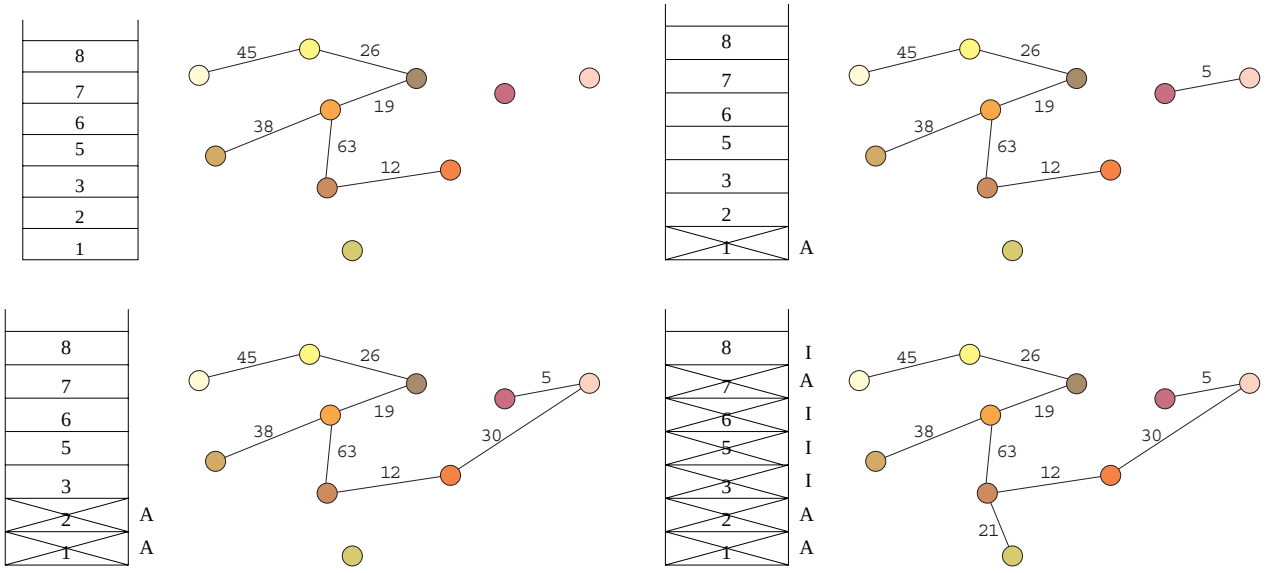


FIG. 7.12 – Exemple pédagogique illustrant la projection de la hiérarchie.

FIG. 7.13 – Résultats pour la séquence « Mother & Daughter » (images 1 à 100).

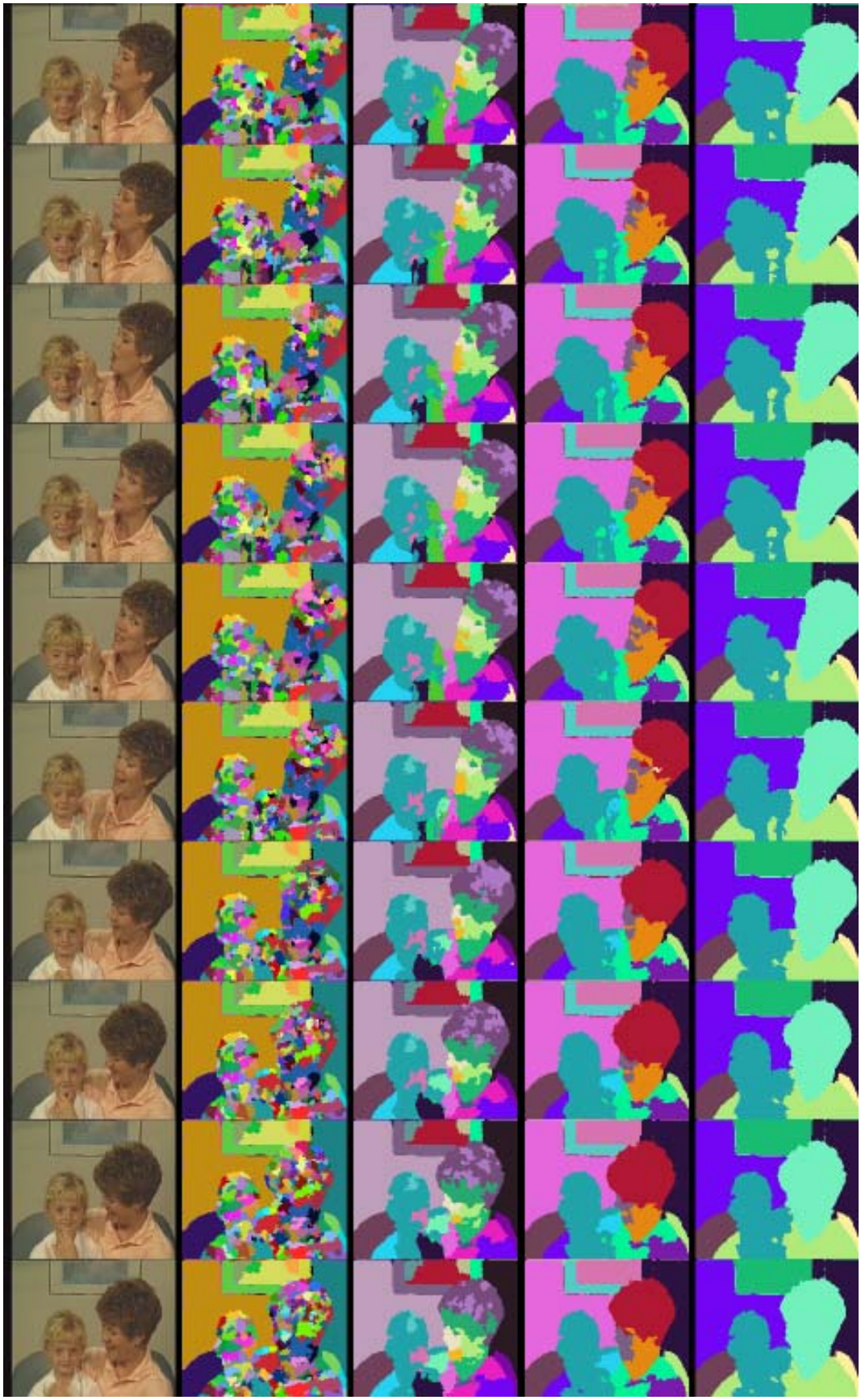


FIG. 7.14 – Résultats pour la séquence « Mother & Daughter » (images 100 à 200).

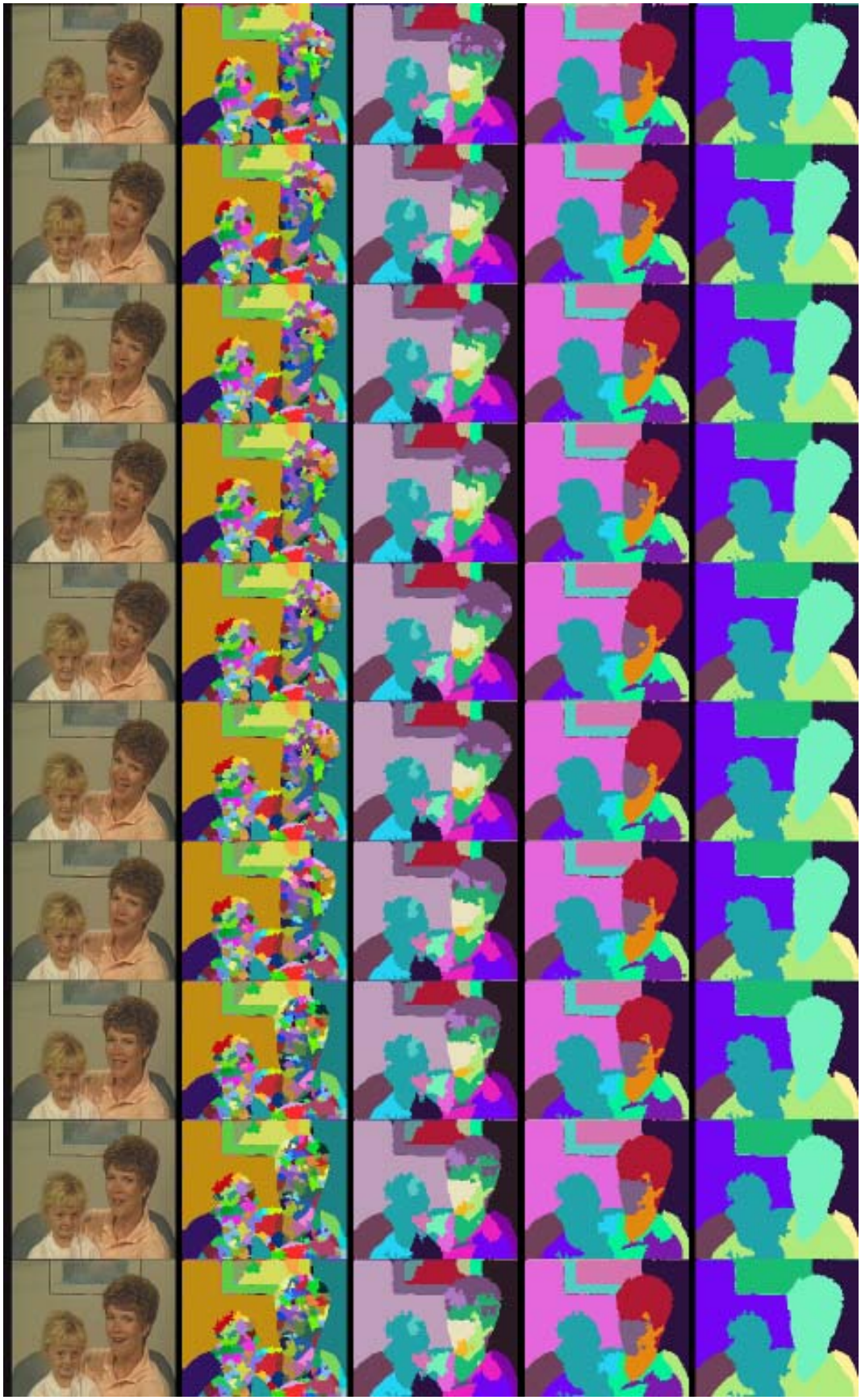


FIG. 7.15 – Résultats pour la séquence « Mother & Daughter » (images 200 à 300).

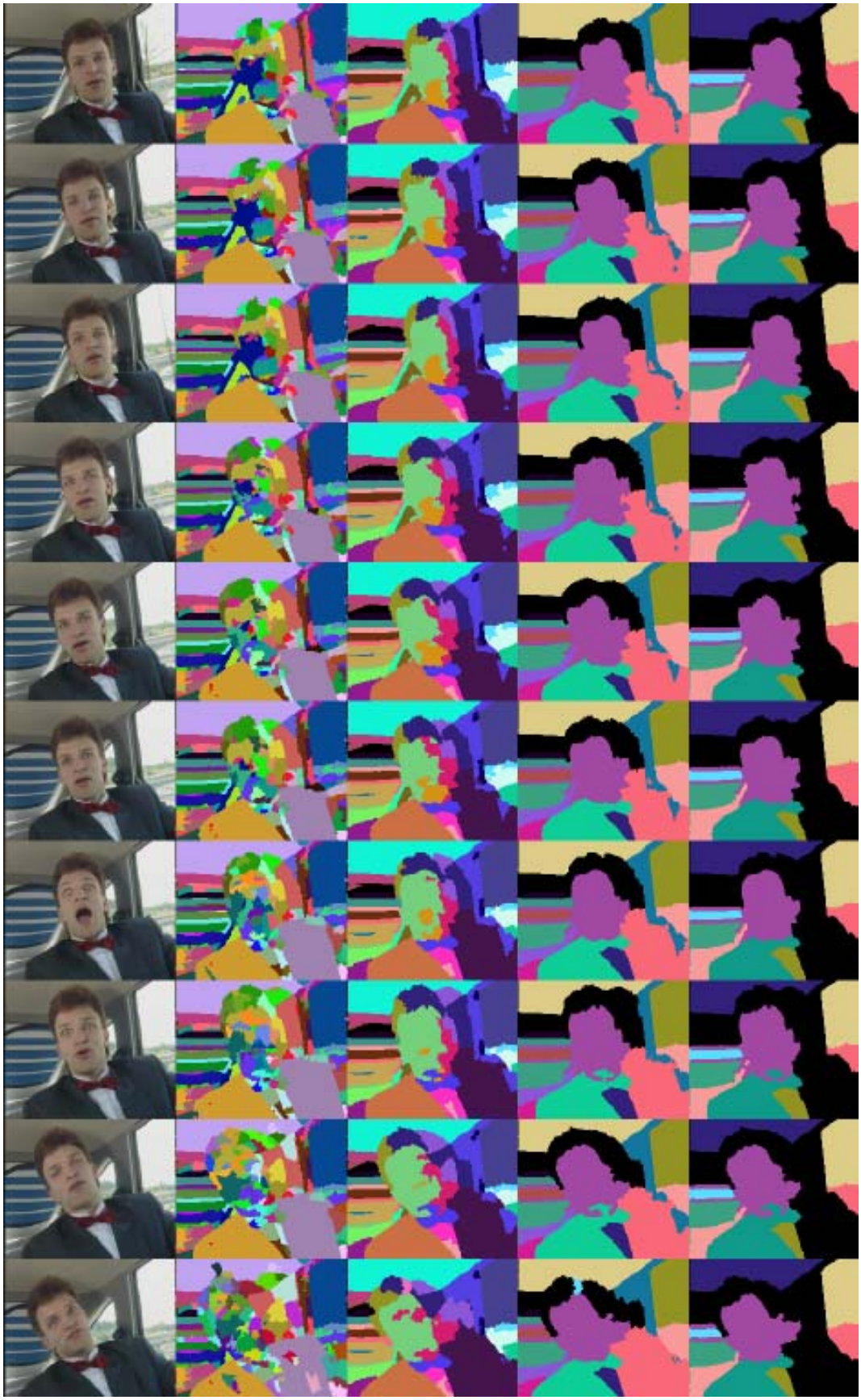


FIG. 7.16 – Résultats pour la séquence « Carphone » (images 1 à 100).

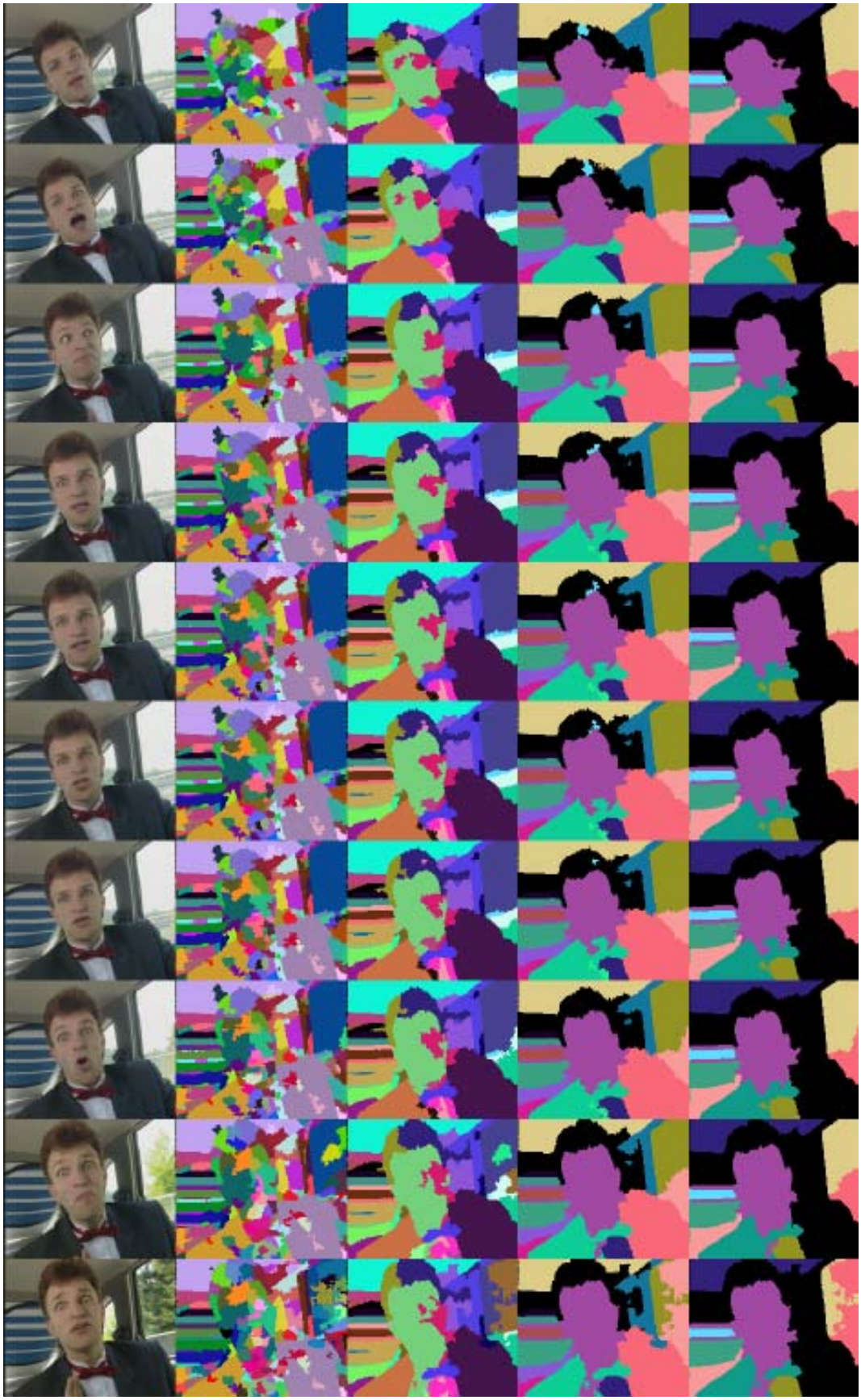


FIG. 7.17 – Résultats pour la séquence « Carphone » (images 100 à 200).

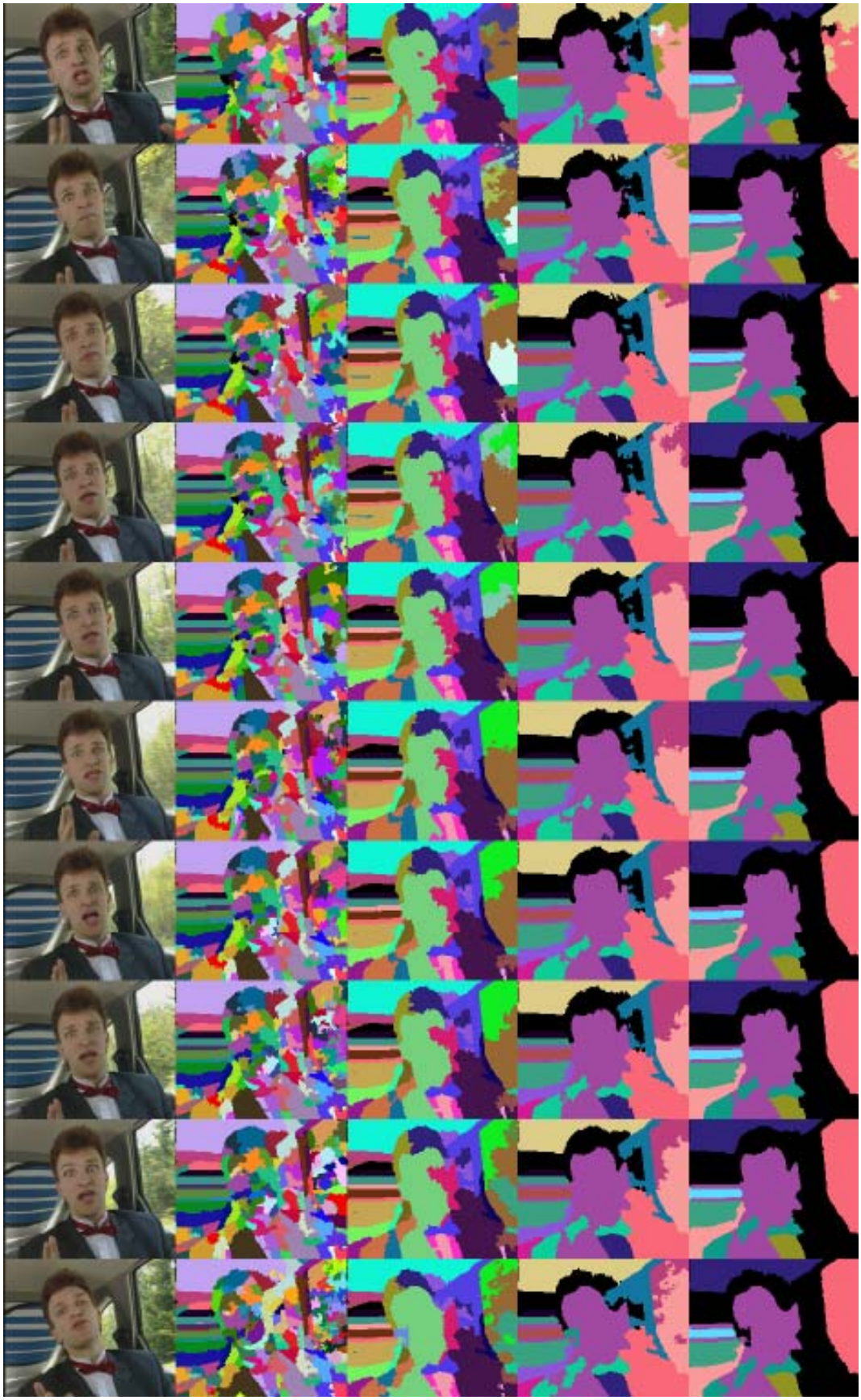


FIG. 7.18 – Résultats pour la séquence « Carphone » (images 200 à 300).

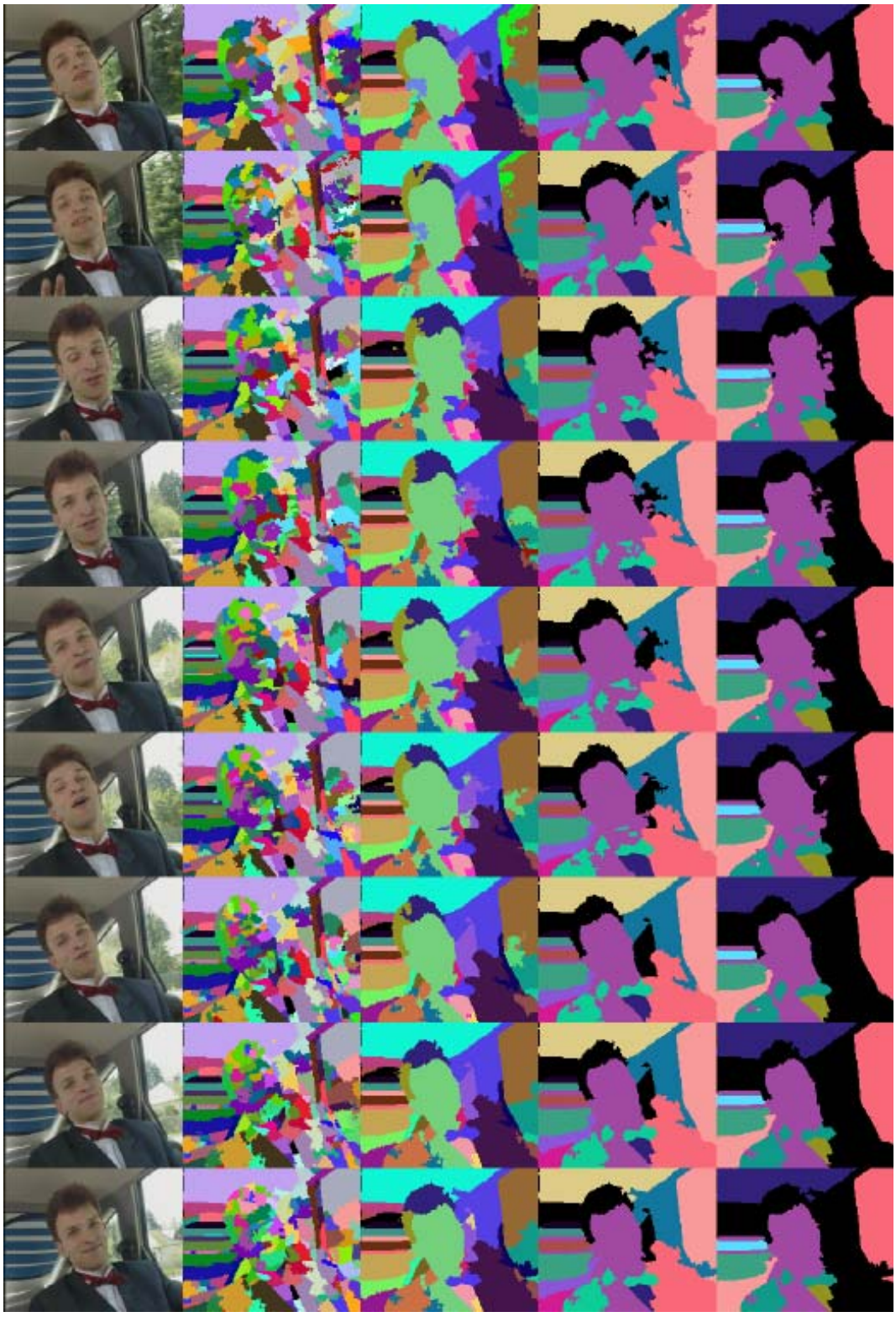


FIG. 7.19 – Résultats pour la séquence « Carphone » (images 300 à 380).

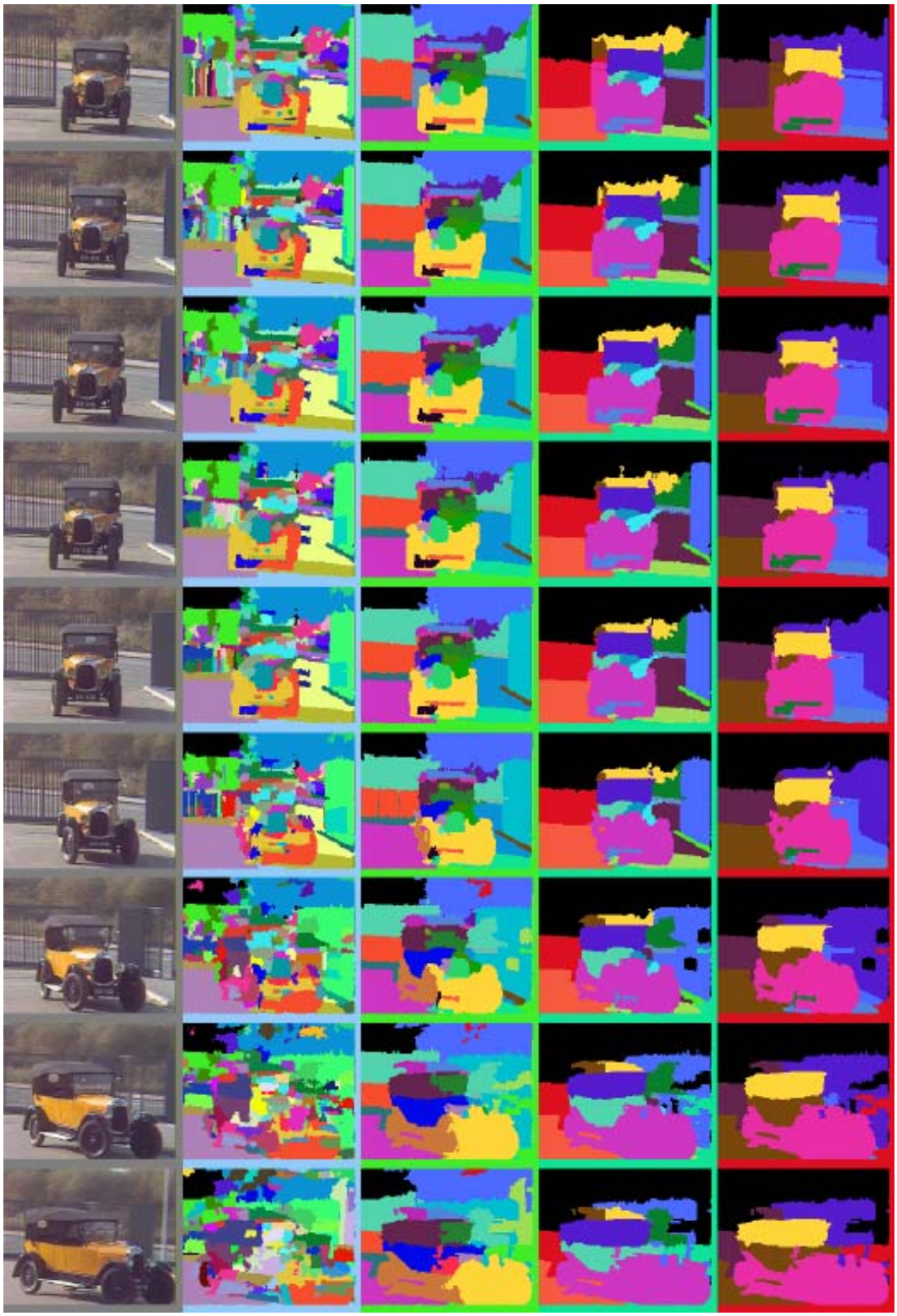


FIG. 7.20 – Résultats pour la séquence « Car » (images 1 à 90).

est une reconstruction par nivellement d'un marqueur obtenu par filtre alterné séquentiel de taille 3. La figure 7.21 illustre les résultats obtenus avec et sans pré-filtrage pour les séquences « Mother & Daughter » et « Silent Voice ». Pour la séquence « Mother & Daughter », aucune des deux partitions fines ne présente de fuites. Cependant, la qualité de segmentation des petits objets (les yeux et la bouche, par exemple) est meilleure pour la version filtrée. La qualité du niveau de la hiérarchie avec 10 régions est également meilleure pour la version avec pré-filtrage (pas de fusion de la chemise avec le fond). Pour la séquence « Silent Voice », les différences sont plus importantes. En effet, la partition fine obtenue sans pré-filtrage a une fuite (fusion du bras avec le fond) qui n'apparaît pas dans la version avec pré-filtrage. Une fuite à ce niveau a des conséquences graves, car elle apparaîtra dans tous les niveaux de la hiérarchie. En général, le pré-filtrage améliore les résultats, l'amélioration étant d'autant plus importante quand les scènes traitées sont bruitées ou texturées. Pour des scènes non bruitées ou sans texture importante, et quand il y a des soucis de temps d'exécution, il est possible d'éliminer cette étape, mais en acceptant une possible perte de qualité des partitions résultantes. Puisque ce n'est pas le nivellement en lui même qui consomme en temps de calcul, mais l'obtention du marqueur, une solution de compromis consiste à prendre comme marqueur d'autres types de filtres, d'exécution plus rapide. Comme l'étape de nivellement reconstruira les contours, le filtre choisi n'a pas à les respecter. Ainsi, des filtres fréquentiels passe-bas où des filtres par ondelettes pourront être utilisés pour obtenir l'image marqueur.

7.7 Discussion

7.7.1 Évolution du nombre de nœuds

Notre méthode propose de construire un arbre unique pour représenter une séquence complète. Sur cet arbre, des nœuds sont ajoutés pour représenter les nouvelles régions au fur et à mesure que la séquence évolue, tandis que les régions qui disparaissent ne sont pas éliminées. Ainsi, le nombre de nœuds augmente avec le nombre d'images traitées. Le nombre de nouvelles régions qui apparaissent pour chaque bloc dépend de chaque séquence, et est plus grand si la séquence présente un fort mouvement. On peut s'interroger sur le rythme d'augmentation du nombre de nœuds, car une augmentation trop rapide rendrait la manipulation de l'arbre lourde du point de vue du temps de calcul, et éliminerait ainsi une partie des avantages de notre système pour la segmentation interactive. La figure 7.22 montre l'évolution du nombre de nœuds pour les séquences « Carphone », « Mother & Daughter » et « Silent Voice ». La séquence a été traitée par blocs de 5 images. Comme il y a une image commune entre deux blocs différents, la taille effective du bloc est de 4 images. Ainsi, par exemple, le début du bloc 50 correspond à l'image 200 de la séquence. Le nombre total de régions est naturellement différent pour chaque séquence. Le nombre de régions de la partition fine (nombre de nœuds de l'arbre) dépend de la complexité de la scène, ainsi que de l'intensité du mouvement. La séquence « Mother & Daughter » étant à la fois peu texturée et n'ayant pas beaucoup de mouvement, elle montre un plus faible nombre de régions. La séquence « Carphone » présente un très fort mouvement, mais elle n'est pas très texturée. Finalement, la séquence « Silent Voice » présente à la fois les deux caractéristiques : elle a un fond très texturé, et le mouvement de la locutrice est très important, ce qui explique le plus grand nombre de régions dans la plupart de la séquence. On voit que, dans tous les cas, au bout de 300 images le nombre de nœuds n'atteint pas 2000, ce qui représente un arbre de taille très raisonnable et constitue une simplification très remarquable par rapport au nombre total de pixels constituant la séquence.

FIG. 7.21 – Résultats obtenus pour le deuxième bloc des séquences « Mother & Daughter » et « Silent Voice » (images 4 à 8), avec et sans pré-filtrage. De haut en bas : images originales, partition fine obtenue sans pré-filtrage, partition fine obtenue avec pré-filtrage, niveau de la hiérarchie avec 10 régions pour la hiérarchie obtenue sans pré-filtrage, idem pour la hiérarchie obtenue avec pré-filtrage.

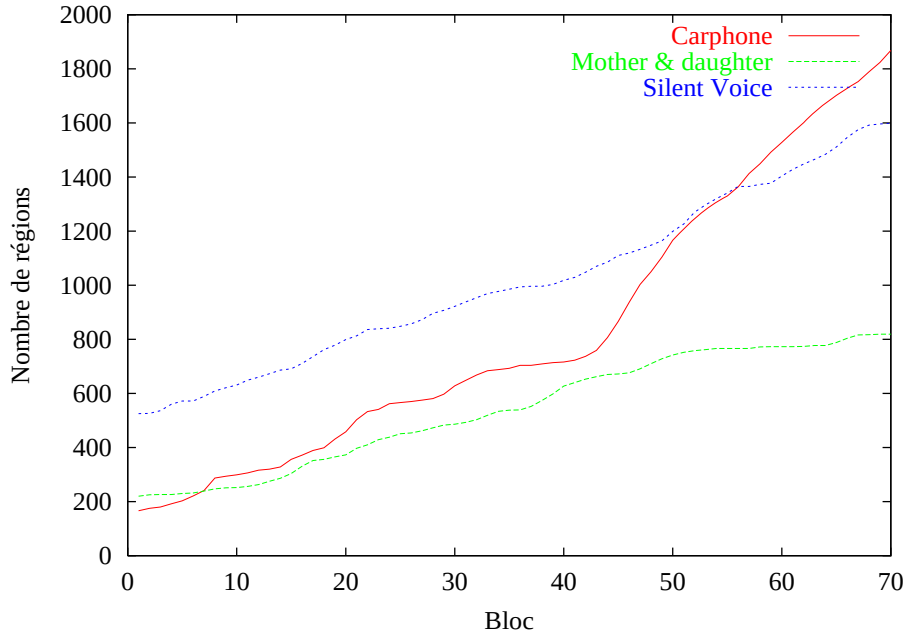


FIG. 7.22 – Évolution du nombre de régions pour les séquences « Carphone », « Mother & Daughter » et « Silent Voice ».

7.7.2 Régions non connexes dans l'espace

Notre approche 3D de la segmentation de séquences définit une région comme un ensemble de pixels connexes *dans l'espace spatio-temporel*. Mais une région connexe en 3D peut ne pas l'être en connexité 2D. C'est-à-dire, quand on regarde la séquence image par image, il est possible de trouver des régions avec la même étiquette et qui ne sont pas connexes dans l'espace. Ceci est une conséquence de la définition de région que nous avons adoptée. Cette définition est conceptuellement correcte : en effet, dans le cas d'une occlusion partielle, un même objet peut être divisé en deux parties non connexes dans l'espace, et il est intéressant d'être capable de les identifier comme faisant partie d'un même objet. La séquence « Mother & Daughter » offre un exemple de cette situation : vers l'image 60 de la séquence, la tête de la mère touche le bord de l'image, divisant le fond en deux régions non connexes. Notre technique détecte ces deux composantes connexes comme faisant partie d'un objet unique. À l'inverse, deux régions initialement non connexes qui fusionnent plus tard dans un objet unique ne seront pas fusionnées, même si elles ont des couleurs très proches. Ceci ne suppose pas un problème, car au niveau de la partition fine une certaine sursegmentation est parfaitement acceptable.

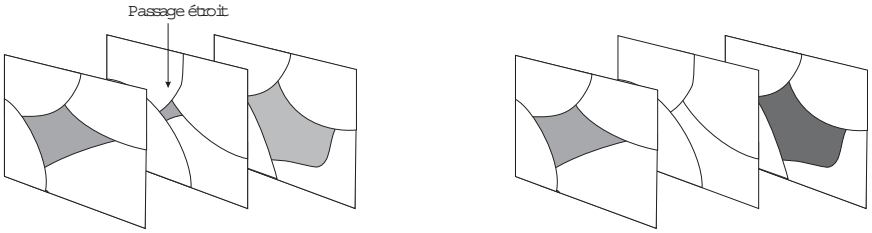
Le fait d'autoriser l'existence de régions spatialement non connexes peut amener à des erreurs de segmentation lorsque ces régions correspondent à des fuites de propagation et non pas au vrai contenu de l'image. De telles fuites de propagation sont le plus souvent causées par un fort mouvement, et peuvent se produire à deux niveaux :

- au niveau de la partition fine ;
- au niveau de la hiérarchie.

7.7.2.1 Erreurs de propagation au niveau de la partition fine

Ces erreurs sont liées à des fuites dans l'étape de propagation des zones plates. Elles se produisent quand, à cause du mouvement, deux objets différents deviennent temporellement voisins et le hasard veut que certains des pixels appartenant aux différents objets aient une couleur similaire. La propagation des zones plates décide alors de fusionner ces pixels comme appartenant au même objet. De telles fuites se caractérisent le plus souvent par des régions très petites et spatialement non connexes. Ces régions ne sont pas éliminées par l'étape de suppression de petites régions car, selon la connexité spatio-temporelle, elles font partie d'une région beaucoup plus grande.

Une solution à ce problème consiste, une fois la partition fine calculée, et avant le passage à la représentation sous forme de graphe, à réaliser un post-traitement de façon à éliminer les régions dont la surface 2D est trop petite. La partition fine est ainsi traitée image par image, en connexité 2D, de façon à ce que toutes les régions dont la surface est plus petite qu'un certain seuil soient fusionnées avec la région voisine dont la couleur est plus proche. Il est naturel de choisir la valeur du seuil de surface en fonction de celui utilisé pour le calcul de la partition fine, comme : seuil_surface / taille_bloc. Cette opération peut cependant déconnecter des régions spatio-temporelles par élimination de petits passages (fig. 7.23). Une dernière étape de réétiquetage des régions spatio-temporelles est alors nécessaire. Les régions qui ont été divisées en plusieurs parties sont considérées comme de nouvelles régions.



(a) Région 3D avant post-traitement. (b) Déconnexion de la région après post-traitement.

FIG. 7.23 – Déconnexion d'une région 3D à cause du post-traitement de la partition.

La figure 7.24 montre les résultats de ce post-traitement sur un bloc de la séquence « Carphone ». Le seuil de taille utilisé est de 5 pixels par image. Remarquons la disparition des très petites régions – surtout dans la zone de la fenêtre, où le mouvement est très fort – pour donner une partition fine plus « propre ».

7.7.2.2 Fuites au niveau de la hiérarchie

Lorsque des forts mouvements se produisent, on remarque certaines « fuites » pour des niveaux relativement grossiers de la hiérarchie. La raison est la suivante : comme l'arbre est calculé selon les chemins minimaux, il suffit que deux objets de couleur similaire se rapprochent pendant quelques instants dans un même bloc pour que les deux objets deviennent voisins sur le graphe et qu'ils soient reliés par une arête de faible valeur. Quand les objets se séparent à nouveau quelques images plus tard, ils restent cependant unis dans la hiérarchie. Ce problème ne peut avoir lieu qu'à l'intérieur d'un bloc car, tel que nous avons conçu notre algorithme, une fois que deux régions ont été hiérarchisées sur l'arbre, c'est-à-dire, ont été reliées par un chemin, il n'est plus possible de le changer.

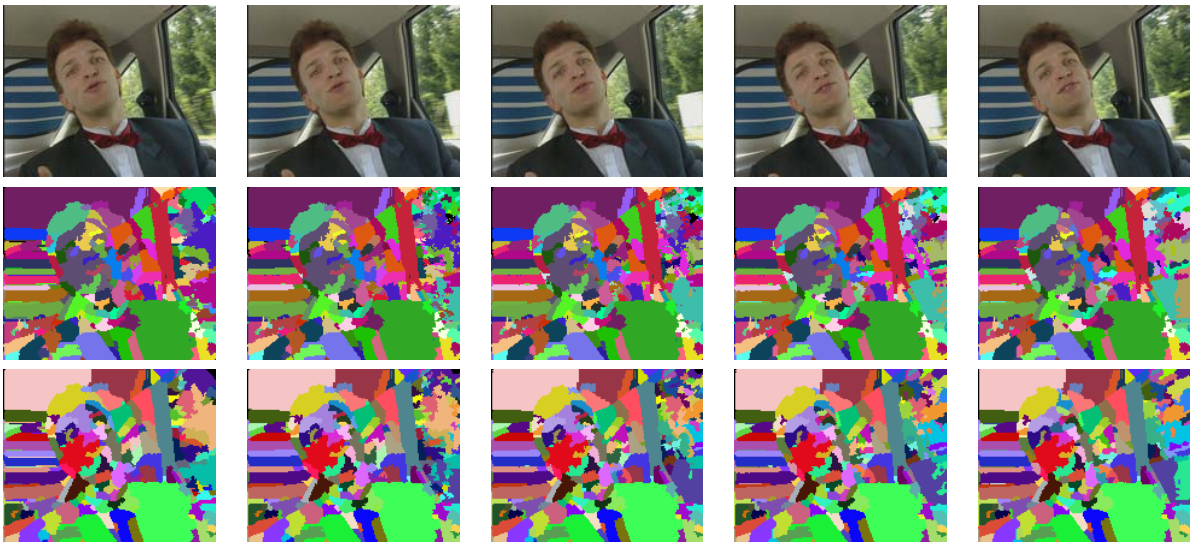


FIG. 7.24 – Partitions fines obtenues avec et sans post-filtrage, pour les images 292-296 de la séquence « Carphone ». De haut en bas : images originales, partition fine avant post-traitement, partition fine après post-traitement.

Ce type de fuites peut être minimisé en modifiant la manière dont le graphe de voisinage G_N est construit. Jusqu'à maintenant, deux régions voisines sur le graphe étaient toujours liées par une arête dont la valeur exprimait la dissimilarité entre les couleurs moyennes des deux régions. Considérons maintenant que les arêtes sont valuées avec la dissimilarité couleur *seulement si les deux régions sont spatialement voisines pour toutes les images du bloc*. Si elles ne le sont pas, alors l'arête est valuée avec une très forte valeur (par exemple la valeur maximale que l'on peut associer à l'arête) correspondant à une forte dissimilarité entre régions. De cette façon une arête reliant deux régions qui ne sont pas voisines pour toutes les images du bloc n'appartiendra à l'arbre que si elle constitue le seul chemin entre les deux régions sur le graphe de voisinage. Les régions qui disparaissent pendant la durée du bloc feront exception à cette règle ; celles-ci obtiennent toujours une valeur d'arête par dissimilarité couleur. Cette méthode a pour but de pénaliser fortement les chemins fusionnant des régions ayant un mouvement différent. En effet, deux régions qui sont voisines à un moment donné, mais qui ne le sont plus après quelques images, correspondent à des régions ayant des mouvements différents. Ces régions ne doivent pas être fusionnées, même si elles ont une couleur similaire. Les fortes valeurs des arêtes empêchent ces fusions.

La figure 7.25 présente des résultats obtenus avec cette méthode de construction du graphe. La partition fine a subi le post-traitement décrit dans la section 7.7.2.1 afin d'éliminer de possibles erreurs de propagation et distinguer plus clairement les fuites de hiérarchisation. La partition en 10 régions obtenue pour le premier bloc de la séquence « Car » est montrée pour la hiérarchie obtenue avec construction « traditionnelle » du graphe de voisinage (au milieu), et avec la construction du graphe que nous venons de proposer (en bas). Dans la hiérarchie « traditionnelle » il y a deux fuites de hiérarchisation importantes : le pare-brise de la voiture a été fusionné avec la barrière. Dans les deux cas, les régions ne sont en contact que pour quelques images du bloc. L'application de la nouvelle méthode permet d'éliminer ces fuites en tenant compte du différent mouvement des objets.

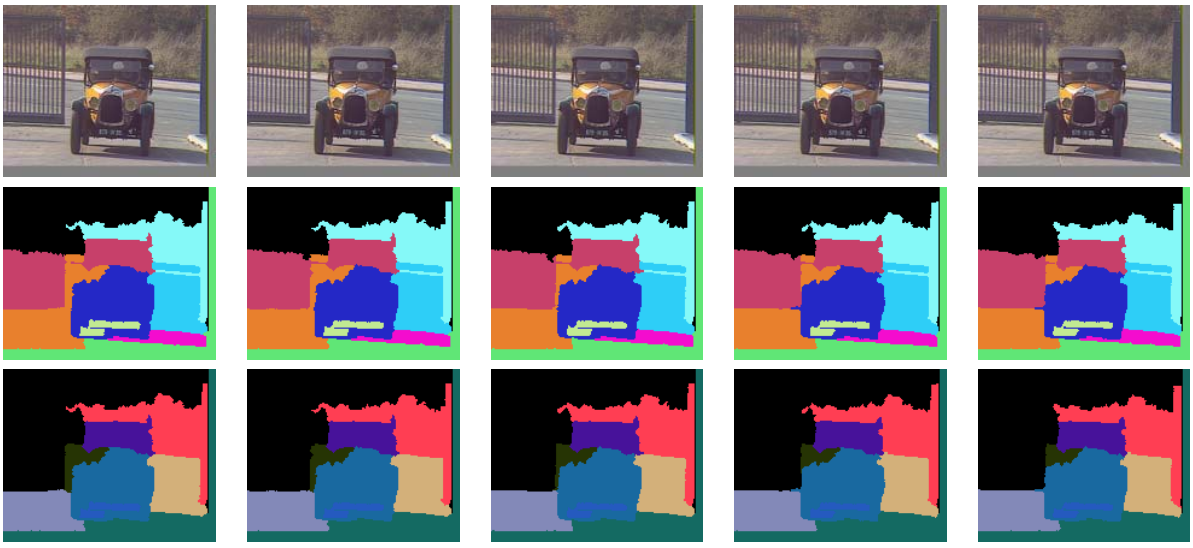


FIG. 7.25 – Partitions en 10 régions pour les images 1 à 5 de la séquence « Car », sans et avec traitement des fuites de hiérarchisation causées par le mouvement. De haut en bas : images originales, partition en 10 régions sans traitement des fuites, partition en 10 régions avec traitement des fuites.

7.7.3 Limitations imposées par le mouvement

Dans toute notre approche de construction d'une hiérarchie spatio-temporelle correspondant à une séquence vidéo nous n'avons pas fait appel à des techniques d'estimation de mouvement. En identifiant les régions qui sont les mêmes sur les différentes images, notre méthode peut être considérée, en elle-même, comme une technique d'analyse du mouvement. Elle a cependant une limitation : les régions qui à cause du fort mouvement sont temporellement déconnectées, ne peuvent pas être détectées. Si l'on disposait d'une méthode d'estimation de mouvement suffisamment robuste, ce problème pourrait être résolu en utilisant les vecteurs de mouvement pour définir les relations de voisinage temporelles. Mais les méthodes existantes ne sont pas suffisamment fiables pour permettre d'appliquer cette méthode sans introduire davantage d'erreurs.

Une autre possible solution consisterait à analyser – image par image – les nouvelles régions au niveau de la partition fine, en essayant de les mettre en correspondance avec des régions qui ont disparu pour l'image précédente et dont la couleur est suffisamment proche. Afin de simplifier la recherche, on pourrait ne considérer qu'un voisinage limité dans le graphe de voisinage, par exemple les deuxièmes où troisièmes voisins de la région que l'on cherche à apparier.

7.8 Interaction avec l'utilisateur

Une fois la hiérarchie associée à la séquence vidéo calculée de manière totalement automatique, on dispose d'un arbre aux arêtes valuées représentant cette information. On se retrouve alors dans la même situation que pour les images fixes (cf. chap. 6). Ainsi, toutes les opérations de manipulation de l'arbre décrites pour les images fixes (baguette magique, resegmentation

et fusion de régions, accès à un niveau de la hiérarchie, lasso) peuvent être appliquées sans restriction au cas des séquences vidéo, car elles consistent en simples éliminations ou ajouts d'arêtes sur l'arbre.

Du point de vue de l'utilisateur, nous proposons de présenter l'interface sous forme de magnétoscope virtuel, tel qu'on le retrouve dans VOGUE (cf. chap. 8) [44]. Ce magnétoscope ne présente qu'une seule image à la fois, ce qui simplifie l'interface et la rend très similaire à celle utilisée pour les images fixes. Des commandes semblables à celles d'un magnétoscope réel sont disponibles pour accéder aux différentes images de la séquence : jouer vers l'avant, jouer vers l'arrière, arrêter, accès aléatoire à une image. L'utilisateur interagit avec n'importe quelle image de la séquence, exactement de la même manière que pour les images fixes. La différence est que, maintenant, l'interaction affecte la séquence complète. Ainsi, quand l'utilisateur joue la séquence, les résultats sont montrés pour toute la séquence sans calcul additionnel, car il s'agit simplement de passer l'information de l'arbre à l'image (conf. chapitre 6, section 6.7) . La figure 7.26 présente un exemple de ce type d'interface. Les résultats sont présentés à droite en forme d'étiquettes, et à gauche en forme de contours superposés à l'image originale.

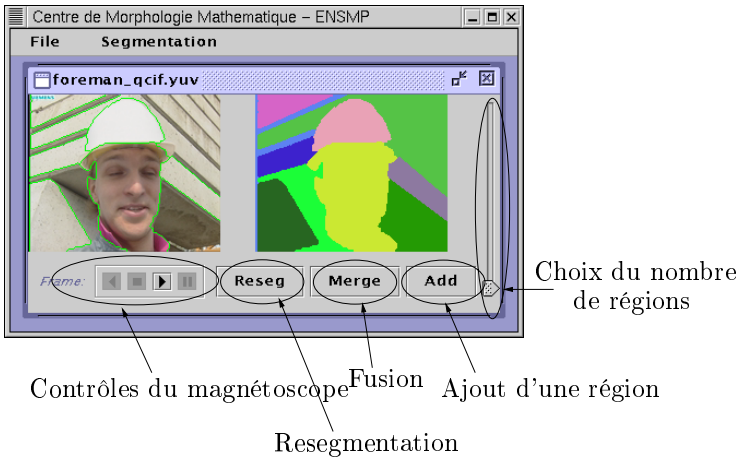


FIG. 7.26 – Interface graphique pour le traitement de séquences sous forme de magnétoscope virtuel.

La figure 7.28 présente une segmentation obtenue après l'intervention de l'utilisateur. La séquence originale est montrée dans la figure 7.27. La hiérarchie a été calculée automatiquement (temps de calcul donnés dans la section 7.6). Ensuite, l'utilisateur a interagi sur *une seule image de la séquence*, comme s'il s'agissait d'une image fixe et selon les méthodes décrites dans les chapitres 4 et 6. Le changement d'état de l'arbre affecte toute la séquence. Le temps de manipulation de l'arbre sur un Pentium III à 800MHz est inférieur à un millième de seconde par opération. Ce temps constitue le temps de réponse de l'algorithme aux actions de l'utilisateur.

7.9 Conclusions

Dans ce chapitre nous avons présenté une technique pour construire une hiérarchie de partitions emboîtées associée à une séquence vidéo. Cette méthode a comme caractéristique principale de réduire de manière considérable la redondance de la représentation choisie. En

FIG. 7.27 – Images 1 à 50 de la séquence « Coastguard ».

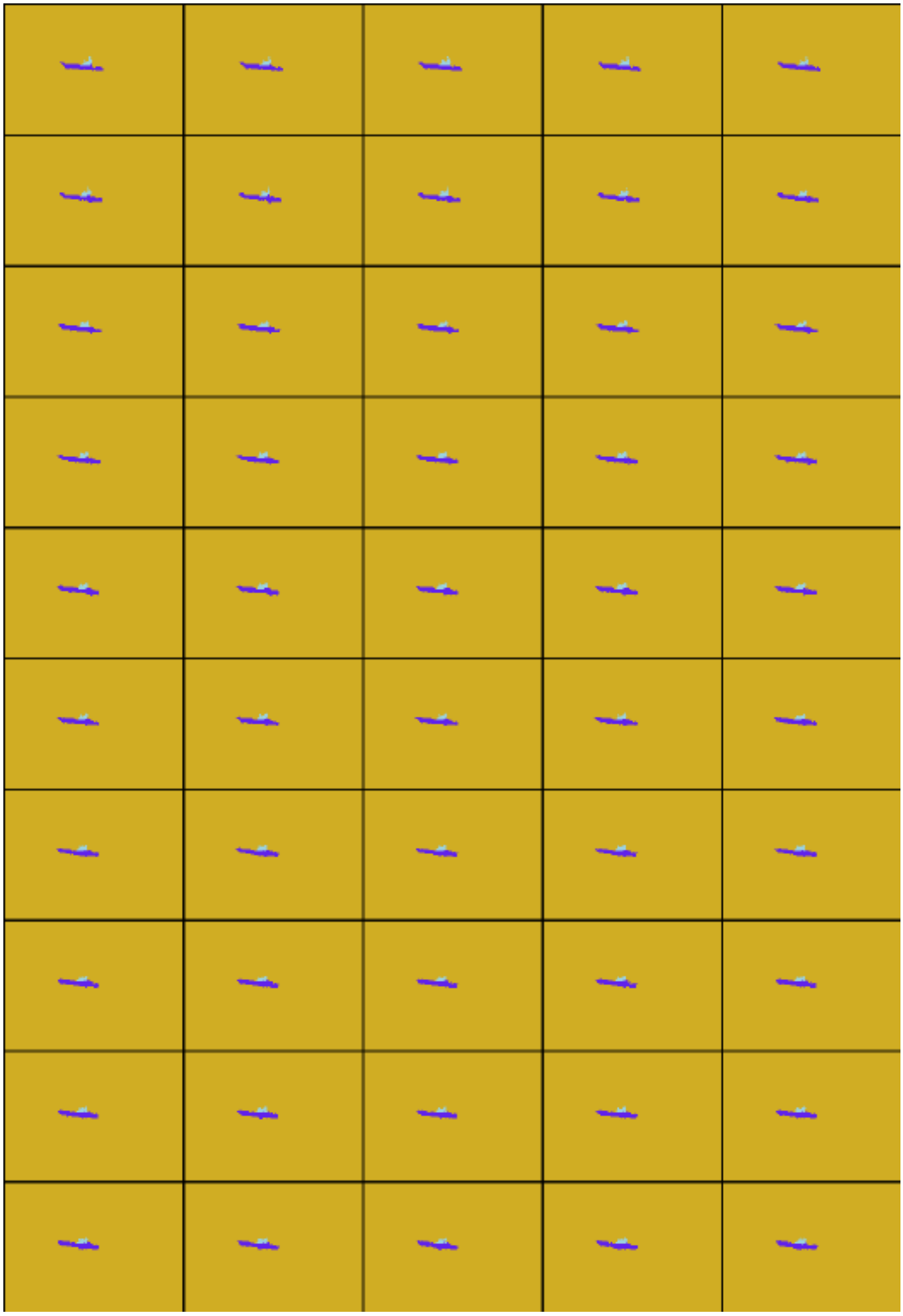


FIG. 7.28 – Objet vidéo extrait à partir de la hiérarchie en appliquant interaction sur une seule image de la séquence, pour la séquence de l'image 7.27.

effet, la hiérarchie associée à une séquence de plusieurs centaines d'images – et plusieurs dizaines de milliers de pixels par image – est représentée par un arbre contenant seulement quelques milliers de nœuds. Les nœuds de l'arbre représentent des régions spatio-temporelles. Les arêtes représentent des relations de voisinage spatio-temporelles, et la valeur des arêtes donne l'importance relative des régions.

La réduction de redondance dans la représentation a de grands avantages du point de vue de l'utilisateur :

- Les calculs plus lourds de construction de la hiérarchie sont réalisés sans supervision. Ensuite, les calculs nécessaires à définir ou modifier les objets sont réalisés sur l'arbre par ajout ou suppression d'arêtes et leur temps d'exécution est imperceptible par l'utilisateur. Sauf pour la segmentation à partir de marqueurs, toute modification dans la définition des objets est réalisée également sur l'arbre et ne requiert pas un recalcul de la hiérarchie.
- Comme les nœuds de l'arbre représentent des régions spatio-temporelles, il suffit à l'utilisateur d'interagir sur une seule image de la séquence pour définir les objets d'intérêt, et cette interaction affecte automatiquement toute la séquence, sans temps de calcul additionnel.

La qualité des hiérarchies obtenues pour les séquences testées est satisfaisante. En particulier, la partition fine présente rarement des fuites, et les régions constituant les différents niveaux de la hiérarchie sont suivies de façon cohérente tout au long de la séquence. Les nouvelles régions apparaissant au fur et à mesure de l'évolution de la séquence sont correctement hiérarchisées. Une fois la hiérarchie calculée, la définition de l'objet est faite par l'utilisateur en quelques secondes sur une unique image de la séquence.

Le problème principal de cette méthode est l'impossibilité de suivre des régions non connexes dans la dimension temporelle. Bien que ce problème est bien réel, il n'apparaît que dans des séquences où il y a de petits objets en mouvement rapide (par exemple la balle dans la séquence « Table tennis »). Dans la plupart des séquences traitées, où les objets sont relativement grands et le mouvement n'est pas trop rapide, ce problème n'apparaît pas, comme en témoigne la faible augmentation du nombre de nouvelles régions. Comme possible ligne de résolution du problème, nous avons suggéré l'analyse des nouvelles régions au niveau de la partition fine, en essayant de les mettre en correspondance avec des régions disparues dans l'image précédente et dont la couleur et/ou la forme indiqueraient qu'il s'agit de la même région.

Chapitre 8

Perspectives industrielles

8.1 Introduction

Les outils développés dans cette thèse ont des applications dans tous les domaines où une certaine quantité d'interaction est tolérée dans le processus de segmentation. De possibles applications se trouvent dans la médecine ou l'infographie. En particulier, les outils utilisés en infographie sont encore assez rudimentaires, et les versions améliorées des outils baguette magique et lasso basées sur une hiérarchie de partitions seraient de grande utilité, sans mentionner la possibilité de segmenter des séquences vidéo au lieu de se restreindre aux images fixes. Ici, nous décrivons les perspectives dans un domaine d'application particulier : la norme MPEG-4.

MPEG-4 est une norme de codage qui, à la différence des normes antérieures telles que MPEG-1 ou MPEG-2, n'est pas uniquement focalisée sur les aspects de compression, mais ajoute toute une série de nouvelles fonctionnalités basées sur la définition d'*objet audiovisuel*. Ainsi, une séquence vidéo n'est plus considérée comme une série d'images consécutives, mais comme un ensemble d'objets avec des relations spatio-temporelles, qui sont codés dans différents bitstreams. Par exemple, une personne qui parle constitue un objet audiovisuel. Puisque nous nous concentrons sur la segmentation d'images et vidéo, nous ne tenons pas compte de la partie audio. Ainsi, nous parlerons d'*objets vidéo*. Basées sur cette nouvelle décomposition de la scène, la norme MPEG-4 définit toute une série de fonctionnalités. En voici quelques-unes [36], [77] :

- Interactivité associée aux objets : des actions différentes peuvent être associées à chacun des objets. L'utilisateur sélectionne l'un des objets, ce qui déclenche une action.
- Composition de scènes : puisque les objets sont codés dans des bitstreams séparés, il est possible de combiner des objets provenant de sources différentes pour composer une nouvelle scène.
- Scalabilité : afin d'optimiser la largeur de bande, MPEG-4 permet d'allouer un plus grand nombre de bits aux objets les plus importants, tandis que le reste de la scène est codé à faible débit. De plus, les objets peuvent être codés à des débits différents selon la largeur de bande disponible à chaque instant, ce qui permet d'adapter le codage d'une même scène à différents canaux de transmission.

Ces nouvelles fonctionnalités nécessitent un découpage de la scène en objets vidéo. Or la norme ne définit pas la manière dont ces objets doivent être générés. La disponibilité de logiciels permettant leur extraction est alors fondamentale pour le succès de MPEG-4.

Nous présentons dans la section 8.2 un logiciel prototype qui intègre une partie des algorithmes présentés dans cette thèse. Ce logiciel a été développé dans le cadre du projet européen MoMuSys et a pour fonction l'extraction d'objets vidéo. Ensuite, dans la section 8.3 nous décrivons comment un module d'extraction d'objets vidéo pourrait s'intégrer dans un logiciel plus complexe dédié à la création de contenu MPEG-4.

8.2 VOGUE

VOGUE (Video Object Generator with User Environment) [44] est un prototype d'extraction d'objets vidéo développé dans le contexte du projet européen MoMuSys (Mobile Multimedia Systems) [18], par le Centre de Morphologie Mathématique de l'École des Mines de Paris (CMM-ENSMP), l'Instituto Superior Tecnico (IST) du Portugal, l'Universitat Politècnica de Catalunya (UPC) en Espagne et l'Universität Hanover (UH) en Allemagne.

MoMuSys est un projet financé par l'Union Européenne qui a démarré en 1995 et qui a eu pour mission principale d'appuyer le développement de la norme MPEG-4 moyennant la mise en œuvre d'un système de communication mobile intégrant les nouvelles fonctionnalités spécifiées dans ce standard. Le succès du projet a amené à une extension en 1998 afin d'adresser des aspects qui ne figuraient pas dans le cahier des charges de départ. La création d'un logiciel d'extraction d'objets vidéo est l'un de ces aspects. En effet, l'utilité d'une grande partie des fonctionnalités MPEG-4 dépend de la disponibilité d'outils permettant de définir facilement des objets vidéo. Même si cette partie n'est pas spécifiée par la norme, elle s'avère indispensable à son succès. VOGUE a été développé dans le cadre de cette extension, au sein du groupe de travail 5.2. VOGUE a été utilisé pour réaliser des tâches de segmentation dans le contexte d'autres projets européens tels que MODEST ou COST 211, et a reçu des critiques très favorables de la part de ses utilisateurs. En particulier, le haut niveau d'automatisation par rapport à des outils commerciaux tels que PhotoPaint a été souligné.

Ce prototype est composé de plusieurs modules intégrés dans une interface graphique qui, combinés, permettent d'extraire les objets vidéo de manière semi-automatique. Trois modules complémentaires ont été intégrés : segmentation interactive d'images fixes, suivi d'objets et détection de mouvement. Nous présentons brièvement chacun de ces modules.

8.2.1 L'interface graphique

L'interface graphique a été développée en Java par l'IST. Elle inclut des fonctionnalités de visualisation et manipulation de séquences, ainsi que l'accès et les outils d'interaction associés aux différents algorithmes.

La figure 8.1 présente une illustration de l'interface. Une barre de menus propose des fonctions d'ouverture/enregistrement de fichiers, d'édition ainsi que d'accès aux algorithmes. Une partie de ces fonctions est également accessible en utilisant la barre d'outils contextuelle. La zone de travail contient les fenêtres associées aux différents algorithmes.

8.2.2 Segmentation interactive d'images fixes

Le module de segmentation d'images fixes est basé sur la hiérarchie de partitions emboîtées présentée dans le chapitre 6. Quand l'utilisateur démarre l'algorithme, la hiérarchie est construite et le niveau de la hiérarchie avec 10 régions est présenté à l'utilisateur (fig. 8.2). L'interface montre à droite la partition sous forme d'étiquettes de couleurs différentes, tandis

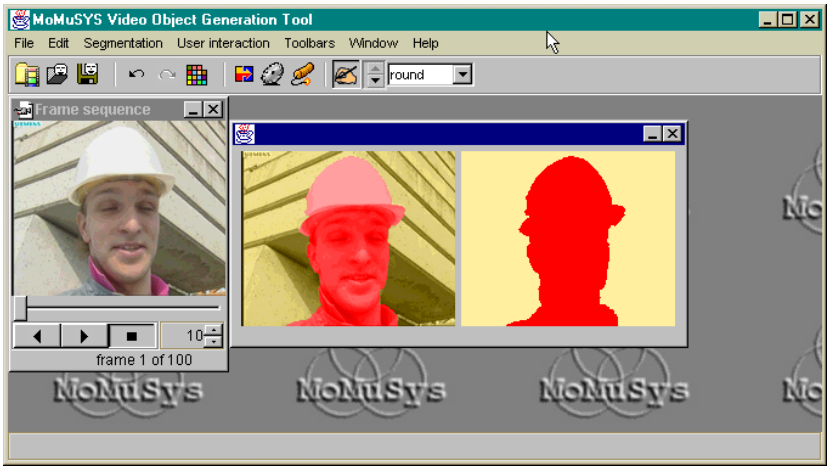


FIG. 8.1 – Interface graphique de VOGUE.

qu'à gauche les étiquettes sont superposées à la luminance pour mieux apprécier la précision des contours. La barre de défilement permet à l'utilisateur de choisir le nombre de régions le mieux adapté au contenu de l'image.

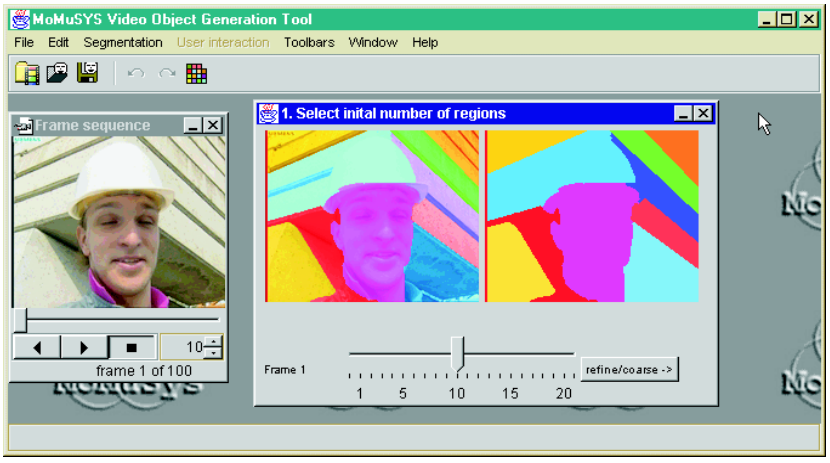


FIG. 8.2 – Étape de sélection du nombre de régions à l'aide d'une barre de défilement.

Ensuite, l'utilisateur passe à l'étape suivante, où un outil de resegmentation et de fusion de régions permet de descendre ou de monter localement dans la hiérarchie. L'utilisateur choisit une région avec la souris, et ensuite presse le bouton de resegmentation pour rediviser la région choisie en des régions plus petites (descente dans la hiérarchie) ou sur le bouton de fusion pour fusionner la région choisie avec ses voisines les plus similaires (monter dans la hiérarchie). Le nombre de régions dans lequel la région est resegmentée ou avec lesquelles elle est fusionnée est déterminé par un paramètre que l'utilisateur peut modifier. La figure 8.3 illustre cette étape. Remarquons que la partition présentée ne correspond pas à un niveau de la hiérarchie, mais à une composition de différents niveaux.

L'étape suivante consiste à modifier les contours des régions obtenues s'ils ne sont pas totalement satisfaisants. Cette étape n'utilise pas directement la hiérarchie, mais la partition

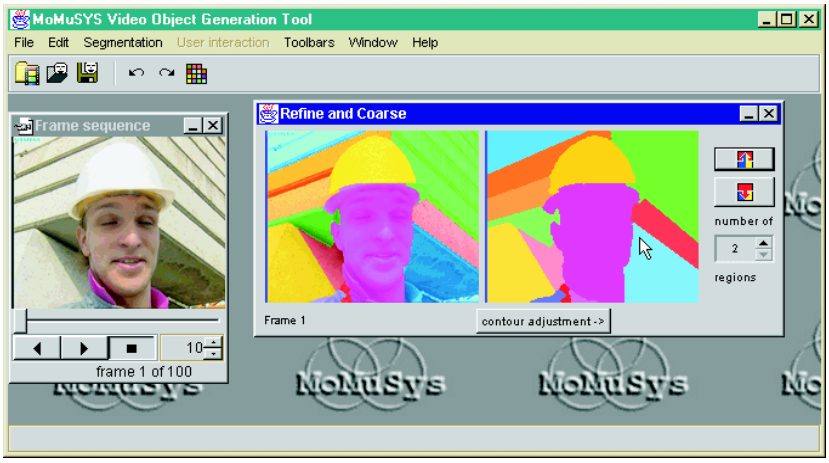


FIG. 8.3 – Étape de resegmentation et fusion de régions.

fine sous-jacente. L'utilisateur clique sur le bord d'une région et étire le contour avec la souris. Les régions de la partition fine sur lesquelles l'utilisateur a déplacé la souris sont ajoutées à l'objet (fig. 8.4). Les contours qui ont été modifiés de cette manière dans la figure sont, par rapport à la partition de la figure 8.3, ceux correspondant au volet du casque, et aux cheveux près de l'oreille.

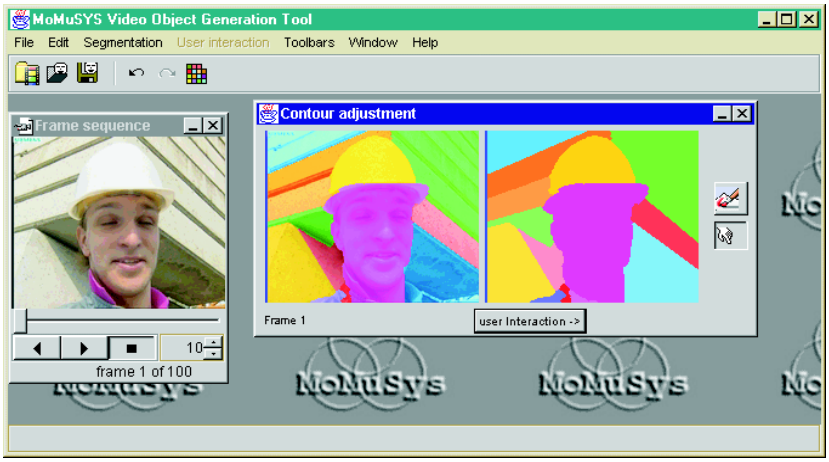


FIG. 8.4 – Ajustement des contours par ajout de régions de la partition fine.

Finalement, la dernière étape permet de réaliser des fusions de régions non prévues par la hiérarchie, afin d'obtenir l'objet d'intérêt. Dans cette étape, et à la différence de l'étape de segmentation avec la hiérarchie, toute fusion est permise. L'utilisateur clique sur les régions qui constituent l'objet d'intérêt. Ensuite, une option « définir objet » est disponible, ce qui provoque la fusion automatique de toutes les régions qui n'ont pas été désignées comme appartenant à l'objet. D'autres outils sont également disponibles à cette étape, qui permettent de réaliser de petites modifications manuelles sur la partition. Par exemple, il est possible de modifier pixel à pixel les contours. Les outils manuels doivent être disponibles afin d'assurer une flexibilité totale dans la définition de l'objet, mais ils ne sont que très rarement nécessaires. La

figure 8.1 présente l'objet final. La définition de l'objet n'a pris au total que quelques dizaines de secondes.

L'étape de sélection d'un niveau avec la barre de défilement peut être remplacée par une étape d'introduction de marqueurs. Cette étape donne une partition de départ. Ensuite, les étapes de resegmentation, fusion et modification de contours continuent à être disponibles. La figure 8.5 illustre l'étape de segmentation par marqueurs.

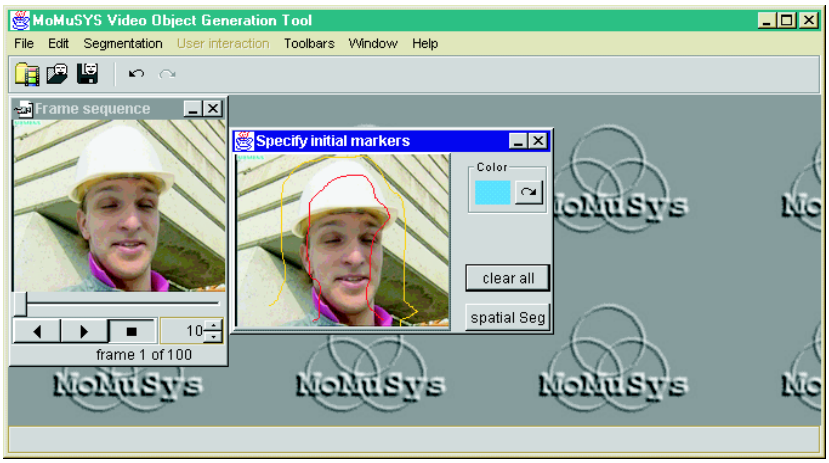


FIG. 8.5 – Segmentation à partir de marqueurs.

8.2.3 Suivi d'objets

Le module de suivi d'objets a été développé par l'Universitat Politècnica de Catalunya [46] et il est destiné au suivi automatique des objets définis à partir du module de segmentation interactive d'images fixes. Cet algorithme est basé sur le suivi d'une partition fine, qui se fait par projection de marqueurs de la partition précédente, suivi d'une ligne de partage des eaux. Les régions de la partition fine sont attribuées à l'objet à partir de l'information donnée par une « look up table ». Quand de nouvelles régions apparaissent, une décision est prise sur leur appartenance à l'objet, à partir de critères de couleur, position et forme.

Les résultats sont présentés au fur et à mesure qu'ils sont calculés (fig. 8.6). Ce module permet également de faire des corrections. Quand des erreurs se produisent, l'utilisateur arrête l'algorithme et une fenêtre de correction d'erreurs est proposée, permettant de modifier la partition en utilisant les mêmes outils que pour la segmentation interactive d'images fixes.

8.2.4 Détection du mouvement

Le module de détection du mouvement a été développé par l'Universität Hanover [48] et permet de détecter les objets bougeant dans une scène. À la différence du suivi, cet algorithme n'a pas besoin d'un masque initial. En contrepartie, les contours des objets détectés sont moins précis que ceux obtenus avec l'algorithme de suivi. Ce type d'algorithme est adapté aux applications dont le but est de détecter les objets en mouvement et où la précision des contours n'est pas le souci primordial, comme c'est le cas de la vidéosurveillance. De la même manière que pour le suivi, il est possible d'arrêter l'algorithme à tout moment et de réaliser des corrections en utilisant les outils interactifs développés pour les images fixes.

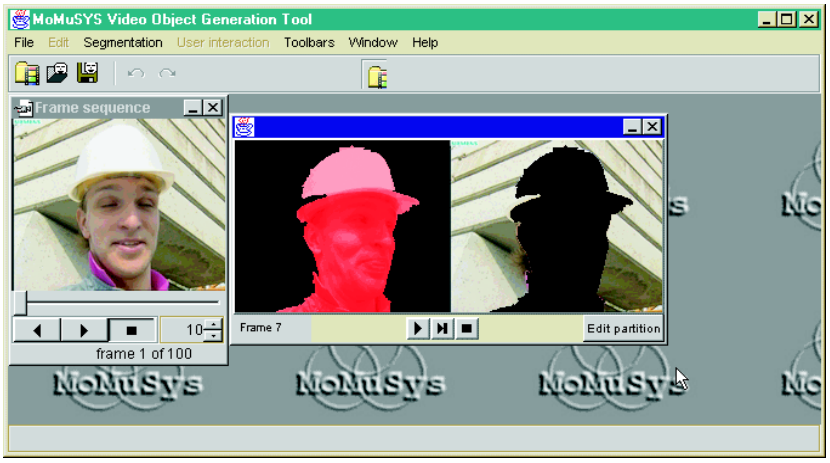


FIG. 8.6 – Fenêtre de suivi des objets.

8.3 Logiciel d'édition de scènes MPEG-4

La disponibilité d'un logiciel d'extraction d'objets vidéo comme VOGUE ne suffit pas pour accéder aux fonctionnalités offertes par MPEG-4. En allant un peu plus loin, et si l'on veut mettre la norme MPEG-4 à la portée du grand public, il est nécessaire d'intégrer l'extraction d'objets dans un système beaucoup plus complexe dédié à la composition et à l'édition de scènes MPEG-4. Un exemple est celui proposé dans [8]. Un tel système intègre un éditeur de scènes avec un codeur-décodeur MPEG-4. Les scènes composées par l'utilisateur sont codées au format MPEG-4 et l'utilisateur peut ensuite, par exemple, les publier sur internet, où elles seront accessibles aux personnes munies d'un décodeur.

Pour composer les scènes, l'utilisateur a la possibilité de combiner différents objets audiovisuels, naturels et synthétiques. La composition de la scène se fait de manière simple à partir d'une interface graphique permettant d'insérer et de déplacer les objets avec la souris. Ensuite, les caractéristiques des objets sont particularisées à l'aide de menus déroulants. Pour faciliter d'avantage la tâche, des « templates » sont disponibles pour une variété de scènes. C'est-à-dire, le logiciel dispose de scènes pré-stockées qui peuvent être adaptées par l'utilisateur.

Chaque type d'objet audiovisuel dispose de son propre éditeur qui permet de le manipuler. Ainsi, un fichier de son sera manipulé avec un éditeur d'audio, et une séquence vidéo sera manipulée en utilisant l'éditeur de films. C'est ici qu'un module d'extraction d'objets vidéo trouve sa place. En effet, l'utilisateur peut disposer d'un film pré-enregistré dont il veut extraire un ou plusieurs objets pour les inclure dans la scène. En choisissant la fonction d'extraction d'objets, une nouvelle fenêtre s'ouvre. Les algorithmes et outils de segmentation interactive d'images fixes et séquences vidéo présentés dans cette thèse sont alors disponibles, et l'utilisateur peut extraire les objets vidéo d'intérêt avec quelques clicks de souris.

Chapitre 9

Conclusions et perspectives

9.1 Apports de cette thèse

Dans cette thèse nous avons développé un système permettant de segmenter interactivement aussi bien des images fixes que des séquences vidéo. Ce système présente les caractéristiques suivantes :

- C'est un système généraliste, qui ne fait pas de supposition sur le contenu des images. Néanmoins, nous avons également proposé des mécanismes permettant d'exploiter les informations *a priori* dans les cas où de telles informations sont disponibles.
- L'utilisation d'une hiérarchie de partitions plutôt qu'une partition unique donne une très grande flexibilité au système. De multiples définitions d'objet sont possibles pour une même image ou séquence d'images sans avoir à refaire des calculs.
- La qualité des hiérarchies utilisées permet de minimiser la quantité d'interaction nécessaire pour extraire les objets.
- Grâce à l'importante réduction de redondance temporelle de la représentation sous forme d'arbre choisie, la segmentation interactive de séquences d'images ne requiert pas beaucoup plus d'effort de la part de l'utilisateur que la segmentation d'images fixes. En effet, le plus souvent il suffit d'interagir sur une seule image de la séquence pour obtenir la segmentation de la séquence complète. De plus, cette réduction de redondance et la simplicité de la représentation en forme d'arbre permettent de répondre aux actions de l'utilisateur en un temps très court. Ainsi, l'utilisateur perçoit la réponse à l'interaction comme étant immédiate.
- Les outils d'interaction proposés sont très intuitifs et permettent à l'utilisateur de définir les objets d'intérêt sans nécessité d'aucune connaissance sur les algorithmes sous-jacents. Cette caractéristique rend notre système très approprié pour les applications destinées au grand public.

Le système proposé est basé sur le calcul d'une famille de partitions emboîtées correspondant à une image fixe ou une séquence d'images. La première étape consiste à filtrer l'image de manière à réduire le niveau de bruit ainsi que des textures qui pourraient nuire aux étapes suivantes. Cette étape facilite également le calcul de la partition fine. Nous avons à ce sujet proposé une extension des nivellements aux images couleur qui améliore les versions précédentes en donnant une définition formelle et en même temps un algorithme de calcul simple.

Nous avons ensuite proposé une méthode pour le calcul d'une hiérarchie de partitions emboîtées à partir de concepts morphologiques. En combinant l'arbre de poids minimum avec les

ultramétriques obtenues par inondation synchrone nous obtenons des hiérarchies qui, contrairement à d'autres approches basées sur l'arbre de poids minimum, donnent des partitions très significatives même pour les niveaux de la hiérarchie contenant très peu de régions. D'autre part, moyennant l'utilisation des zones plates couleur, nous obtenons une partition fine adaptée au contenu de l'image et en même temps très peu sursegmentée, qui peut être représentée par un arbre avec un faible nombre de nœuds.

Nous avons également présenté des mécanismes permettant d'améliorer les résultats quand on dispose de connaissances sur les caractéristiques des images. Les algèbres d'ultramétriques et l'utilisation de distances lexicographiques permettent de combiner l'information de différentes hiérarchies en une seule afin d'obtenir des résultats mieux adaptés à certains types d'images. D'ailleurs, les marqueurs flous permettent de donner plus d'importance à certaines régions en leur facilitant l'accès aux niveaux élevés de la hiérarchie.

Les concepts présentés pour les images fixes ont été ensuite étendus aux séquences vidéo. À la différence des systèmes antérieurs, où une unique partition est suivie au long de la séquence, nous avons développé une technique permettant de calculer une hiérarchie de partitions emboîtées associée à la séquence complète. Les nouvelles régions apparaissant plus tard dans la séquence sont naturellement incluses dans la hiérarchie. La hiérarchie est représentée par un arbre aux arêtes valuées dont les nœuds représentent des régions spatio-temporelles, ce qui amène à une réduction très importante de la redondance temporelle et résulte en un arbre simple qui peut être manipulé avec peu de calculs. L'utilisation d'une hiérarchie spatio-temporelle permet également de réduire énormément la quantité d'interaction nécessaire pour définir les objets d'intérêt. Une modification de la définition de l'objet se traduit par une altération de l'état de l'arbre et affecte automatiquement toute la séquence sans besoin de refaire des calculs.

Dans le calcul de la hiérarchie associée à une séquence, nous n'avons utilisé aucune analyse de mouvement. De fait, le calcul même de la hiérarchie peut être considéré en soi comme une estimation de mouvement, car on connaît la position et les contours de chaque région pour toutes les images de la séquence. Un inconvénient de cette méthode est cependant le fait que nous ne sommes pas capables de suivre des objets dans le cas où l'échantillonage temporel de la séquence les aurait déconnectés.

Toujours en rapport avec le mouvement, nous avons proposé une technique pour éviter que des régions ayant un mouvement différent ne soient pas fusionnées trop tôt dans la hiérarchie même si leurs couleurs se ressemblent, ce qui se fait en analysant l'évolution des relations de voisinage des régions de chaque image à l'intérieur du bloc.

La disponibilité d'outils d'interaction performants étant un aspect fondamental d'un système interactif, nous avons proposé une batterie d'outils permettant d'interagir avec la hiérarchie de manière simple et intuitive, sans nécessité de connaître le fonctionnement des algorithmes sous-jacents. Quelques uns de ces outils, comme la baguette magique, le pinceau intelligent ou le lasso, sont des versions très améliorées des outils d'interaction que l'on trouve dans des logiciels commerciaux. D'autres sont des nouveaux outils qui découlent naturellement de la notion de hiérarchie : sélection du niveau de résolution, resegmentation ou fusions locales. Finalement, nous avons développé une version adaptée à la hiérarchie d'un outil bien connu de la morphologie mathématique : la segmentation à partir de marqueurs.

Une partie de ces outils a été intégré dans des interfaces graphiques, ce qui a permis de démontrer leur utilité pratique pour la segmentation d'images fixes et de séquences vidéo.

9.2 Perspectives

Nous avons proposé dans le chapitre 6 de nouvelles manières de combiner des hiérarchies, comme les distances lexicographiques et les algèbres d'ultramétriques. Il serait intéressant d'explorer plus en profondeur les possibilités d'application de ces techniques. Les possibilités offertes par les marqueurs flous sont également très nombreuses et méritent d'être étudiées.

En rapport avec le calcul de la hiérarchie associée à une séquence vidéo, remarquons que, tel que nous avons conçu le système, c'est le premier bloc où une région apparaît qui détermine sa position dans la hiérarchie. Cette stratégie donne des résultats satisfaisants en pratique. Néanmoins, au fur et à mesure que des blocs sont traités on dispose de plus en plus d'informations sur les caractéristiques des régions. Les résultats pourraient être améliorés en utilisant ces informations pour réactualiser la hiérarchie moyennant une réévaluation des arêtes. Cette actualisation aurait cependant un coût additionnel en termes de temps de calcul.

Un problème associé à notre système est l'impossibilité de suivre des régions qui, à cause du sous-échantillonage temporel, sont temporellement déconnectées. La résolution de ce problème permettrait d'utiliser notre système dans des problèmes d'analyse du mouvement. Une possible stratégie serait de rechercher, après calcul de la partition fine et du graphe de voisinage, des régions qui disparaissent pour une image donnée mais qui ont dans le graphe de voisinage un deuxième ou troisième voisin de caractéristiques de couleur et de forme similaires.

Finalement, nous avons présenté tout un ensemble d'outils d'interaction principalement basés sur des régions. Ceci est naturel puisque nous travaillons avec des partitions emboîtées. Cependant, dans certains cas il peut être intéressant pour l'utilisateur de pouvoir définir l'objet en dessinant un contour de manière semi-automatique, tel que proposé dans [63] et [16]. Ce type d'interaction demande à l'utilisateur d'introduire un premier point appartenant au contour. Ensuite, un contour se dessine automatiquement entre le point initial et la position de la souris. Quand l'utilisateur bouge la souris, le contour change dynamiquement pour s'adapter toujours au meilleur contour entre le dernier point sélectionné par l'utilisateur et la position de la souris. Une stratégie similaire pourrait être appliquée en utilisant la hiérarchie de partitions. Les contours seraient alors choisis entre ceux appartenant aux régions les plus importantes. L'utilisation du graphe dual semble être une piste intéressante. Sur le graphe dual, les nœuds correspondent à des points triples de la partition (extrémités de chaque morceau de contour), et les arêtes aux contours. L'arbre de poids minimum du graphe de voisinage correspond à l'arbre de poids maximum sur le graphe dual. L'élimination d'une arête sur l'arbre de poids minimum sépare l'arbre en deux composantes connexes et équivaut à la création d'un cycle sur l'arbre de poids maximum du graphe dual (fermeture d'un contour). Ces caractéristiques peuvent être intéressantes pour mettre au point des algorithmes permettant de passer de l'interaction par régions à celle par contours pour une même hiérarchie.

Annexe A

La ligne de partage des eaux

La ligne de partage des eaux (LPE) est une transformation très utilisée en segmentation morphologique. Proposée initialement par Lantuejoul [40], Beucher et Meyer l'ont appliquée à la segmentation d'images [50], [5], [59], [7].

A.1 Définition

La LPE interprète une image à niveaux de gris comme un relief topographique (figure A.1), où la valeur de gris du pixel correspond à la hauteur du point sur le relief. Un tel relief est formé de pics et de vallées. Le point le plus bas de chaque vallée constitue un minimum local du relief.

Imaginons qu'un trou est percé sur chaque minimum du relief, et que celui-ci est plongé dans un grand lac. Au fur et à mesure que l'eau inonde le relief, les bassin versants se remplissent, le niveau de l'eau étant toujours le même pour tous les bassins versants. Quand deux lacs appartenant à des bassins versants différents se retrouvent, leur fusion est empêchée par la construction d'une digue d'épaisseur nulle. Une fois que l'eau a inondé tout le relief, l'ensemble des digues forme des contours fermés passant par les lignes de crête et délimitant les bassins versants du relief topographique. C'est ce qu'on appelle la ligne de partage des eaux.

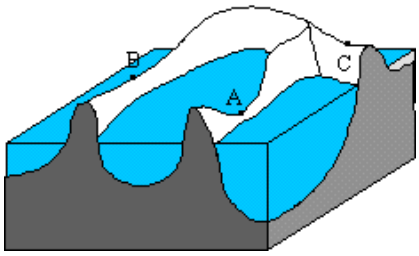


FIG. A.1 – Inondation du relief topographique.

A.2 Application à la segmentation

La LPE permet de détecter les bassins versants de l'image. Or l'application à la segmentation n'est pas directe, car il y a deux obstacles à surmonter :

1. Les contours des objets dans une image ne correspondent pas aux lignes de crête. La solution consiste à utiliser non pas l'image originale, mais son gradient. En effet, le gradient contient des valeurs élevées là où l'image originale contient des fortes transitions, et donc des contours.
2. Le gradient d'une image naturelle contient en général beaucoup trop de minima. Ainsi, puisque chaque minimum local donne lieu à une région de la segmentation finale, le résultat de la LPE correspond à une sur-segmentation de l'image. La solution à ce problème est donnée par la LPE à partir de marqueurs. Ainsi, au lieu de percer des trous sur tous les minima, seulement quelques uns d'entre eux sont sélectionnés. On appelle ces heureux élus des *marqueurs*. De cette manière, les digues n'apparaissent que sur les crêtes les plus élevées séparant deux marqueurs. La segmentation résultante contient une région par marqueur. On résout ainsi le problème de la sur-segmentation, mais un nouveau problème apparaît : celui du choix les marqueurs, dont la solution est spécifique à chaque application.

A.2.1 Algorithme

Il existe des algorithmes permettant de calculer la LPE de façon très efficace. Nous décrivons ici celui basé sur des files d'attente hiérarchiques, qui consiste à étiqueter les pixels dans un ordre particulier.

Une file d'attente hiérarchique est un ensemble de queues FIFO (First In First Out, le premier entré est le premier sorti) organisées par priorités croissantes. Les pixels ne sont extraits d'une queue que lorsque toutes les queues plus prioritaires sont vides. La queue ayant priorité 1 sera considérée en premier, ensuite celle ayant priorité 2, etc. L'idée est de traiter les pixels dans l'ordre établi par l'inondation – et qui correspond à leur niveau de gris – de façon à ce que chaque pixel ne soit traité qu'une seule fois.

Faisons d'abord une remarque. Nous avons défini la LPE comme ayant des lignes de séparation entre régions d'épaisseur nulle, ce qui convient à nos applications (traditionnellement la frontière entre deux régions a une épaisseur d'un pixel). La mise en œuvre sur la grille numérique ne permet cependant pas l'introduction d'une frontière sans épaisseur. Ainsi, afin de distinguer les frontières entre deux régions, une étiquette est attribuée aux pixels appartenant à chacune des régions. De cette façon chaque région de la segmentation se distingue par son étiquette et il n'est plus nécessaire d'introduire des frontières pour les séparer.

L'algorithme A.1 décrit le calcul de la LPE. L'algorithme a deux images en entrée : une image gradient I_G qui contient la fonction à inonder, et une image de marqueurs I_M qui contient les marqueurs étiquetés. La valeur de gris du pixel p est donnée par $I_G(p)$, et son étiquette par $I_M(p)$. Dans I_M, les pixels appartenant à un marqueur sont étiquetés avec une valeur entière positive identifiant le marqueur, les pixels restants étant à zéro. La sortie de l'algorithme est dans l'image I_M. Nous appellerons FA la file d'attente hiérarchique, sur laquelle plusieurs opérations sont possibles :

- $Ajouter(FA, p)$: introduit le pixel p dans la file d'attente de même priorité que le niveau de gris du pixel ;
- $Extraire(FA)$: retourne le premier pixel entré dans la file d'attente la plus prioritaire et l'élimine de la file ;
- $Vide(FA)$: retourne $VRAI$ si la file d'attente FA est vide, et $FAUX$ autrement.

Cet algorithme utilise les étiquettes suivantes pour décrire l'état des pixels : une valeur positive identifie un pixel déjà étiqueté, une valeur à zéro identifie un pixel qui n'a pas encore été traité

et une valeur négative identifie un pixel qui est dans la file d'attente, mais n'en est pas encore sorti. Nous noterons par $N_G(p)$ l'ensemble de voisins du pixel p sur la grille numérique.

Pendant l'étape d'initialisation, l'image de marqueurs est parcourue. Tous les pixels voisins d'un marqueur qui n'ont pas encore été traités (ceux dont l'étiquette est zéro) sont introduits dans la file d'attente, avec priorité égale à leur niveau de gris. En même temps, ils obtiennent l'étiquette de leur voisin marqueur, mais avec signe négatif, ce qui signifie que le pixel est dans la file d'attente, mais qui n'en est pas encore sorti. La deuxième étape extrait les pixels de la file un par un, par priorités croissantes. Le signe de l'étiquette du pixel sortant de la file est alors inversée pour indiquer que le pixel a été traité, et ses voisins non traités sont introduits dans la file, de la même manière que pour l'étape d'initialisation. L'algorithme s'arrête quand la file d'attente est vide, ce qui arrive quand tous les pixels ont été étiquetés.

Il y a dans cet algorithme un point qui mérite d'être discuté, et qui concerne ce qui arrive quand une des files de priorité inférieure se vide. Si l'on considère l'inondation à partir de tous les minima de l'image, le fait que l'une des files se soit vidée implique que tous les pixels correspondant à ce niveau de gris ont été étiquetés. Ceci est équivalent à dire que le niveau de l'eau a dépassé un certain seuil, correspondant à la priorité de la file vide. De ce point de vue, cette file peut être supprimée, puisque plus aucun pixel avec ce niveau de gris apparaîtra au cours de l'inondation. Cependant, quand il s'agit de la LPE à partir de marqueurs, cette supposition n'est plus vérifiée. En effet, il est possible qu'après suppression d'une file d'une certaine priorité il apparaisse des pixels de priorité inférieur, correspondant à des vallées ne contenant pas de marqueur. Ces pixels seront alors introduits dans la file active de plus petite priorité, ce qui équivaut à boucher les minima ne contenant pas de marqueur.

Algorithme A.1 Algorithme de calcul de la LPE par files d'attente hiérarchiques

```
Image I_M, I_G
Pixel p, v
Etiquette e
for all p ∈ I_M do
  e ← I_M(p)
  if e > 0 then {Nous avons trouvé un marqueur}
    for all v ∈ N_G(p) do {Introduire les voisins de p dans la queue}
      if I_M(v) = 0 then
        I_M(v) ← −e
        Ajouter(FA,v)
      end if
    end for
  end if
end for
while ¬Vide(FA) do
  p ← Extraire(FA)
  e ← −I_M(p) {Le pixel est sorti de la queue : inverser l'étiquette}
  I_M(p) ← e
  for all v ∈ N_G(p) do {Introduire les voisins de p dans la queue}
    if I_M(v) == 0 then
      I_M(v) ← −e
      Ajouter(FA,v)
    end if
  end for
end while
```

Bibliographie

[1] Astola J., Haavisto P. et Neuvo Y. – Vector median filters. *Dans : Proceedings of the IEEE.* – avril 1990.

[2] Beaulieu J.-M. et Goldberg M. – Hierarchy in picture segmentation : a stepwise optimization approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, n2, février 1989.

[3] Benzécri J. P. – *L'analyse des données. Tome I : La taxinomie.* – Dunod, 1973.

[4] Berge C. et Chouila-Houri A. – *Programmes, jeux et réseaux de transport.* – Dunod, 1962.

[5] Beucher S. – *Segmentation d'images et morphologie mathématique.* – Thèse de doctorat, École Supérieure des Mines de Paris, juin 1990.

[6] Beucher S. – Watershed, hierarchical segmentation and waterfall algorithm. *Dans : Mathematical Morphology and its Applications to Image Processing, ISMM'94*, éd. par Serra J. et Soille P. pp. 69–76. – Kluwer Academic Publishers, 1994.

[7] Beucher S. et Meyer F. – The morphological approach to segmentation : the watershed transformation. *Dans : Mathematical morphology in image processing*, éd. par Dougherty E., chap. 12, pp. 433–481. – Marcel Dekker, 1993.

[8] Boughoufalah S., Dufourd J.-C. et Bouilhaguet F. – MPEG-Pro, and authoring system for MPEG-4 with temporal constraints and template guided editing. *Dans : Proceedings of ICME 2000, International Conference on Multimedia and Expo.* – New York, États Unis, 30 juillet - 2 août 2000.

[9] Canny J. – A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, n6, novembre 1986, pp. 679–698.

[10] Castagno R., Ebrahimi T. et Kunt M. – Video segmentation based on multiple features for interactive multimedia applications. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, n5, septembre 1998, pp. 562–571.

[11] Chalom E. et Bove V. M. – Segmentation of an image sequence using multi-dimensional image attributes. *Dans : Proceedings of the IEEE International Conference on Image Processing, ICIP*, pp. 525–528. – Lausanne, Suisse, septembre 1996.

[12] Cichosz J. et Meyer F. – Morphological multiscale image segmentation. *Dans : Workshop on Image Analysis for Multimedia Interactive Services*, pp. 161–166. – juin 1997.

[13] Crespo J. et Schafer R. W. – The flat zone approach and color images. *Dans : Mathematical Morphology and its Applications to Image Processing*, éd. par Serra J. et Soille P. pp. 85–92. – Kluwer Academic Publishers, 1994.

[14] Crespo J., Schafer R. W., Serra J., Gratin C. et Meyer F. – The flat zone approach : a general low-level region merging segmentation method. *Signal Processing*, vol. 62, n1, octobre 1997, pp. 37–60.

[15] Delgado Olabarriaga S., Koelma D. et Smeulders A. W. – A simple application framework for interactive segmentation systems. *Dans : Annual Conference of the Advanced School of Computing and Imaging.* – 1997.

[16] Falcão A. X., Udupa J. K., Samarasekera S., Hirsch B. E. et Lotufo R. – User-steered image segmentation paradigms : Live wire and live lane. *Graphical Models and Image Processing*, vol. 60, 1998, pp. 233–260.

[17] Fu K. et Mui J. – A survey on image segmentation. *Pattern Recognition*, vol. 13, n1, 1981, pp. 3–16.

[18] Fuchs H. – *MoMuSys Final Report.* – Rapport technique, AC-098 project, MoMuSys, 2001.

[19] Gatica-Pérez D., Sun M.-T. et Gu C. – Semantic video object extraction based on backward tracking of multivalued watershed. *Dans : Proceedings of the IEEE International Conference on Image Processing, ICIP'99.* – Kobe, Japon, 24-28 octobre 1999.

[20] Gomila C. – Levelings in vector spaces. *Dans : Proceedings of the IEEE Conference on Image Processing, ICIP'99.* – Kobe, Japon, 24-28 octobre 1999.

[21] Gondran M. et Minoux M. – *Graphes et algorithmes.* – Editions Eyrolles, 1985, *Collection de la Direction des Études et Recherches d'Électricité de France.*

[22] Gower J. et Ross G. – Minimum spanning trees and single linkage cluster analysis. *Apl. Statistics*, vol. 18, 1969, pp. 54–65.

[23] Grimaud M. – *La géodésie numérique en Morphologie Matématique. Application a la détection automatique de microcalcifications en mammographie numérique.* – Thèse de doctorat, Ecole des Mines de Paris, 1991.

[24] Grimaud M. – A new measure of contrast : the dynamics. *Dans : Proceedings SPIE, Image Algebra and Morphological Image Processing III*, pp. 292–305. – San Diego, États Unis, 1992.

[25] Gu C. et Lee M.-C. – Semiautomatic segmentation and tracking of semantic video objects. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, n5, septembre 1998.

[26] Guo J., Kim J. et Kuo C.-C. J. – An interactive object segmentation system for MPEG video. *Dans : Proceedings of the International Conference on Image Processing, ICIP'99.* – Kobe, Japon, 24-28 octobre 1999.

[27] Haralick R. M. et Shapiro L. G. – Image segmentation techniques. *Computer Vision, Graphics and Image Processing*, vol. 29, n1, janvier 1985, pp. 100–132.

[28] Hartigan J. – Representation of similarity matrices by trees. *Journal of American Statistics Association*, vol. 62, 1967, pp. 1140–1158.

[29] Hartigan J. et Wong M. – A K-means clustering algorithm. *Applied Statistics*, vol. 28, 1979, pp. 100–108.

[30] Horowitz S. et Pavlidis T. – Picture segmentation by a tree traversal algorithm. *J. Assoc Comput. Mach.*, vol. 23, 1976, pp. 368–388.

[31] Hu T. C. – The maximum capacity route problem. *Operations Research*, vol. 9, 1961, pp. 898–900.

[32] Jardine C., Jardine N. et Sibson R. – The structure and construction of taxonomic hierarchies. *Mathematical Biosciences*, 1967, pp. 173–179.

[33] Johnson S. C. – Hierarchical clustering schemes. *Psychometrika*, vol. 32, n3, septembre 1967, pp. 241–254.

[34] Kass M., Witkin A. et D.Terzopoulos. – Snakes : active contour models. *International Journal on Computer Vision*, vol. 1, n4, janvier 1988, pp. 321–331.

[35] Klein J.-C. – *Conception et réalisation d'une unité logique pour l'analysise quantitative d'images.* – Thèse de doctorat, Université de Nancy, 1976.

[36] Koenen R., Pereira F. et Chiariglione L. – MPEG-4 : Context and objectives. *Signal Processing : Image Communication*, vol. 9, n4, 1997, pp. 295–303.

[37] Kompatsiaris I. et Strinzis M. G. – Spatiotemporal segmentation and tracking of objects in image sequences. *Dans : Proceedings of the IEEE International Conference on Image Processing, ICIP'99.* – Kobe, Japon, 24-28 octobre 1999.

[38] Kruse S., Graffunder A. et Askar S. – A new tracking scheme for semi-automatic video object segmentation. *Dans : Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'99.* – Berlin, Alemagne, mai-juin 1999.

[39] Kruskal J. – On the shortest spanning subtree of a graph. *Proc. Amer. Math. Soc.*, vol. 7, 1956, p. 48.

[40] Lantuéjoul C. – *La squelettisation et son application aux mesures topologiques de mosaïques polycristallines.* – Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1978.

[41] Lemaréchal C. et Fjørtoft R. – Comments on geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, n7, juillet 1998, pp. 762–763.

[42] Leung T. et Malik J. – Contour continuity in region based image segmentation. *Dans : Proceedings of the Fifth European Conference on Computer Vision.* – Freiburg, Alemagne, 2-6 juin 1998.

[43] Marcotegui B. – *Segmentation de séquences d'images en vue du codage.* – Thèse de doctorat, École Nationale Supérieure des Mines de Paris, 1996.

[44] Marcotegui B., Correia P., Marqués F., Mech R., Rosa R., Wollborn M. et Zanoguera F. – A video object generation tool allowing friendly user interaction. *Dans : Proceedings of the IEEE International Conference on Image Processing.* – Kobe, Japon, octobre 1999.

[45] Marcotegui B. et Meyer F. – Bottom-up segmentation of image sequences for coding. *Annales des Telecommunications*, vol. 52, n7-8, 1996, pp. 397–407.

[46] Marqués F. et Llach J. – Tracking of generic objects for video object generation. *Dans : Proceedings of the IEEE International Conference on Image Processing, ICIP'98.* – Chicago, États Unis, 1998.

[47] Matheron G. – *Les nivellements.* – Rapport technique, École Nationale Supérieure des Mines de Paris, 1997.

[48] Mech R. et Wollborn M. – A noise robust method for 2-D shape estimation of moving objects in video sequences considering a moving camera. *Signal Processing : Special Issue on Video Sequence Segmentation for Content-Based Processing and Manipulation*, vol. 66, n2, avril 1998, pp. 203–217.

[49] Meier T. et Ngan K. – Automatic segmentation of moving objects for video object plane generation. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, n 5, septembre 1998, pp. 525–538.

[50] Meyer F. – Algorithmes séquentiels. *Dans : 11ème Colloque GRETSI.* – Nice, France, 1987.

[51] Meyer F. – *Algorithmes à base de file d'attente hiérarchiques.* – Rapport technique, École Nationale Supérieure des Mines de Paris, 1990 1990.

[52] Meyer F. – Color image segmentation. *Dans : 4th Conf. Image Processing and Applications*, pp. 53–56. – Maastrich, Pays Bas, 1992.

[53] Meyer F. – Minimum spanning forests for morphological segmentation. *Dans : Mathematical Morphology and its Applications to Image Processing, ISMM'94*, éd. par J.Serra et P.Soille, pp. 77–84. – Kluwer Academic Publishers, 1994.

[54] Meyer F. – From connected operators to levelings. *Dans : Mathematical Morphology and its Applications to Image and Signal Processing, ISMM'98*, pp. 191–198. – juin 1998.

[55] Meyer F. – The levelings. *Dans : Mathematical Morphology and its Applications to Image and Signal Processing, Proc. ISMM'98*, pp. 199–206. – juin 1998.

[56] Meyer F. – Morphological multiscale and interactive segmentation. *Dans : IEEE-EURASIP Workshop on Non-Linear Signal and Image Processing (NSIP'99).* – Antalya, Turkie, juin 1999.

[57] Meyer F. – Flooding and segmentation. *Dans : Mathematical Morphology and its Applications to Image Processing, ISMM'2000.* – Palo Alto, États Unis, 2000. Submitted.

[58] Meyer F. – Vectorial levelings and flattenings. *Dans : Mathematical Morphology and its Applications to Image Processing, ISMM'2000*, éd. par Goutsias J., Vincent L. et Bloomberg D. S. – Palo Alto, États Unis, 2000.

[59] Meyer F. et Beucher S. – Morphological segmentation. *Journal of Visual Communications and Image Representation*, vol. 1, n1, 1990, pp. 21–45.

[60] Monga O. – *Segmentation d'images par croissance hiérarchique de régions.* – Thèse de doctorat, Université de Paris XI, 1988.

[61] Montanvert A., Meer P. et Rosenfeld A. – Hierarchical image analysis using irregular tessellations. *IEEE Transations on Pattern Analysis and Machine Intelligence*, vol. 13, n4, avril 1991, pp. 307–316.

[62] Morris O., Lee M. et Constantinides A. – Graph theory for images analysis : an approach based on the shortest spanning tree. *Dans : Proceedings of the IEE*, pp. 146–152. – avril 1986.

[63] Mortensen E. N. et Barrett W. A. – Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, vol. 60, 1998, pp. 349–384.

[64] MPEG-4 : Applications document. – Technical Report ISO/IEC JTC1/SC29/WG11/w2724, MPEG,Seoul, Korea, mars 1999.

[65] MPEG-4 : Requirements document. – Technical Report ISO/IEC JTC1/SC29/WG11/w2723, MPEG,Seoul, Korea, mars 1999.

[66] MPEG-7 : Applications document. – Technical Report ISO/IEC JTC1/SC29/WG11/w2860, MPEG,Vancouver, Canada, juillet 1999.

[67] MPEG-7 : Requirements document. – Technical Report ISO/IEC JTC1/SC29/WG11/w2723, MPEG,Vancouver, Canada, juillet 1999.

[68] Nack F. et Lindsay A. – Everything you wanted to know about MPEG7, part i. *IEEE MultiMedia*, juillet-septembre 1999.

[69] Nacken P. F. M. – Image segmentation by connectivity preserving relinking in hierarchical graph structures. *Pattern Recognition*, vol. 28, n6, 1995, pp. 907–920.

[70] Nagamochi H., Ono T. et Ibaraki T. – Implementing an efficient minimum capacity cut algorithm. *Mathematical Programming*, vol. 67, 1994, pp. 325–341.

[71] Najman L. et Schmitt M. – Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Transations on Pattern Analysis and Machine Intelligence*, vol. 18, n12, décembre 1996.

[72] O'Connor N. E. et Marlow S. – Supervised semantic object segmentation and traking via EM-based estimation of mixture density parameters. *Dans : Noblesse Workshop on Non-Linear Model Based Image Analysis.* – Glasgow, Royaume Uni, juillet 1998.

[73] Pal N. R. et Pal S. K. – A review on image segmentation techniques. *Pattern Recognition*, vol. 26, n9, 1993, pp. 1277–1294.

[74] Pardàs M. – *Segmentación morfológica de secuencias de imágenes : aplicación a la codificación.* – Thèse de doctorat, Universitat Politècnica de Catalunya, 1994.

[75] Pardàs M. et Salembier P. – 3D morphological segmentation and motion estimation for image sequences. *Signal Processing*, vol. 38, 1994, pp. 31–43.

[76] Pavlidis T. et Liow Y.-T. – Integrating region growing and edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, n 3, mars 1990, pp. 225–233.

[77] Pereira F. – MPEG-4 : Why, what, how and when ? *Signal Processing : Image Communication*, vol. 15, janvier 2000, pp. 271–279.

[78] Reese L. J. – *Intelligent paint : region-based interactive image segmentation.* – Provo, États Unis, Thèse, Department of Computer Science, Brigham Young University, 1999.

[79] Salembier P. – Morphological multiscale segmentation for image coding. *Signal Processing*, vol. 38, 1994, pp. 359–386.

[80] Salembier P. et Garrido L. – Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Transactions on Image Processing*, vol. 9, n4, avril 2000, pp. 561–576.

[81] Salembier P. et Marqués F. – Region-based representations of image and video : segmentation tools for multimedia services. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, n8, décembre 1999.

[82] Salembier P., Oliveras A. et Garrido L. – Motion connected operators for image sequences. *Dans : European Signal Processing Conference, EUSIPCO'96.* – Trieste, Italie, 10-13 septembre 1996.

[83] Salembier P. et Pardàs M. – Hierarchical morphological segmentation for image sequence coding. *IEEE Transactions on Image Processing*, vol. 3, n5, septembre 1994, pp. 639–651.

[84] Salembier P. et Serra J. – Morphological multiscale image segmentation. *Dans : SPIE Visual Communications and Image Processing*, pp. 620–631. – 1992.

[85] Salembier P. et Serra J. – Flat zones filtering, connected operators, and filters by reconstruction. *EEE Transactions on Image Processing*, vol. 4, n8, août 1995, pp. 1153–1160.

[86] Schmitt M. – Response to the comment on geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, n7, juillet 1998, pp. 764–766.

[87] Selim S. Z. et Ismail M. A. – K-means type algorithms : a generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, n1, 1984, pp. 81–87.

[88] Serra J. – *Image Analysis and Mathematical Morphology.* – Londres, Royaume Uni, Academic Press, 1982.

[89] Serra J. (édité par). – *Image Analysis and Mathematical Morphology. Theoretical Advances.* – Academic Press, 1988.

[90] Serra J. – Connections for sets and functions. *Fundamenta Informaticae*, vol. 20, 2000, pp. 147–186.

[91] Serra J. et Salembier P. – Connected operators and pyramids. *Dans : Proc. SPIE Image Algebra and Morphological Image Processing IV*, pp. 65–76. – San Diego, États Unis, 1993.

[92] Shen X., Spann M. et Nacken P. – Segmentation of 2D and 3D images through a hierarchical clustering based on region modelling. *Pattern Recognition*, vol. 31, n9, 1998, pp. 1295–1309.

[93] Shi J. et Malik J. – Normalized cuts and image segmentation. *Dans : Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 731–737. – 1997.

[94] Shi J. et Malik J. – Motion segmentation and tracking using normalized cuts. *Dans : International Conference on Computer Vision.* – janvier 1998.

[95] Shi J. et Malik J. – Self inducing relational distance and its application to image segmentation. *Dans : Proceedings of the Fifth European Conference on Computer Vision, ECCV.* – Freiburg, Alemagne, 2-6 juin 1998.

[96] Soille P. – *Morphological image analysis.* – Springer-Verlag, 1999.

[97] Subramanian K. R., Lawrence D. M. et Mostafavi M. T. – Interactive segmentation and analysis of fetal ultrasound images. *Dans : 8th EG Workshop on ViSC.* – Boulogne sur Mer, France, avril 1997.

[98] Vachier C. – *Extraction de caractéristiques, segmentation d'image et morphologie mathématique.* – Thèse de doctorat, École Nationale Supérieure des Mines de Paris, 1995.

[99] Vachier C. et Meyer F. – Extinction value : a new measure of persistence. *Dans : IEEE Workshop on Non-Linear Signal and Image Processing*, pp. 254–257. – 1995.

[100] Vincent L. – Morphological area openings and closings for grayscale images. *Dans : Proc.NATO Shape in Picture Workshop.* – Springer Verlag, 1992.

[101] Vincent L. – Greyscale area openings and closings, their efficient implementation and applications. *Dans : Workshop on Mathematical Morphology*, pp. 22–27. – Barcelone, Espagne, mai 1993.

[102] Wang D. – Unsupervised video segmentation based on watersheds and temporal tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, n5, septembre 1998, pp. 539–546.

[103] Wang S. et Siskind J. M. – Image segmentation with minimum mean cut. *Dans : Proceedings of the International Conference on Computer Vision, ICCV.* – Vancouver, Canada, 9-12 juillet 2001.

[104] Wu X. – Adaptive split-and-merge segmentation based on piecewise least-square approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, n8, août 1993, pp. 808–815.

[105] Wu Z. et Leahy R. – An optimal graph theoretic approach to data clustering : theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, n11, novembre 1993, pp. 1101–1113.

[106] Yu X. – *Vision dynamique et morphologie mathématique - Application à l'analyse des scènes routières.* – Thèse de doctorat en morphologie mathématique, École Nationale Supérieure des Mines de Paris, 1993.

[107] Yu X., Beucher S. et Bilodeau M. – Road tracking, lane segmentation and obstacle recognition by mathematical morphology. *Dans : Proc. Intelligent Vehicles'92 Symposium*, pp. 166–170. – Detroit, États Unis, 29 juin 29 - 1 juillet 1992. 2623.

[108] Zanoguera F., Marcotegui B. et Meyer F. – An interactive colour image segmentation system. *Dans : Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'99.* – Berlin, Alemagne, mai-juin 1999.

[109] Zanoguera F., Marcotegui B. et Meyer F. – A toolbox for interactive segmentation based on nested partitions. *Dans : Proceedings of the IEEE International Conference on Image Processing.* – Kobe, Japon, octobre 1999.

[110] Zanoguera F., Marcotegui B. et Meyer F. – A segmentation pyramid for the interactive segmentation of 3-d images and video sequences. *Dans : Mathematical Morphology and its Applications to Image and Signal Processing, ISMM'00*, pp. 223–232. – Palo Alto, États Unis, juin 2000.

[111] Zhong D. et Chang S.-F. – AMOS : An active system for MPEG-4 video object segmentation. *Dans : International Conference on Image Processing.* – Chicago, États Unis, octobre 1998.