Restauration automatique de films anciens

Etienne Decencière

RESTAURATION AUTOMATIQUE DE FILMS ANCIENS

THÈSE

présentée à

l'École Nationale Supérieure des Mines de Paris

par

Etienne DECENCIÈRE FERRANDIÈRE

pour obtenir le titre de

DOCTEUR

en

MORPHOLOGIE MATHÉMATIQUE

Soutenue le 9 Décembre 1997 à Fontainebleau
devant le jury composé de :

Juan J.	VILLANUEVA	*Président et rapporteur*
Moncef	GABBOUJ	*Rapporteur*
Bruno	DESPAS	*Examinateur*
Louis	LABORELLI	*Examinateur*
Jean	SERRA	*Examinateur*
Pascal	TANNHOF	*Examinateur*

À mes parents
y a mis hermanitas
Isabelle y Beatriz.

Remerciements

Ces trois années passées au Centre de Morphologie Mathématique constituent assurément une des périodes les plus enrichissantes de ma vie. Et ceci non seulement d'un point de vue professionnel et scientifique, mais aussi d'un point de vue humain.

Je voudrais remercier pour cela:

Jean Serra, qui a réveillé en moi l'intérêt pour la morphologie mathématique et pour l'analyse d'images en général, et qui m'a offert la possibilité de faire une thèse,

Fernand Meyer, de qui j'ai appris enormément de choses, et avec qui travailler a été un plaisir,

Chantal de Fouquet, qui m'a dévoilé les mystères du krigeage,

Juan José Villanueva, Moncef Gabbouj, Bruno Despas, Louis Laborelli et Pascal Tannhof, qui ont jugé et enrichi mon travail de thèse,

Michel Bilodeau et Dimitri Gorokhovik, qui ont toujours été disposés à m'aider et qui n'ont jamais laissé une question sans réponse (et pourtant, j'ai essayé!),

Liliane Pipault, pour sa gentillesse et éternelle bonne humeur,

Laura Andriamasinoro, pour ses nombreuses relectures soignées et la multitude de "petites questions" auxquelles elle a répondu,

Fabrice Lemonnier pour sa sympathie,

Lothar Bergen et Frédéric Zana, qui ont passé de longues heures à relire la première version de ce document, et avec qui j'ai bu de très nombreux cafés,

les partenaires du projet NOBLESSE, et en particulier Stephan Marshall et son équipe, qui m'ont accueilli chaleureusement à l'Université de Strathclyde, à Glasgow, pendant l'été 1996,

et tous mes autres collègues et amis du laboratoire, grâce à qui ces trois années ont été très agréables. Je pense en particulier à Antoine Aubert, Serge Beucher, Jacek Cichoz, Luc Decker, Claire Hélène Demarty, Michel Gauthier, Cristina Gomila, Christophe Gratin, Ronaldo Hashimoto, Marcin Iwanowski, Dominique Jeulin, Jean-Claude Klein, Pascal Laurenge, Mariusz Mlynarczuk, René Peyrard, Oscar Ribes, Valéry Risson, Raphaël Sasportas, Laurent Savary, Marc Waroquier et Francisca Zanoguera.

Finalement, si la morphologie mathématique a bouleversé ma vie, c'est parce qu'elle m'a amené à rencontrer Bea. A elle seule elle constitue la raison nécessaire et suffisante de mon séjour à Fontainebleau.

Abstract

Most motion pictures produced before the fifties have a very short life span. They are very damaged and their condition continues to deteriorate. It is necessary, and urgent, to restore and preserve them. Besides, if these films were restored, they could be used to feed the rapidly increasing audio-visual market.

Classical restoration methods can correct some defects, but not all. Computer based restoration methods are paid more attention, but most of them treat the frames manually, one by one: results are good, but still very expensive. In order to restore a greater number of motion pictures, new faster techniques must be developed. One way of doing this is to increase the degree of automatism in the restoration process.

This PhD thesis is among the first to propose automatic restoration algorithms for old motion pictures. We give below a list of the defects that we treat, as well as a short description of the corresponding restoration algorithms.

Pompage This defect corresponds to an abnormal variation of the luminance in the scene. In order to eliminate it, we limit the variations of the frames histograms along time.

Jittering We propose a simple method for evaluating the global translation between consecutive frames. Next, we filter the sequence of translations in order to separate the vibrations component from the natural movement of the scene.

Vertical scratches We detect vertical scratches frame by frame using morphological operators. We then spatially interpolate the missing grey levels.

Local random defects All defects that cover a relatively small region of each frame and that most of the time do not appear in the same position between consecutive frames belong to this category.

We detect them by using spatio-temporal connectivity criteria, implemented through openings and closings by reconstruction.

However, when there is fast motion in the scene, these criteria may find problems. In order to solve them, we have developed a new motion compensation algorithm. It is based on a segmentation of the reference image, and uses kriging in order to interpolate and filter the displacement vector field.

Once the detection phase is completed, lost grey levels have to be recovered. We have developed spatial and temporal interpolation methods to restore the missing pixels.

Finally, all these algorithms have been embedded in a restoration system that we have called SARSA, which stands for *Système Automatique de Restauration de Séquences Animées* (Automatic Restoration System for Animated Sequences). We have used it to restore severely damaged sequences with success.

Résumé

La plus grande partie des films tournés avant les années cinquante ont été tirés sur des pellicules dont la durée de vie est assez courte. Ils ont déjà subi d'importants dommages et ils continuent à se dégrader. Il est nécessaire, et même urgent, de les restaurer. Par ailleurs, grâce au développement considérable des marchés audiovisuels, si ces films sont remis en état, ils peuvent connaître une deuxième jeunesse.

Les méthodes physico-chimiques existantes permettent de corriger un certain nombre de types de dégradations, mais pas toutes. Les outils informatiques occupent aujourd'hui une place de plus en plus importante dans l'industrie de la restauration cinématographique et video. Cependant, la plupart des techniques employées traitent les images une par une, à la main, ce qui certes permet d'obtenir une qualité excellente, mais qui est très coûteux en temps, et par conséquent en argent. Pour pouvoir restaurer un plus grand nombre de films anciens, il faut développer des méthodes plus rapides.

Cette thèse est parmi les premières à proposer des techniques de restauration automatique de films anciens. Il suffit qu'un opérateur choisisse les paramètres de la restauration, tels que les types de défauts à considérer, pour que nos algorithmes traitent sans intervention extérieure des séquences entières d'images. Cette approche permet d'accélérer considérablement la vitesse de traitement.

Nous donnons dans ce qui suit une liste des défauts que nous traitons, ainsi qu'un très court résumé de la méthode de restauration employée.

Pompage Ce défaut se caractérise par une variation indésirable de l'éclairage de la scène au cours du temps. Nous le traitons en limitant les variations de l'histogramme entre deux images consécutives, tout en autorisant une certaine dérive pour ne pas interdire les variations naturelles de l'éclairage.

Vibrations Nous proposons une méthode pour mesurer la translation du fond de la scène entre images consécutives. Ensuite, nous filtrons la suite des translations pour estimer les vibrations parasites et les corriger.

Rayures verticales Les rayures verticales, blanches ou noires, sont très courantes dans les films anciens. Nous les détectons en utilisant des opérateurs morphologiques tels que le chapeau haut de forme, puis nous récupérons l'information perdue grâce à des interpolations.

Taches et autres défauts aléatoires Dans cette catégorie nous classons tous les défauts qui apparaissent rarement à la même position sur deux images consécutives du film. Nous les détectons en appliquant des critères de connexité spatio-temporels, mis en œuvre grâce à des ouvertures et des fermetures par reconstruction.

Lorsque le mouvement dans la scène est important les critères de connexité peuvent être pris à défaut. Pour palier cet inconvénient nous avons mis au point un algorithme de compensation de mouvement. Il est basé sur une segmentation de l'image de référence, et utilise le krigeage pour interpoler et filtrer le champ de vecteurs de déplacement.

Nous proposons plusieurs méthodes, certaines spatiales, d'autres temporelles, pour interpoler les textures dans les zones endommagées.

Nous avons bâti un système de restauration réunissant ces différents algorithmes, et nous l'avons appliqué à de nombreux cas pratiques, obtenant de bons résultats.

Table des matières

Notations

Séquence d'images

I: Séquence d'images.
$I(t)$ ou I_t: Trame t de la séquence I.
$I(x, y, t)$ ou $I_t(x, y)$: Valeur de gris du pixel de coordonnées (x, y, t) de la séquence I.
G: Nombre de niveaux de gris de l'image. Les valeurs de gris possibles sont: $0, 1, \ldots G-1$.
X, Y: Dimensions spatiales de l'image I.
T: Dimension temporelle de l'image.
$\mathcal{D}$: Domaine de définition de I. $\mathcal{D} = \{0, 1, \ldots X-1\} \times \{0, 1, \ldots Y-1\} \times \{0, 1, \ldots T-1\}$. Nous noterons $\mathcal{D}_t$ le domaine de définition de I_t, c'est à dire $\{0, 1, \ldots X-1\} \times \{0, 1, \ldots Y-1\}$.

Morphologie Mathématique

δ: Dilatation.
ϵ: Érosion.
ϕ: Fermeture.
γ: Ouverture.

Connexité

δ_s: Dilatation correspondant à la connexité spatiale d'une séquence d'images.
δ_t: Dilatation correspondant à la connexité temporelle de la séquence.
CCM: Abréviation de composante connexe maximale.

Chapitre 1

Introduction à la restauration digitale de vieux films

1.1 Contexte

Les films cinématographiques constituent un patrimoine artistique et culturel inestimable. Malheureusement, beaucoup d'entre eux sont définitivement perdus, ou sur le point de l'être, et le reste, dans un délai plus ou moins court, risque de subir le même sort. En effet, d'une part on estime que 90% des films muets ont disparu, ainsi que 50% des films tournés avant 1950, et d'autre part la grande majorité des films existants sont voués à disparaître avant un siècle ou deux [24].

Depuis quelques années on commence à prendre conscience de cette situation critique. Les conservateurs de films ont entrepris un travail gigantesque d'amélioration des conditions d'entreposage des bobines, et de transfert des films sur des pellicules à durée de vie plus longue. Dans la plupart des cas, il est alors nécessaire de restaurer le film avant le transfert, opération qui peut être très coûteuse. Avec les moyens informatiques sans cesse grandissants, on peut maintenant envisager d'effectuer au moins une partie de cette opération sur ordinateur, et de réduire ainsi les coûts et améliorer la qualité des restaurations obtenues.

Par ailleurs, avec le développement du câble et des chaînes satellite, le besoin en matériel télévisuel croît rapidement. Cette demande pourrait offrir une deuxième jeunesse aux films anciens. En parallèle se développent des supports digitaux, tels que les vidéodisques, et des moyens de diffusion de plus en plus perfectionnés, qui exigent des images de meilleure qualité. Donc on voit apparaître un marché pour la restauration digitale de films et de vidéo. En témoigne le nombre croissant de compagnies qui offrent des services et des outils de restauration.

1.2 Dégradations et restauration

1.2.1 Les causes de la dégradation d'un film

Les informations contenues dans cette section proviennent pour l'essentiel de [12].

La pellicule cinématographique est un moyen de stockage fragile et éphémère. Elle est constituée de trois couches (voir figure 1.1).

1. Un support transparent. Jusqu'aux années 50 ce support était en nitrate de cellulose. Il fut abandonné du fait de son inflammabilité. Actuellement on utilise du triacétate de cellulose ou du polyester. Le premier peut se conserver quelques centaines d'années dans des conditions optimales; dans le cas contraire apparaît le "syndrome du vinaigre", qui provoque la dégradation des colorants. Le polyester est le support le plus stable; on pense que dans des conditions de stockage idéales il peut être conservé plus de 500 ans.

2. Une couche de gélatine contenant les émulsions photosensibles (soit à base d'argent pour les films en noir et blanc, soit à base de colorants organiques pour les films couleur).

3. Une couche anti-abrasion pour protéger les émulsions.

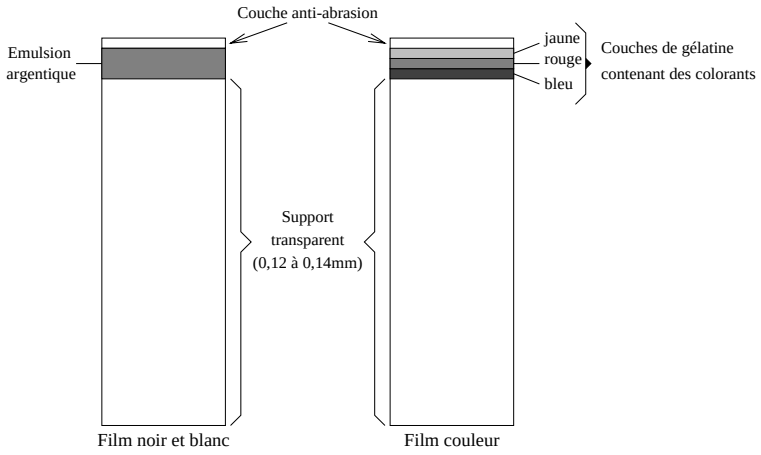


FIG. 1.1 - *Coupe d'un film noir et blanc et d'un film couleur.*

Le film peut se dégrader avec le temps. Les facteurs responsables de cette détérioration sont les suivants.

1. L'humidité. Une humidité relative élevée favorise le développement de bactéries et de champignons dans les couches de gélatine. Elle provoque aussi la dégradation des colorants par hydrolyse. De même l'humidité élevée provoque la décomposition du

triacétate de cellulose en acide acétique et en cellulose (syndrome du vinaigre). Une fois cette réaction déclenchée elle s'autocatalyse et ne peut plus être arrêtée. Cette production d'acide acétique contribue à la dégradation des colorants.

2. L'acidité. Il est très important de conserver le film à un pH aussi proche que possible de la neutralité.

3. La température. Une température élevée accélère les réactions chimiques de dégradation et favorise le développement de bactéries et de champignons.

4. Composés chimiques. Les colorants et l'argent peuvent réagir avec certains composés chimiques issus du traitement du film ou apportés par l'atmosphère.

5. Mauvaise manipulation.

1.2.2 Restauration de vieux films

Dans son sens le plus général, la restauration de vieux films est l'ensemble des opérations qui permettent de récupérer une version plus proche de l'originale pour un film donné. Elle englobe aussi bien les recherches historiques destinées à savoir à quoi ressemblait le film original, que le choix de copies de bonne qualité, les opérations de montage et la correction des défauts de la pellicule. Cependant, dans cette thèse, lorsque nous parlerons de restauration, nous nous limiterons à la correction des défauts de la séquence d'images.

Comme dans toute restauration d'une œuvre historique ou artistique, les contraintes de qualité appliquées à la restauration de vieux films sont draconiennes. Il faut réduire au minimum les risques de détérioration supplémentaire pendant le traitement du film.

Depuis quelques années on utilise la puissance de traitement informatique dans le milieu du cinéma. Les applications principales sont la synthèse d'images et les effets spéciaux, mais on commence à l'utiliser aussi dans la restauration.

1.3 La restauration digitale de vieux films

1.3.1 Très bref historique

En 1989, dans un article sur la restauration de films [67], l'auteur ne parle des méthodes digitales que dans la conclusion, et comme un projet à long terme.

Quatre années après, en 1993, le film "Blanche Neige" de Walt Disney est le premier à être restauré en utilisant uniquement des méthodes digitales [11], grâce au procédé Cinéon de Kodak. Cependant, le travail se fait pratiquement image par image et se révèle de ce fait très coûteux.

Cette même année démarre le projet de conservation des archives de nouvelles de la Fox ("Movietone News"), qui contient environ 13 millions de mètres de film 35mm. Trois ans après le stock entier a été numérisé avec une résolution de 1008×1018 et transféré sur bande digitale [77].

La Communauté Européenne a financé plusieurs projets de recherche sur la restauration de films et de séquences vidéo.

Le projet EUREKA LIMELIGHT (07/1994-01/1997) visait à développer des outils de digitalisation de films et des algorithmes de restauration.

Le but du projet ACTS AURORA, commencé en Septembre 1995 et d'une durée de 3 ans, est de mettre au point des techniques qui permettront de restaurer efficacement et rapidement de grandes quantités d'archives audiovisuelles.

Le projet ESPRIT LTR NOBLESSE (01/1996-12/1998) est destiné à développer des modèles non-linéaires orientés objet pour la description d'images et de séquences d'images. Notre travail s'est déroulé dans ce cadre. La restauration automatique de films anciens, et plus généralement de séquences d'images, est l'une des applications visées par NOBLESSE.

Par ailleurs, un signe supplémentaire du développement rapide du marché de la restauration digitale de films est l'apparition de plusieurs compagnies qui offrent du matériel et des services de restauration digitale.

Finalement, la Commission Supérieure Technique de l'Image et du Son (CST) a publié en 1997 le livre "La restauration numérique de films cinématographiques" [12]. Les auteurs décrivent le cadre de la restauration digitale de films, ainsi que les moyens nécessaires pour la mener à bien, mais ne décrivent aucun algorithme de restauration.

1.3.2 La chaîne de restauration digitale

Le sujet de cette thèse implique que nous travaillons avec des images numériques. Or, à l'origine les films se présentent sous forme analogique, soit sur une pellicule, comme dans le cas des films de cinéma, soit sur une bande magnétique vidéo. Donc une première étape de digitalisation est nécessaire.

Cette phase est facile à mettre en œuvre dans le cas de la vidéo, mais pour les films sur pellicule la procédure est beaucoup plus délicate. En effet, si on veut que le transfert ne perde rien de la qualité initiale de la séquence il faut que la digitalisation soit faite avec une résolution très élevée (jusqu'à 4000 points par ligne). Des scanners spécialisés ou des télécinémas haute définition peuvent être utilisés.

Réciproquement, une fois la restauration achevée il faut stocker les images. On pourrait envisager de les sauvegarder sous forme digitale, mais la durée de vie des supports numériques est si courte, et les volumes nécessaires si gigantesques, que cette solution est difficile à mettre en œuvre dans un futur immédiat [24]. L'alternative la plus simple est d'enregistrer le résultat au format vidéo. Mais, dans le cas de la conservation des films cette solution est inacceptable. Le seul choix qui reste alors est le retour sur la pellicule cinématographique, par l'intermédiaire d'un appareil appelé "imageur". Rappelons qu'on estime que les pellicules les plus modernes, dans des conditions idéales de conservation, ont une durée de vie de 500 ans.

La chaîne complète de restauration numérique est résumée par la figure 1.2.

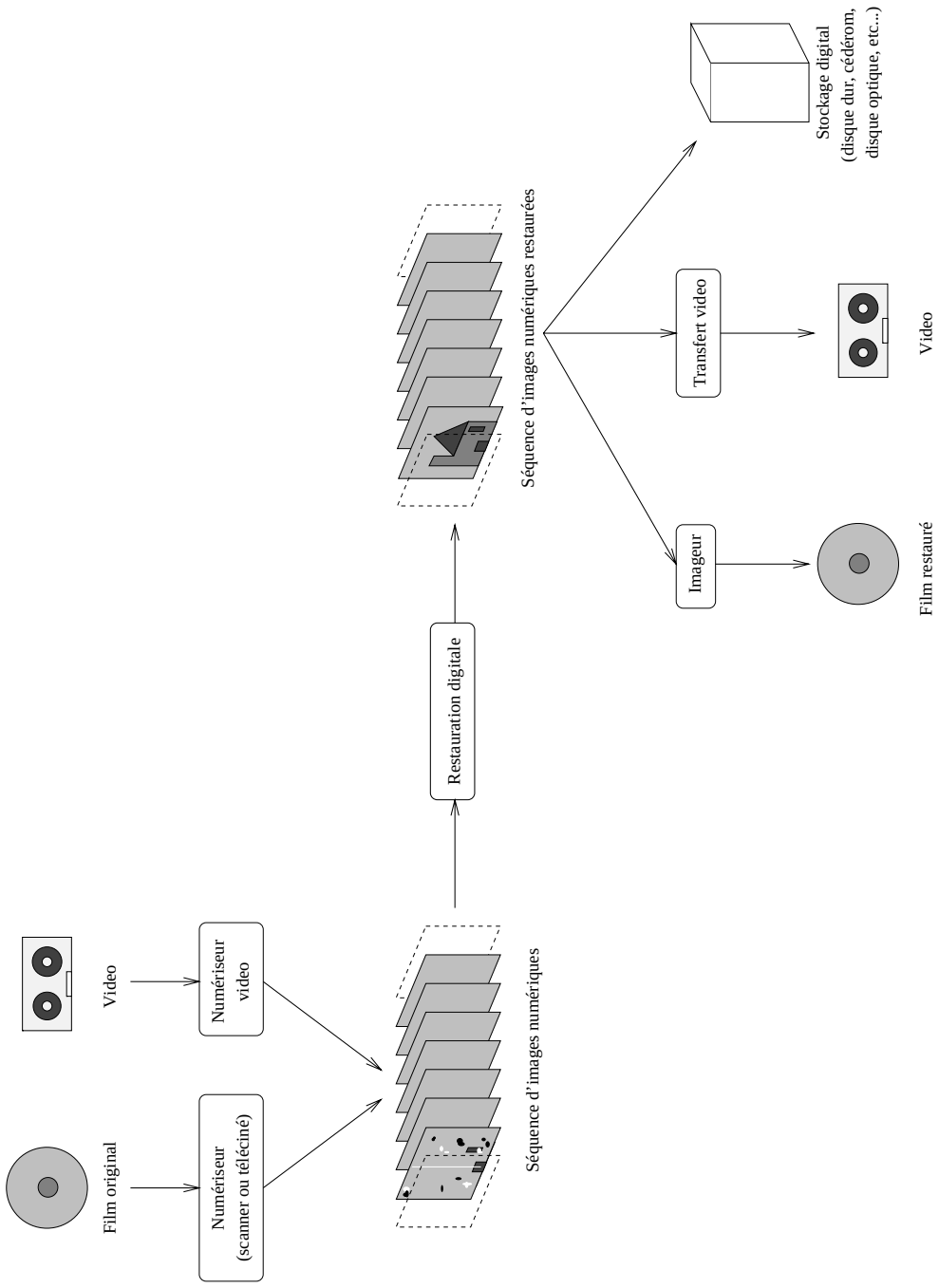


FIG. 1.2 - *Chaîne de restauration d'un film.*

1.3.3 Contraintes

Comme nous l'avons dit, lorsqu'il s'agit de restaurer une œuvre cinématographique les contraintes de qualité sont très importantes. Il faut que le travail soit parfait. Il vaut mieux laisser le film en l'état plutôt que de risquer d'introduire de nouveaux défauts, même si on en enlève beaucoup d'autres. Donc un système de restauration doit être très robuste.

Par ailleurs nous voulons que le système de restauration soit automatique. Nous voulons dire par là que, quitte à choisir certains paramètres à la main, il faut que le système soit capable de traiter une séquence entière du film sans intervention humaine.

Finalement, il faut que le traitement soit rapide par rapport au traitement manuel image par image.

1.4 Objectifs et structure de cette thèse

1.4.1 Objectifs

Dans cette thèse nous mettrons au point de nouvelles méthodes générales d'analyse de séquences d'images et les appliquerons à la restauration de vieux films. Nous développerons aussi des procédés spécifiques pour la correction de certains types de défauts propres aux films anciens. Étant données les contraintes que nous nous sommes imposées, il faut que les algorithmes résultants soient automatiques, robustes et efficaces.

Nous supposerons que chaque séquence destinée à être restaurées correspond à une seule scène du film, c'est à dire qu'elle ne comporte pas de coupures ou *cuts*. Cette hypothèse de base est raisonnable. Des algorithmes de détection de *cuts* sont actuellement en cours de développement.

Les méthodes développées constitueront une boîte à outils servant à restaurer des films anciens. Nous les utiliserons pour monter un système automatique de restauration, qui nous permettra de tester, dans des conditions proches de la réalité, l'efficacité de nos algorithmes.

Une partie des outils développés est basée sur la morphologie mathématique, mais nous utilisons aussi des méthodes provenant de domaines aussi variés que le traitement linéaire du signal et la géostatistique.

Avant de décrire la structure de cette thèse, nous donnerons la classification de défauts que nous avons établie. Chaque chapitre de la première partie de la thèse est destiné à corriger un de ces types de défauts.

1.4.2 Classification des défauts

Cette classification est basée sur les caractéristiques spatio-temporelles des défauts. Elle est fonctionnelle: chaque type de défaut sera traité avec une méthode de restauration spécifique.

1.4.2.1 Défauts globaux

Nous appelons globaux les défauts qui affectent une image en entier. En conséquence de quoi le plus souvent ils n'apparaissent que lorsqu'on voit la séquence en mouvement.

Dans notre travail nous avons rencontré deux de ces défauts: le pompage et les vibrations.

Pompage Le pompage apparaît comme une variation anormale de la luminosité du film au cours du temps. Cette variation peut être périodique ou non. Elle peut avoir plusieurs causes, parmi lesquelles des problèmes de synchronisation lors de la recopie du film.

Vibrations De nombreux films anciens présentent des vibrations parasites. Ce défaut est très gênant. Là encore il peut avoir plusieurs causes. En particulier, il peut être produit lors du transfert du film sur un support vidéo. Ce transfert s'effectue à l'aide d'un appareil appelé "télécinéma", qui peut être mal réglé ou déficient. De même, si la séquence digitale est obtenue grâce à un scanner, les images peuvent ne pas être correctement alignées.

1.4.2.2 Défauts locaux

Les films endommagés sont souvent couverts de taches ou de rayures. Ces défauts ont ceci en commun qu'ils affectent une zone relativement restreinte de chaque image. Nous les appelons donc "défauts locaux". Cette classification est donc uniquement basée sur leurs caractéristiques spatiales. Nous avons défini deux sous-classes en fonction, cette fois-ci, des caractéristiques temporelles de ces défauts.

Défauts locaux immobiles Certains défauts locaux apparaissent à exactement la même position pendant plusieurs images de suite. Nous les appelons défauts locaux immobiles. Le seul exemple que nous ayons rencontré est celui des rayures verticales, blanches ou noires. Elles sont produites par frottement d'une particule sur la pellicule ou sur le négatif pendant la projection ou la recopie du film.

Défauts locaux aléatoires Beaucoup de défauts locaux ne sont pas immobiles: leur position est aléatoire, et donc il est peu probable qu'ils se superposent entre images consécutives. Nous les appelons "défauts locaux aléatoires". Les taches produites par les champignons ou les bactéries, ainsi les dépôts de poussière, appartiennent à cette sorte de défaut.

1.4.2.3 Autres défauts

Dans cette catégorie nous avons classé tous les défauts qui ne rentrent pas dans les catégories précédentes. Nous les appelons graves parce que nous ne voyons pas comment les traiter de façon automatique. Ceci ne veut pas dire que des méthodes numériques ne puissent pas les restaurer.

Par exemple, nous ne savons pas comment détecter de façon automatique l'absence d'une image manquante dans une séquence, surtout quand il n'y a pas de mouvement. Mais si on précise à la main l'emplacement de l'image absente, on peut envisager de l'interpoler à partir des images voisines.

1.4.3 Structure

Les premiers chapitres de cette thèse décrivent les méthodes que nous avons développées pour restaurer les différents types de défauts, présentés dans la section précédente. Les chapitres suivants sont consacrés au krigeage, une méthode d'interpolation utilisée en géostatistique, et à son application à l'estimation du mouvement.

- Chapitre 1: introduction.
- Chapitre 2: cadre de travail et outils de base.

  Nous présentons le cadre mathématique que nous avons adopté pour décrire les opérations que nous appliquons aux images, ainsi que les outils de base de la morphologie mathématique que nous utilisons dans cette thèse.
- Chapitre 3: correction de l'effet de pompage.

  Dans ce chapitre nous proposons plusieurs méthodes pour corriger le pompage dans une séquence. Elles sont toutes basées sur l'analyse de l'histogramme des images successives.
- Chapitre 4: correction des vibrations.

  Nous traitons ensuite les vibrations. Nous commençons par proposer une méthode de détection des translations globales entre deux images, que nous appliquons ensuite à la correction des vibrations. Notre méthode nous permet de séparer les vibrations des vraies translations de la séquence.
- Chapitre 5: correction de défauts locaux immobiles.

  Une fois les méthodes de restauration des défauts globaux présentées, nous passons aux défauts locaux, en commençant par les défaut locaux immobiles, qui se réduisent aux rayures verticales.

  La restauration est divisée en deux phases. Une phase de détection, et une phase de récupération de l'information perdue ou d'interpolation.

  Nous effectuons la détection en combinant des opérateurs de la morphologie mathématique et de simples moyennes. La récupération d'information est réalisée aussi grâce à la morphologie mathématique.
- Chapitre 6: correction des défauts locaux aléatoires.

  Nous nous occupons ensuite des défauts locaux aléatoires. Comme pour les défauts locaux immobiles la restauration est divisée en deux phases.

Pour les détecter nous mettons au point de nouvelles méthodes d'analyse des séquences d'images, basées sur la connexité.

Pour récupérer l'information perdue, nous développons des méthodes d'interpolation aussi bien spatiales que temporelles.

– Chapitre 7: Application du krigeage à la compensation de mouvement.

Nous présentons le krigeage et décrivons le krigeage inverse; nous illustrons leur fonctionnement dans le cadre de la modélisation de texture. Nous généralisons les outils décrits aux champs de vecteurs de mouvement, ce qui nous permet de développer un algorithme de compensation de mouvement orienté objet.

– Chapitre 8: Présentation du système global.

Dans ce chapitre nous assemblons les différents algorithmes présentés dans les chapitres précédents pour construire un système complet de restauration automatique de films anciens.

Chapitre 2

Cadre de travail et outils de base

2.1 Introduction

Dans ce chapitre nous posons le décor pour le travail de cette thèse. Nous commençons par présenter le modèle mathématique adopté pour décrire les séquences d'images, puis nous définissons rapidement les outils de la morphologie mathématique que nous utilisons dans les autres chapitres.

2.2 Modèle mathématique d'images

Nous nous plaçons dès le début dans des espaces digitaux. Pour représenter graphiquement $\mathbb{Z}^2$ nous n'utilisons pas des points, mais des carrés centrés sur ces points (voir figure 2.1).

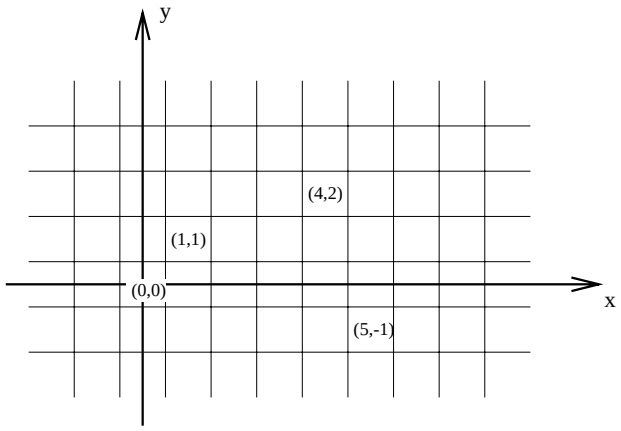


FIG. 2.1 - *Représentation graphique de* $\mathbb{Z}^2$.

2.2.1 Connexité de $\mathbb{Z}^n$

Définition 2.1 *: Relation binaire*
Une relation binaire $\mathcal{R}$ de $\mathbb{Z}^n$ est un sous-ensemble de $\mathbb{Z}^n \times \mathbb{Z}^n$. Soient M et P deux points de $\mathbb{Z}^n$. On dit que P est en relation avec M, et on note $M\mathcal{R}P$, si et seulement si le couple (M, P) appartient à $\mathcal{R}$.

Une relation binaire est dite:

- *réflexive si* $\forall M \in \mathbb{Z}^n \quad M\mathcal{R}M$,
- *symétrique si* $\forall M, P \in \mathbb{Z}^n \quad M\mathcal{R}P \Rightarrow P\mathcal{R}M$,
- *antisymétrique si* $\forall M, P \in \mathbb{Z}^n \quad M\mathcal{R}P,\ P\mathcal{R}M \Rightarrow M = P$,
- *transitive si* $\forall M, P, Q \in \mathbb{Z}^n \quad M\mathcal{R}P,\ P\mathcal{R}Q \Rightarrow M\mathcal{R}Q$.

Nous noterons $\mathcal{R}(M)$ l'ensemble de tous les points en relation avec M:

$$\mathcal{R}(M) = \{P \mid M\mathcal{R}P\} \tag{2.1}$$

Définition 2.2 *: Relation de voisinage*
Soit $\mathcal{V}$ une relation binaire de $\mathbb{Z}^n$. $\mathcal{V}$ est une ***relation de voisinage*** *si et seulement si $\mathcal{V}$ est réflexive et symétrique.*

Cette relation de voisinage définit une **connexité** sur $\mathbb{Z}^n$. Deux points P et M sont dits voisins si et seulement si $P\mathcal{V}M$. L'ensemble des voisins de M est $\mathcal{V}(M)$.

Par exemple, une des relations de connexité les plus simples parmi celles utilisées dans $\mathbb{Z}^2$ est la 4-connexité. Les voisins du point P de coordonnées (x, y) sont alors $\mathcal{V}(P) = \{(x, y), (x+1, y), (x, y+1), (x-1, y), (x, y-1)\}$.

Définition 2.3 *Chemin.*
Soit $\mathcal{V}$ une relation de voisinage de $\mathbb{Z}^n$. Un chemin L de $\mathbb{Z}^n$ est une suite de points $(P_i)_{0 \leq i \leq k-1}$ appartenant à $\mathbb{Z}^n$ telle que:

$$\forall i,\ 0 \leq i \leq k-2 \quad P_{i+1}\mathcal{V}P_i$$

La figure 2.2 montre un chemin entre deux points P et M dans $\mathbb{Z}^2$ en 4-connexité.

Définition 2.4 *Ensemble connexe et composante connexe.*
Soit $\mathcal{V}$ une relation de voisinage. Un sous-ensemble E de $\mathbb{Z}^n$ est dit connexe si et seulement si pour tout couple de points (P, M) appartenant à E il existe un chemin inclus dans E reliant P et M.

Une composante connexe de E est un sous-ensemble connexe de E.

La figure 2.3 montre deux ensembles A et B. Si nous munissons $\mathbb{Z}^2$ de la 4-connexité, alors A est connexe et B ne l'est pas.

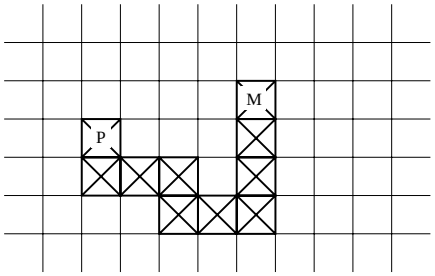


FIG. 2.2 - *Chemin entre les points P et M en 4-connexité. Les points appartenant au chemin sont marqués par des croix. Remarquez que P et M appartiennent aussi au chemin.*

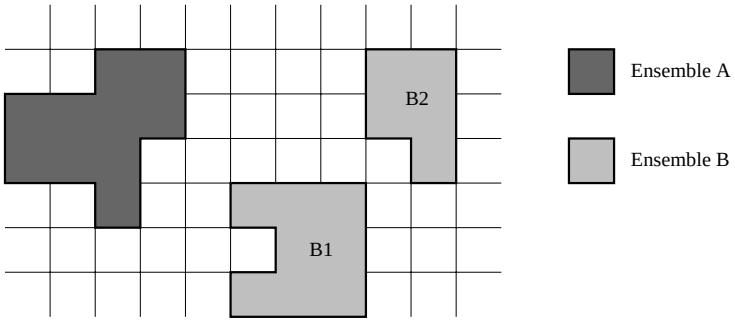


FIG. 2.3 - *En 4-connexité l'ensemble A est connexe, alors que B ne l'est pas.*

Définition 2.5 *Composante connexe maximale*
Soit E un ensemble inclus dans $\mathbb{Z}^n$. Une composante connexe maximale (CCM) de E est une composante connexe de E telle qu'il n'existe pas d'autre composante connexe de E la contenant.

Dans l'exemple de la figure 2.3, l'ensemble A, étant donné qu'il est connexe, ne contient qu'une seule composante connexe maximale, alors que B comporte deux composantes connexes maximales: B_1 et B_2.

2.2.2 Images

Définition 2.6 *: Image de dimension n*
*Soit D un sous-ensemble de $\mathbb{Z}^n$. Soit E un ensemble quelconque. Une image I de dimension n, de support D et à valeurs dans E est une fonction de D dans E. Un point P de D est appelé **pixel**. La valeur de gris de P est tout naturellement $I(P)$.*

Cette définition est très générale. Elle comprend tous les types d'images digitales.

2.2.2.1 Quelques exemples

Par exemple, si nous prenons $E = \{0, \ldots, 255\}$ nous obtenons les images à 256 niveaux de gris. Dans le cas des images RGB on a $E = \{0, \ldots, 255\} \times \{0, \ldots, 255\} \times \{0, \ldots, 255\}$. Nous pouvons même avoir des images de vecteurs en choisissant $E = \mathbb{R}^2$.

Un autre cas intéressant est celui de $E = \{0, 1\}$. On dit alors que I est une **image binaire**. Souvent nous identifierons l'image binaire I au sous-ensemble de D où I vaut 1. Ainsi, au lieu d'écrire pour un pixel P que $I(P) = 1$ nous écrirons: $P \in I$.

Dans cette thèse nous travaillons beaucoup avec des images 2D. Elles sont toutes à support rectangulaire D de dimensions $X \times Y$ du type $D = \{0, \ldots, X-1\} \times \{0, \ldots, Y-1\}$, et à valeurs soit dans $\{0, \ldots, 255\}$, soit dans $\{0, 1\}$.

2.2.2.2 Connexité et images

Soit I une image 2D et P un pixel lui appartenant, de coordonnées (x, y). Les relations de voisinage les plus courantes sont les suivantes (rappelons que D est l'ensemble de définition de l'image):

- **4-connexité**: les voisins de P sont $\mathcal{V}(P) = \{(x, y), (x, y+1), (x, y-1), (x+1, y), (x-1, y)\} \cap D$ (voir figure 2.4).
- **8-connexité**: les voisins de P sont $\mathcal{V}(P) = \{x-1, x, x+1\} \times \{y-1, y, y+1\} \cap D$ (voir figure 2.4).
- **6-connexité**: pour simuler une trame hexagonale on prend comme voisins de P $\mathcal{V}(P) = \{(x, y-1), (x+1, y-1), (x, y), (x+1, y), (x-1, y), (x, y+1), (x+1, y+1)\} \cap D$ si y est pair, et $\mathcal{V}(P) = \{(x-1, y-1), (x, y-1), (x, y), (x+1, y), (x, y), (x-1, y+1), (x, y+1)\} \cap D$ si y est impair. La figure 2.5 illustre l'intérêt de cette définition.

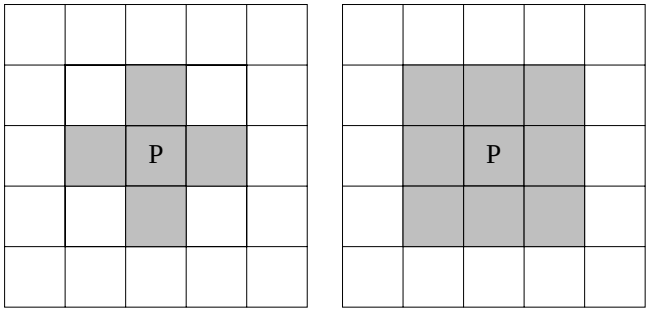


FIG. 2.4 - *Illustration de la 4-connexité et de la 8-connexité*

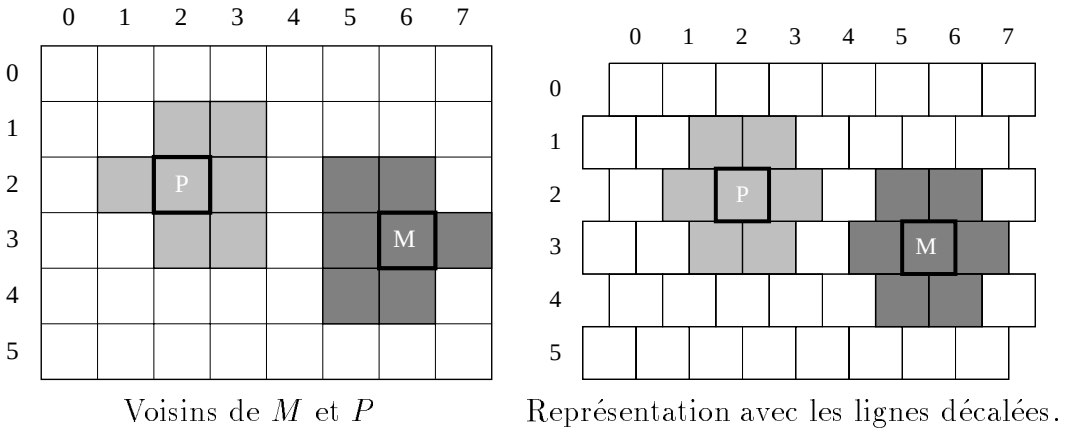


FIG. 2.5 - *Illustration de la 6-connexité*

Grâce à l'augmentation de la puissance de calcul des ordinateurs, les images tridimensionnelles deviennent de plus en plus courantes [32]. D'ailleurs, les séquences animées peuvent être considérées comme des images tridimensionnelles où la troisième dimension correspond au temps [23, 2]. Attardons-nous un peu sur ce cas.

2.2.2.3 Séquences d'images

Une séquence d'images est une suite d'images 2D: $I_0, I_1, \ldots I_n, \ldots I_{T-1}$. T est le nombre d'images de la séquence. Chaque image a les mêmes dimensions $X \times Y$. Nous pouvons alors construire une image tridimensionnelle I, de dimensions $X \times Y \times T$, telle que:

$$I(x, y, t) = I_t(x, y) \tag{2.2}$$

Souvent, nous appellerons **trame** chaque image I_t.

Vu que la dimension temporelle n'a pas du tout le même sens que les deux dimensions spatiales, nous définissons deux connexités partielles pour l'image I.

$\mathcal{V}_s$ est la relation de voisinage spatiale. Donc $\mathcal{V}_s(P)$ donne tous les voisins de P se trouvant dans la même trame que P.

$\mathcal{V}_t$ est la relation de voisinage temporelle. Donc $\mathcal{V}_t(P)$ donne tous les voisins de P se trouvant dans des trames autres que celle contenant P.

La connexité de l'image $\mathcal{V}$ est donnée alors par l'union des deux connexités partielles, *i.e.* $P\mathcal{V}M$ si et seulement si $P\mathcal{V}_sM$ ou $P\mathcal{V}_tM$.

2.2.3 Opérateurs connexes

Maintenant que nous avons défini les images, nous allons décrire une classe d'opérateurs sur celles-ci qui est très utile. Il s'agit des opérateurs connexes [76].

Soit $\mathcal{I}$ l'ensemble de toutes les images à niveaux de gris de dimension n et de support D. Soit I une image de cet ensemble. Nous appelons **opérateur** une fonction de $\mathcal{I}$ dans $\mathcal{I}$

Définition 2.7 *Fonction de seuillage*
*Soit g un niveau de gris (*i.e. *un entier compris entre 0 et 255). Le seuillage de I au niveau g est donné par la fonction:*

$$X_g(I) = \{P \in D \mid I(P) \geq g\} \tag{2.3}$$

$X_g(I)$ est un ensemble qui peut être interprété comme une image binaire.
De même nous définissons le seuillage vers le bas:

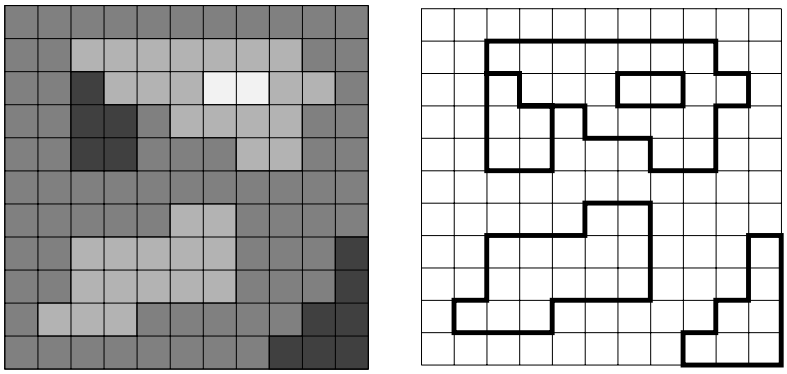
$$X^g(I) = \{P \in D \mid I(P) \leq g\} \tag{2.4}$$

Remarquons alors que $X_g(I) \cap X^g(I)$ est l'ensemble des pixels P de I dont la valeur de gris est égale à g.

Définition 2.8 *Zone plate*
Le sous-ensemble C de $\mathbb{Z}^n$ est une zone plate de l'image I s'il existe un niveau de gris g tel que C soit une composante connexe maximale de $X_g(I) \cap X^g(I)$.

Théorème 2.1 *Partition associée à une image*
L'ensemble des zones plates d'une image I à niveaux de gris constitue une partition de son ensemble de définition D. Nous l'appellerons partition associée à I.

Dans la figure 2.6 nous illustrons cette définition et ce théorème dans le cas d'une image simple à 4 niveaux de gris.



(a) Image à 4 niveaux de gris (b) Zones plates de l'image

FIG. 2.6 - *Exemple d'une image à niveaux de gris (a) et les zones plates associées (b).*

Définition 2.9 *Partition plus fine qu'une autre*
Une partition est dite plus fine qu'une autre si chaque élément de la première est inclus dans un élément de la deuxième.

Définition 2.10 *Opérateurs connexes [76].*
Soit ω un opérateur sur les images de $\mathcal{I}$. L'opérateur ω est connexe si et seulement si pour toute image I de $\mathcal{I}$ la partition associée à I est plus fine que la partition associée à $\omega(I)$.

Les opérateurs connexes ont été définis par Serra et Salembier [76]. Ce sont des outils de traitement d'image très puissants. En effet, par définition ils ne créent pas de nouveaux contours sur les images. Ils ont été d'abord appliqués aux images 2D avec succès, par exemple pour des applications de segmentation [72]. Par la suite ils ont aussi été appliqués aux séquences d'images [66, 49, 70].

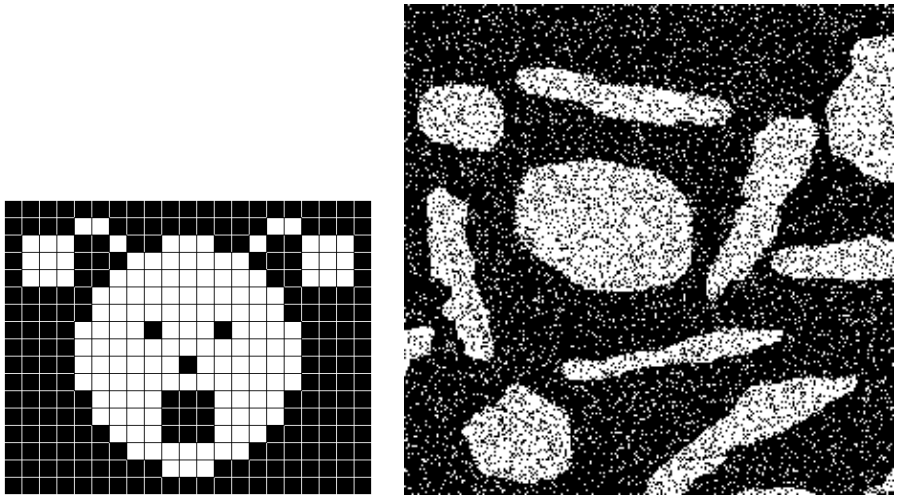
Dans la section suivante nous construirons des opérateurs connexes grâce à la morphologie mathématique.

2.3 Morphologie mathématique

La morphologie mathématique est une branche des mathématiques basée sur l'étude des ensembles qui permet d'analyser les structures géométriques. Ses bases théoriques ont été établies par Matheron [55] et Serra [73, 74].

Dans cette section nous allons décrire uniquement les outils de la morphologie mathématique que nous utilisons dans la suite de cette thèse.

Nous illustrerons le fonctionnement des opérateurs présentés grâce à deux images. La première est une image synthétique de faible résolution (figure 2.7(a)). La deuxième, apparaissant sur la figure 2.7(b) représente des particules très bruitées. Sa taille est de 256×256 pixels. Nous avons représenté en blanc les pixels appartenant à l'ensemble, et en noir les pixels appartenant au fond.



(a) (b)

FIG. 2.7 - *Images binaires utilisées dans les tests.*

2.3.1 Treillis

La plupart des opérations de la morphologie mathématique sont basées sur la recherche de la borne supérieure ou de la borne inférieure d'un ensemble. Nous allons donc commencer par définir les relations d'ordre et les ensembles où la recherche de ces bornes a un sens.

Définition 2.11 *Relation d'ordre*
Une relation binaire $\geq$ sur un ensemble E est une relation d'ordre si et seulement si elle est réflexive, antisymétrique et transitive.

Un ensemble E muni d'une relation d'ordre est dit **ordonné**. Si pour toute paire x et y d'éléments de E on a soit $x \geq y$, soit $y \geq x$, alors on dit que E est **totalement ordonné**.

Définition 2.12 *Treillis*
Un ensemble E muni d'une relation d'ordre $\geq$ est un treillis si et seulement si toute paire d'éléments $\{x, y\}$ de E admet un plus petit majorant, noté $x \bigvee y$, et un plus grand minorant, noté $x \bigwedge y$.

Définition 2.13 *Treillis complet*
*Un ensemble E muni d'une relation d'ordre $\geq$ est un treillis complet si et seulement si tout sous-ensemble A de E admet un plus petit majorant, noté $\bigvee A$ et appelé borne supérieure, ou plus simplement **sup**, et un plus grand minorant, noté $\bigwedge A$ et appelé borne inférieure ou **inf**.*

Remarquons qu'un treillis complet admet un plus grand élément $\bigvee E$, appelé **élément universel**, et un plus petit élément $\bigwedge E$, appelé **élément nul**.

2.3.1.1 Exemples

Ensemble totalement ordonné: Par exemple, tout ensemble totalement ordonné est un treillis. Mais ils ne sont pas tous complets. Ainsi, $\mathbb{R}$ muni de la relation d'ordre classique $\geq$, n'est pas totalement ordonné, puisqu'il n'admet ni de borne supérieure, ni de borne inférieure. Par contre tout ensemble totalement ordonné ET fini est un treillis complet.

Treillis des images à niveaux de gris Soit $\mathcal{I}$ l'ensemble des images à niveaux de gris de support D. Soient I_1 et I_2 deux images de $\mathcal{I}$. Nous munissons $\mathcal{I}$ de la relation d'ordre définie de la façon suivante: $I_1 \geq I_2$ si et seulement si pour tout pixel P: $I_1(P) \geq I_2(P)$. $\mathcal{I}$ muni de cette relation d'ordre est un treillis complet. Si l'ensemble des niveaux de gris est $\{0, \ldots 255\}$ alors l'élément universel de $\mathcal{I}$ est $\bigvee \mathcal{I} = 255$ (c'est à dire l'image qui vaut partout 255) et l'élément nul est $\bigwedge \mathcal{I} = 0$.

Treillis booléen: Soit E un ensemble quelconque. L'ensemble $P(E)$ des parties de E, muni de la relation d'ordre d'inclusion, n'est pas un ensemble totalement ordonné si E contient au moins deux éléments. Cependant, il constitue un treillis complet, avec $\bigvee P(E) = E$ et $\bigwedge P(E) = \emptyset$. Ce treillis est appelé **treillis booléen**.

Dans un premier temps nous commençons par définir les dilatations et les érosions dans le cas binaire, c'est à dire dans le treillis booléen. Ensuite nous généraliserons à des images à niveaux de gris.

2.3.2 Dilatations et érosions

Nous nous plaçons dans le treillis booléen $P(\mathbb{Z}^n)$.

Définition 2.14 *Dilatation*
Une fonction $\delta : P(\mathbb{Z}^n) \mapsto P(\mathbb{Z}^n)$ *est une dilatation si et seulement si elle commute avec le sup,* i.e. *si et seulement si pour toute paire d'ensembles* A *et* B *inclus dans* $\mathbb{Z}^n$*:*

$$\delta(A \bigvee B) = \delta(A) \bigvee \delta(B) \tag{2.5}$$

Cette définition, que nous avons donnée dans le cas du treillis booléen, est très générale et peut être utilisée dans le cadre de n'importe quel treillis. Dans ce cas précis, nous dirons qu'il s'agit d'une **dilatation binaire**.

Propriété 2.1 *Toute dilatation est croissante.*

Soit $\mathcal{R}$ une relation binaire quelconque de $\mathbb{Z}^n$. Comme précédemment, nous noterons $\mathcal{R}(P)$ l'ensemble des points de $\mathbb{Z}^n$ en relation avec P, *i.e.* $\mathcal{R}(P) = \{M \in D \mid P\mathcal{R}M\}$. La fonction:

$$\mathbb{Z}^n \longrightarrow P(\mathbb{Z}^n) \tag{2.6}$$
$$M \longmapsto \mathcal{R}(M) \tag{2.7}$$

est une **fonction structurante** (voir [74]).

Théorème 2.2 *La fonction* δ *suivante est une dilatation ([74], chapitre 3):*

$$\begin{aligned} P(\mathbb{Z}^n) &\longrightarrow P(\mathbb{Z}^n) \\ C &\longmapsto \bigcup_{P \in C} \mathcal{R}(P) \end{aligned} \tag{2.8}$$

Nous dirons que δ *est la dilatation associée à* $\mathcal{R}$*.*

Réciproquement, si δ est une dilatation, alors on peut définir une relation $\mathcal{R}$ de la façon suivante: M est en relation avec P si et seulement si M appartient à $\delta(\{P\})$.

Proposition 2.1 *Soit* $\mathcal{R}$ *une relation binaire de* $\mathbb{Z}^n$ *et* δ *sa dilatation associée.*

- *Si* $\mathcal{R}$ *est réflexive alors* $P \in \mathcal{R}(P)$*, donc* δ *est extensive.*
- *Si* $\mathcal{R}$ *est transitive, alors* δ *est idempotente.*

Exemple: reconstruction géodésique Les relations binaires permettent donc de construire très simplement des dilatations. Par exemple, soit A un sous-ensemble de $\mathbb{Z}^n$, muni d'une certaine connexité. Soit $\mathcal{R}$ la relation binaire définie par: $M\mathcal{R}P$ si et seulement si M et P appartiennent à la même composante connexe de A. On vérifie simplement que $\mathcal{R}$ est symétrique et transitive (mais attention, elle n'est pas réflexive: un point n'appartenant pas à A n'est en relation avec aucun point, *a fortiori* pas avec lui même!). Le fait qu'elle soit transitive implique que la dilatation associée δ est idempotente. Nous verrons dans la suite qu'il s'agit d'une reconstruction géodésique. Nous la noterons $\delta(B) = Rec(B, A)$.

Relation de voisinage et dilatation Si $\mathcal{R}$ est une relation de voisinage, que nous notons $\mathcal{V}$, alors la dilatation δ associée est extensive (du fait de la réflexivité de $\mathcal{V}$) et telle que si $P \in \delta(\{M\})$ alors $M \in \delta(\{P\})$ (conséquence de la symétrie). Et réciproquement, si δ vérifie ces deux propriétés, alors la relation binaire associée est une relation de voisinage.

Supposons maintenant que la relation $\mathcal{R}$ est invariante par translation. En d'autres termes, supposons que pour tout vecteur $\vec{t}$ et pour toute paire de points M et P de $\mathbb{Z}^n$, si M est en relation avec P alors $M + \vec{t}$ est en relation avec $P + \vec{t}$. Alors la dilatation associée sera aussi invariante par translation. Il suffira de connaître les points en relation avec l'origine O de $\mathbb{Z}^n$ pour pouvoir ensuite calculer l'opérateur δ. Ces points constituent l'**élément structurant** S de la dilatation δ: $S = \mathcal{R}(O)$.

Nous avons alors:

$$\mathcal{R}(P) = \{P + M \mid M \in S\} \tag{2.9}$$

Cette deuxième moitié de l'égalité ressemble à l'expression de la somme de Minkovski, dont la définition générale est:

$$A \oplus B = \{P + M \mid P \in A,\ M \in B\} \tag{2.10}$$

Nous obtenons donc finalement la définition d'une dilatation par un élément structurant S.

Définition 2.15 *Dilatation binaire par un élément structurant*
Soient C et S deux sous-ensembles de $\mathbb{Z}^n$. La dilatation de C avec l'élément structurant S est:

$$\delta_S(C) = C \oplus S \tag{2.11}$$

La figure 2.8 montre nos deux images test (voir figure 2.7) dilatées par un élément structurant carré 3×3.

Définition 2.16 *Dilatation géodésique*
Soient A et B deux sous-ensembles de $\mathbb{Z}^n$. La dilatation géodésique de A dans B avec l'élément structurant S est donnée par:

$$\delta_S^B(A) = \delta_S(A) \cap B \tag{2.12}$$

Si A est inclus dans B, et si la dilatation δ_S est associée à la connexité choisie, alors l'itération à l'infini de cette opération nous donne les composantes connexes maximales de B dont l'intersection avec A n'est pas vide. Nous obtenons donc la reconstruction géodésique $Rec(A, B)$ de B à partir de A, que nous avions décrite plus haut. Cette opération, utilisée

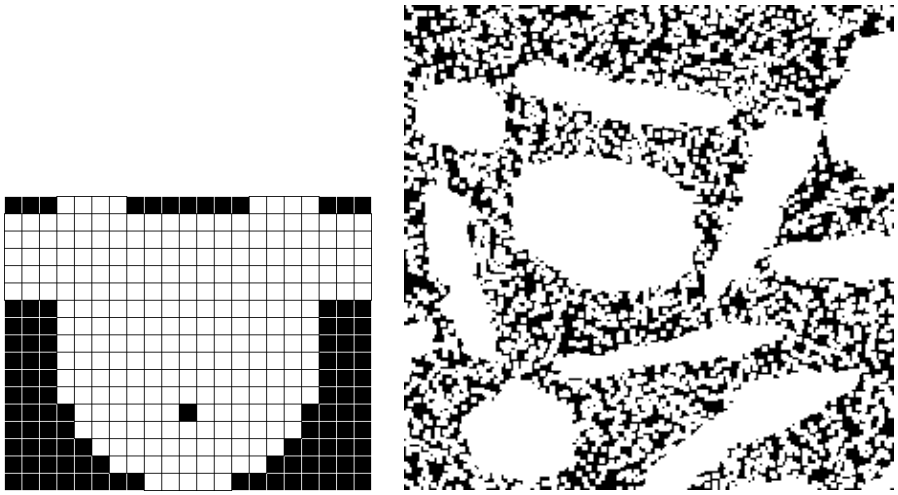


FIG. 2.8 - *Images test dilatées par un élément structurant carré* 3×3.

pour la première fois par Klein [40], permet d'isoler les composantes connexes maximales d'une image, à condition de disposer des marqueurs nécessaires.

Définition 2.17 *Érosion*
Une fonction $\epsilon : P(\mathbb{Z}^n) \mapsto P(\mathbb{Z}^n)$ *est une érosion si et seulement si elle commute avec l'inf,* i.e. *si et seulement si pour toute paire d'ensembles* A *et* B *inclus dans* $\mathbb{Z}^n$*:*

$$\epsilon(A \bigwedge B) \quad = \quad \epsilon(A) \bigwedge \epsilon(B) \tag{2.13}$$

Propriété 2.2 *Toute érosion est croissante.*

Pour définir l'érosion par un élément structurant nous commençons par définir la soustraction de Minkovski:

$$A \ominus B = \{M \ / \ \forall P \in B, M + P \in A\} \tag{2.14}$$

Définition 2.18 *Érosion binaire par un élément structurant*
Soient C *et* S *deux sous-ensembles de* $\mathbb{Z}^n$*. L'érosion de* C *avec l'élément structurant* S *est:*

$$\epsilon_S(C) = C \ominus S \tag{2.15}$$

La figure 2.9 montre nos deux images test érodées par un élément structurant carré 3×3.

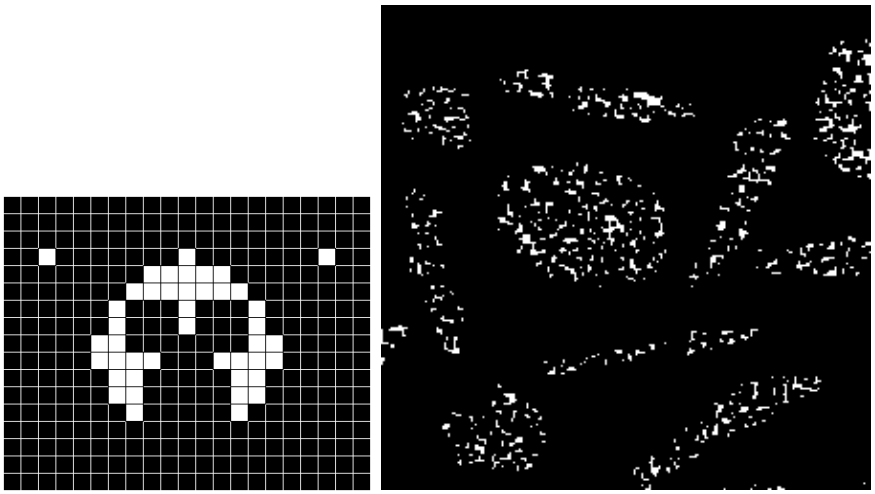


FIG. 2.9 - *Images test érodées par un élément structurant carré de taille* 3×3.

La soustraction de Minkowski n'est pas l'inverse de l'addition de Minkowski. En général l'ensemble A n'est pas égal à $A \oplus S \ominus S$ ni à $A \ominus S \oplus S$. Ceci dit, ces suites d'opérations, *i.e.* $\epsilon_S \circ \delta_S$ et $\delta_S \circ \epsilon_S$, appelées respectivement **fermeture** et **ouverture morphologiques**, ont des propriétés intéressantes, comme nous verrons dans la section qui suit.

2.3.3 Fermetures et ouvertures

Définition 2.19 *Fermeture et ouverture algébriques*
Un opérateur sur le treillis booléen $P(\mathbb{Z}^n)$ *est une fermeture (resp. ouverture) algébrique si et seulement si il est croissant, extensif (resp. anti-extensif) et idempotent.*

Cette définition ne nous dit pas comment construire de tels opérateurs. Un moyen possible est de combiner dilatations et érosions. On obtient alors des fermetures et ouvertures morphologiques.

Théorème 2.3 *Fermeture et ouverture morphologiques*
Soient δ *et* ϵ *une dilatation et une érosion par un même élément structurant* S. *Alors* $\delta\epsilon$ *et une ouverture algébrique appelée ouverture morphologique (notée* γ_S*), et* $\epsilon\delta$ *est une fermeture algébrique appelée fermeture morphologique (notée* ϕ_S*).*

Théorème 2.4 *L'inf de fermetures morphologiques est une fermeture morphologique et le sup d'ouvertures morphologiques est une ouverture morphologique.*

Les ouvertures et fermetures morphologiques sont très utiles. Intuitivement, elles permettent d'effacer les parties d'une image qui sont plus petites que l'élément structurant S. Par ailleurs elles permettent de construire des opérateurs plus complexes tels que les filtres alternés séquentiels [80, 74] ou les chapeaux haut-de-forme [57] ("tophat" en anglais, voir section 2.3.5).

La figure 2.10 montre nos deux images test ayant subi une fermeture par un élément structurant carré de taille 3×3, et l'image 2.11 montre le résultat de l'ouverture avec le même élément structurant.

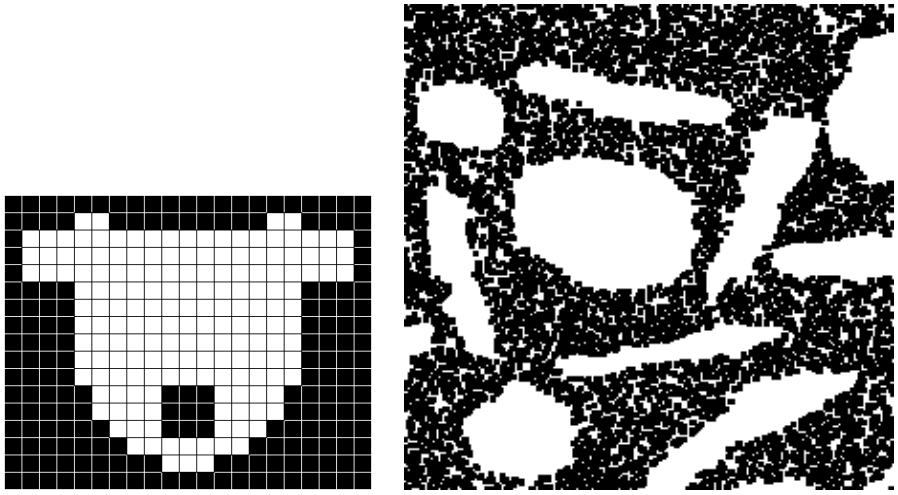


FIG. 2.10 - *Images binaires ayant subi une fermeture par un élément structurant carré de taille* 3×3.

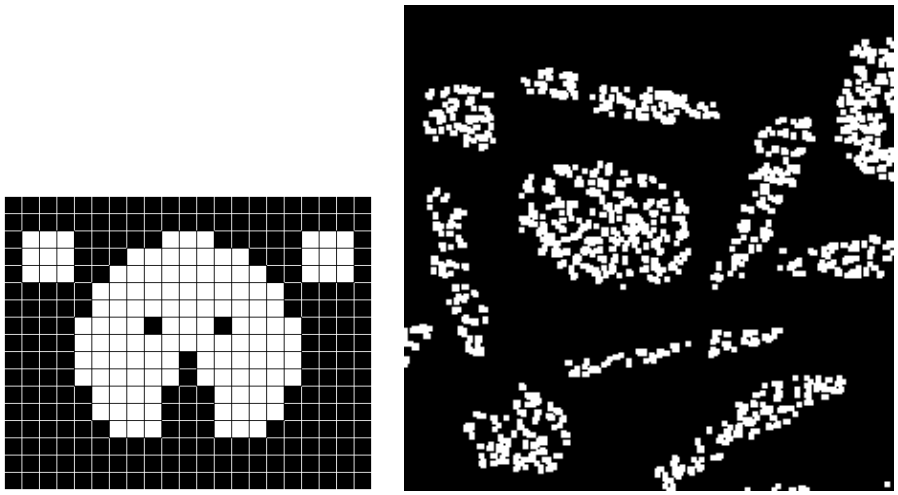


FIG. 2.11 - *Image binaires ayant subi une ouverture par un élément structurant carré de tailla* 3×3.

Cependant, le filtrage résultant peut modifier les contours des objets importants, ce qui pour certaines applications peut être gênant. Pour éviter ce problème les ouvertures et fermetures par reconstruction ont été développées [40].

2.3.3.1 Ouvertures et fermetures par reconstruction

Le principe de l'ouverture par reconstruction est le suivant: soit A un sous-ensemble de $\mathbb{Z}^n$. Soient $A_0, A_1, \ldots, A_n$ ses composantes connexes maximales (CCM). On prend une à une ces CCM et on leur fait subir un certain test. Si la CCM passe le test, on la garde, sinon on l'efface. Ceci peut être représenté par une fonction m qui à une composante connexe associe soit 0, soit 1. Le résultat de l'ouverture est donc:

$$\gamma(A) = \bigcup_{j=0}^{n} \{A_j \mid m(A_j) = 1\} \tag{2.16}$$

La fermeture par reconstruction est l'opération duale:

$$\phi(A) = \gamma(A^c)^c \tag{2.17}$$

Théorème 2.5 *Les ouvertures et fermetures par reconstruction sont des opérateurs connexes [74].*

En pratique on construit m à l'aide d'une **image marqueur** D:

- $m(A_j) = 0$ si $A \cap D = \emptyset$,
- $m(A_j) = 1$ si $A \cap D \neq \emptyset$

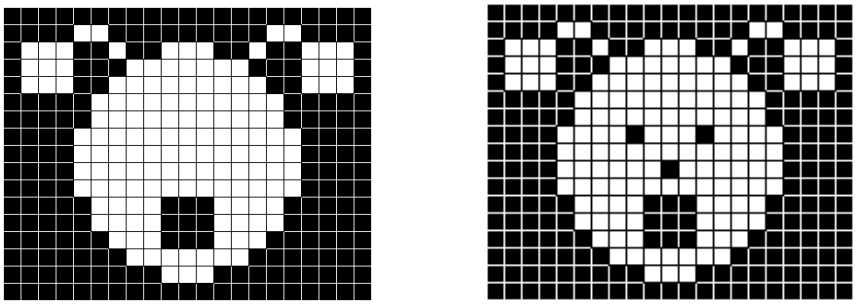
L'image A est souvent appelée **masque**.

Les résultats obtenus avec ces opérateurs sont très intéressants. Ceci dit, il reste le problème du choix des marqueurs. Par ailleurs, pour mettre en œuvre la reconstruction il faut choisir une connexité.

Revenons à notre image test synthétique 2.7(a). Nous allons l'utiliser comme masque dans une fermeture et une ouverture par reconstruction. Nous travaillerons en 8-connexité. Pour la fermeture nous utilisons comme marqueur la fermeture morphologique de cette même image test, qui apparaît dans la figure 2.10. Le résultat apparaît dans la figure 2.12(a). De même pour l'ouverture nous utilisons comme marqueur l'ouverture morphologique (figure 2.12(b)). Observons que la fermeture par reconstruction a bouché les yeux et le nez du visage sans modifier les autres contours de l'image, alors que l'ouverture par reconstruction n'a pas modifié l'image initiale car celle-ci comporte une seule composante connexe qui n'est pas complètement effacée par l'ouverture morphologique.

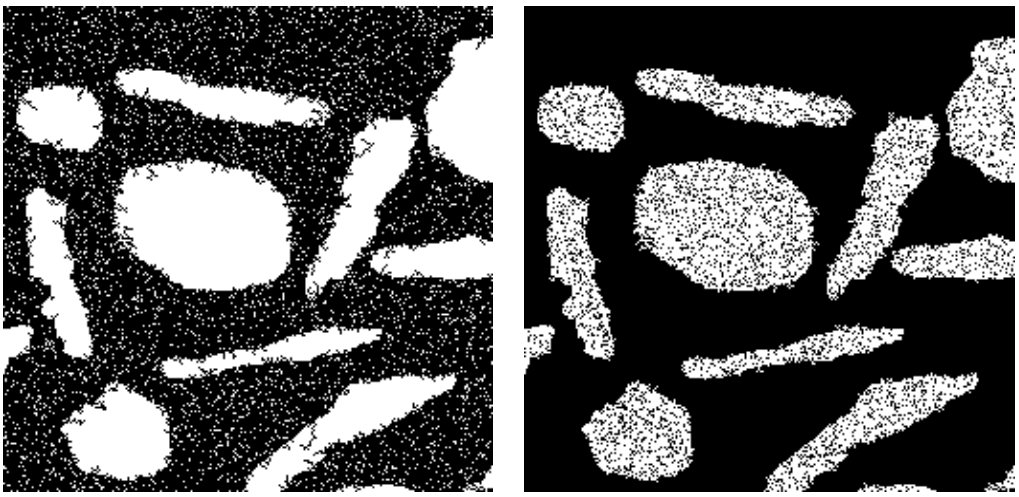
Les mêmes séquences d'opérations appliquées à la deuxième image binaire 2.7(b) donne les résultats de la figure 2.13.

Nous avons réussi à enlever le bruit de façon très satisfaisante, dans le premier cas des particules, dans le deuxième cas du fond. Les contours des particules ont d'ailleurs été préservés: ils sont identiques aux contours de l'image initiale 2.7(b).



(a) Fermeture par reconstruction (b) Ouverture par reconstruction

FIG. 2.12 - *Illustration de la fermeture et de l'ouverture par reconstruction. (a) Fermeture par reconstruction en 8-connexité en utilisant comme masque notre image synthétique et comme marqueur la fermeture morphologique de cette même image. (b) Ouverture par reconstruction en 8-connexité en utilisant comme masque la même image test et comme marqueur l'ouverture morphologique.*



(a) Fermeture par reconstruction (b) Ouverture par reconstruction

FIG. 2.13 - *Illustration de la fermeture et de l'ouverture par reconstruction. (a) Fermeture par reconstruction en 8-connexité en utilisant comme masque notre image test et comme marqueur la fermeture morphologique de cette même image. (b) Ouverture par reconstruction en 8-connexité en utilisant comme masque l'image test et comme marqueur l'ouverture morphologique.*

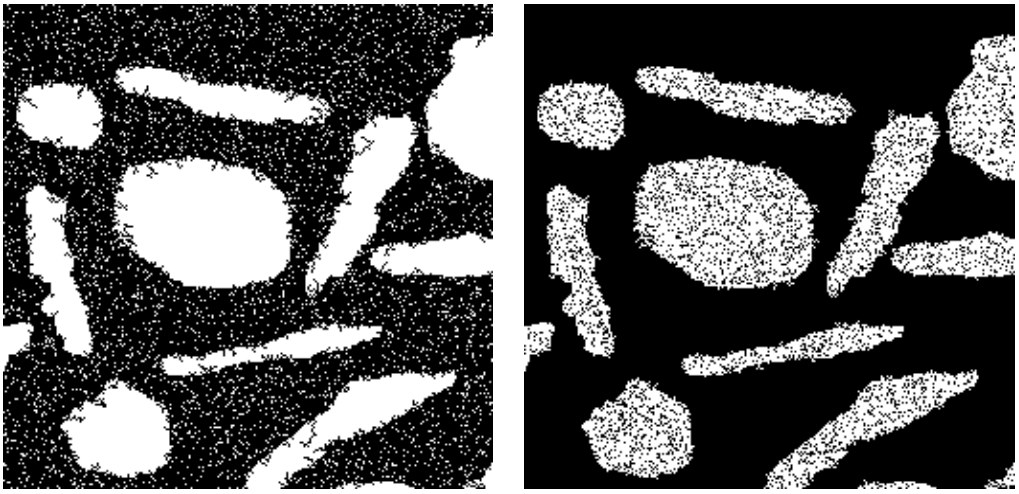
2.3.3.2 Filtres aréolaires

Les **filtres aréolaires** [85, 86], dits aussi filtres surfaciques, constituent un autre type de fermetures et ouvertures par reconstruction. Le principe de l'ouverture aréolaire est très simple: il consiste à garder les CCM de l'image initiale A dont la surface est supérieure à un certain paramètre λ. En d'autres termes l'ouverture aréolaire γ_λ^a s'exprime de la façon suivante:

$$\gamma_\lambda^a(A) = \bigcup_{j=0}^{n} \{A_j \mid Aire(A_j) \geq s_0\} \tag{2.18}$$

La fermeture aréolaire est l'opération duale: elle bouche les trous dont la surface est inférieure à λ.

L'application à notre image test de ces deux opérateurs, avec $\lambda = 50$, donne les images apparaissant sur la figure 2.14. Remarquez que le résultat est très similaire à celui obtenu avec les filtres par reconstruction.



(a) Fermeture aréolaire (b) Ouverture aréolaire

FIG. 2.14 - *Application d'une fermeture aréolaire (a) et d'une ouverture aréolaire (b) avec* $\lambda = 50$ *à notre image test.*

2.3.4 Application aux images

Dans les sections précédentes nous avons décrit les opérateus morphologiques dans le cadre des sous-ensembles de $\mathbb{Z}^n$. Afin de les appliquer aux images (chose que nous avons faite dans nos exemples sans expliquer comment) nous devons résoudre deux problèmes:

- Comment faire pour passer d'un support infini à un support fini D?
- Comment les appliquer à des images à niveaux de gris?

2.3.4.1 Support fini

En fait nous avons déjà montré des exemples d'applications aux images binaires mais nous n'avons pas expliqué comment nous avons fait. L'application n'est pas si triviale car les images, telles que nous les avons modélisées, sont définies sur un support borné D, alors que $\mathbb{Z}^n$ est infini. Donc nous ne pouvons pas parler d'invariance par translation ni franchir le pas entre fonction structurante et élément structurant.

En théorie, avec la connexité telle que nous l'avons définie pour les images, nous pourrions nous passer des éléments structurants, mais en pratique ils sont utiles. Donc nous plongerons notre image I, en tant que fonction de D dans $\{0, \ldots G-1\}$, dans l'ensemble des fonctions de $\mathbb{Z}^n$ dans $\{0, \ldots G-1\}$ et nous donnerons une valeur g_0 aux points n'appartenant pas à D. La fonction I' résultante est donc définie de la façon suivante:

$$I'(M) = \begin{cases} I(M) \; si \; M \in D \\ g_0 \; sinon \end{cases} \tag{2.19}$$

La valeur g_0 choisie dépendra de l'opérateur utilisé:

- Dilatation: $g_0 = 0$,
- Érosion: $g_0 = G - 1$

Les ouvertures et fermetures sont construites à partir des dilatations et érosions.

2.3.4.2 Généralisation aux images à niveaux de gris

Nous pouvons décomposer une image à niveaux de gris I en fonctions binaires à l'aide de seuillages:

$$I = \sum_{g=1}^{G-1} X_g(I) \tag{2.20}$$

Ensuite, on peut appliquer l'opérateur ρ à chaque image binaire $X_g(I)$, obtenant ainsi des images $\rho(X_g(I))$. Pour obtenir $\rho(I)$ il faut alors empiler les images binaires $\rho(X_g(I))$; pour cela il suffit que ρ soit croissant:

$$\rho(I) = \sum_{g=1}^{G-1} \rho(X_I(g)) \tag{2.21}$$

Ceci dit, en pratique cette façon de faire serait très lente. Des algorithmes efficaces ont été mis au point qui permettent d'effectuer ces opérations de façon très efficace (voir par exemple le travail de Luc Vincent [84]).

2.3.5 Autres opérateurs

Les dilatations et érosions morphologiques permettent de construire d'autres opérateurs. Décrivons rapidement ceux que nous utiliserons dans les chapitres suivants. I est une image à niveaux de gris et B un élément structurant.

2.3.5.1 Chapeau haut-de-forme

Le chapeau haut-de-forme, inventé par F. Meyer [57], permet de mettre en évidence les détails d'une image.

Définition 2.20 *Chapeau haut-de-forme blanc*
Le chapeau haut-de-forme blanc avec l'élément structurant B, noté hfb_B, est égal à la différence entre l'image originale et l'ouverture morphologique γ_B:

$$hfb_B(I) = I - \gamma_B(I) \tag{2.22}$$

L'effet d'un chapeau haut-de-forme blanc est de faire ressortir tous les détails clairs de l'image plus petits que l'élément structurant.

Définition 2.21 *Chapeau haut-de-forme noir*
Le chapeau haut-de-forme noir avec l'élément structurant B, noté hfn_B, est égal à la différence entre la fermeture morphologique ϕ_B et l'image originale:

$$hfn_B(I) = \phi_B(I) - I \tag{2.23}$$

L'effet d'un chapeau haut-de-forme noir est de faire ressortir tous les détails sombres de l'image plus petits que l'élément structurant.

La figure 2.15 montre un exemple de chapeau haut-de-forme noir.

(a) Image originale (b) Chapeau haut-de-forme noir

FIG. 2.15 - *Image originale* 256×256 *et son chapeau haut-de-forme noir comme élément structurant un carré* 3×3.

2.3.5.2 Gradient de Beucher

Le gradient de Beucher, ou gradient morphologique [73], est défini par:

$$grad(I) = \delta(I) - \epsilon(I) \tag{2.24}$$

Cet opérateur sert à détecter les contours des images. Il est très utilisé dans des applications de segmentation.

La figure 2.16 montre un exemple de gradient de Beucher.

FIG. 2.16 - *Gradient de Beucher*

2.4 Retour sur la connexité

Nous avons vu que nous avons une équivalence entre l'ensemble des dilatations binaires et l'ensemble des dilatations sur un treillis. En particulier nous pouvons associer à une relation de voisinage $\mathcal{V}$ une dilatation unique δ.

Dans le cas des images 2D, nous adopterons les notations suivantes:

- δ_{s4}: dilatation associée à la 4-connexité.
- δ_{s8}: dilatation associée à la 8-connexité.

Remarquons au passage, comme l'a fait Serra dans [75], que si jamais nous désirons une connexité plus forte il suffit d'itérer la dilatation. Ainsi $\delta_{s4} \circ \delta_{s4}$ nous donne une connexité dont la portée est plus longue que la 4-connexité. Nous noterons la dilatation résultante δ_{2s4}. De façon générale nous poserons: $\delta_{ns4} = \delta_{s4}^n$.

Maintenant appliquons cette façon de construire des connexités 2D à des séquences d'images. Comme nous avons dit dans la section 2.2.2.3 nous distinguons deux connexités

partielles: une connexité spatiale δ_s à l'intérieur de chaque trame, et une connexité temporelle δ_t entre trames voisines, la connexité globale de la séquence étant alors donnée par $\delta_{st} = \delta_s \bigvee \delta_t$. En pratique nous choisirons comme connexité spatiale soit la 4-connexité δ_{s4}, soit la 8-connexité δ_{s8}, qu'éventuellement nous pourrons rendre plus fortes en itérant les dilatations.

Nous utiliserons aussi 3 connexités temporelles (M est un pixel quelconque de coordonnées (x, y, t)):

- Connexité temporelle simple: $\delta_{t0}(M) = \{(x, y, t-1), (x, y, t+1)\}$
- 4-connexité temporelle: $\delta_{t4}(M) = \delta_{s4}(x, y, t-1) \bigvee \delta_{s4}(x, y, t+1)$
- 8-connexité temporelle: $\delta_{t8}(M) = \delta_{s8}(x, y, t-1) \bigvee \delta_{s8}(x, y, t+1)$

Comme pour les connexités spatiales, dans certains cas nous utiliserons des connexités temporelles δ^n_{t4} et δ^n_{t8} plus fortes, définies de la façon suivante:

$$\delta_{nt4}(M) = \delta^n_{s4}(x, y, t-1) \bigvee \delta^n_{s4}(x, y, t+1) \quad (2.25)$$

$$\delta_{nt8}(M) = \delta^n_{s8}(x, y, t-1) \bigvee \delta^n_{s8}(x, y, t+1) \quad (2.26)$$

Attention, cette notation est trompeuse: remarquez que $\delta_{nt4}(M)$ n'est pas égal à δ_{t4} itéré n fois!

Nous utiliserons ces différents types de connexité dans l'analyse des séquences en mouvement.

2.5 Conclusion

Dans ce chapitre nous avons décrit le modèle que nous avons adopté pour décrire les séquences d'images, ainsi que le vocabulaire de base que nous utiliserons. Ceci nous a permis ensuite de présenter les outils de base de la morphologie mathématique que nous employons dans cette thèse. Nous avons accordé une place privilégiée à la notion de connexité car elle sera essentielle dans notre travail.

Cependant, nous ne nous limiterons pas à la morphologie mathématique, et nous utiliserons aussi des opérateurs linéaires.

Chapitre 3

Correction de l'effet de pompage

3.1 Introduction

Le pompage dans un film apparaît sous la forme d'une variation anormale de la luminance au cours de la séquence. Il peut être provoqué par divers facteurs, parmi lesquels nous pouvons citer les défauts de synchronisation au moment de la recopie du film, ou des problèmes au moment du tournage.

Ce type de défaut est très courant et gênant. Cependant, nous n'avons pas trouvé dans la littérature des méthodes de restauration du pompage. Est-ce parce qu'il est trop simple? Analysons de plus près ce défaut.

3.2 Analyse du problème

Dans ce chapitre nous avons effectué nos tests avec la séquence "Max" échantillonnée avec un pas de 8 (obtenant ainsi une séquence de 77 images de taille 160×121) et avec la séquence "Train" échantillonnée avec un pas de 4 (séquence de 40 images de taille 128×128).

Jetons un coup d'œil à l'évolution du maximum, de la moyenne et du minimum des niveaux de gris des trames successives dans nos deux séquences d'images test: "Max"(cf. figure 3.1) et "Train"(cf. figure 3.2). Afin de limiter les effets du bruit, nous avons exclus les 1% de pixels à valeur de gris la plus petite, et les 1% à valeur de gris la plus grande dans le calcul des valeurs extrémales de l'histogramme.

Comme nous pouvons le constater, les variations de ces mesures sont très irrégulières. Par ailleurs la périodicité de l'effet de pompage est difficile à mettre en évidence, et en tout cas sa période est assez importante (une cinquantaine de trames dans le cas de la séquence "Max").

Considérons maintenant les histogrammes de deux trames consécutives dans une séquence et essayons de voir si une transformation simple (*i.e.* une transformation linéaire ou affine de l'image) permet de passer de l'un à l'autre. Dans les cas simples c'est possible, mais souvent ce n'est pas le cas (voir par exemple la figure 3.3).

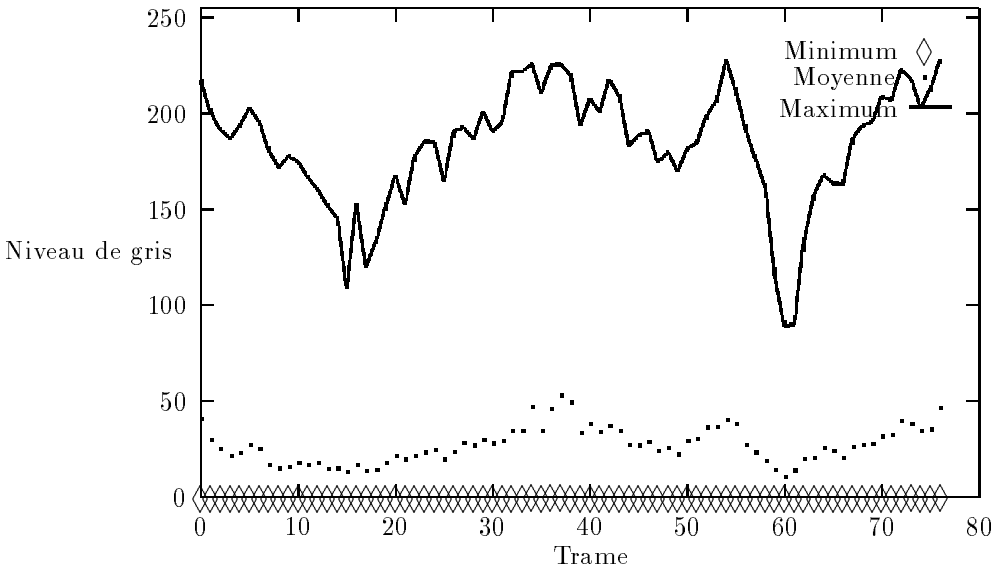


FIG. 3.1 - *Minimum, moyenne et maximum des niveaux de gris en fonction de la trame pour la séquence Max*

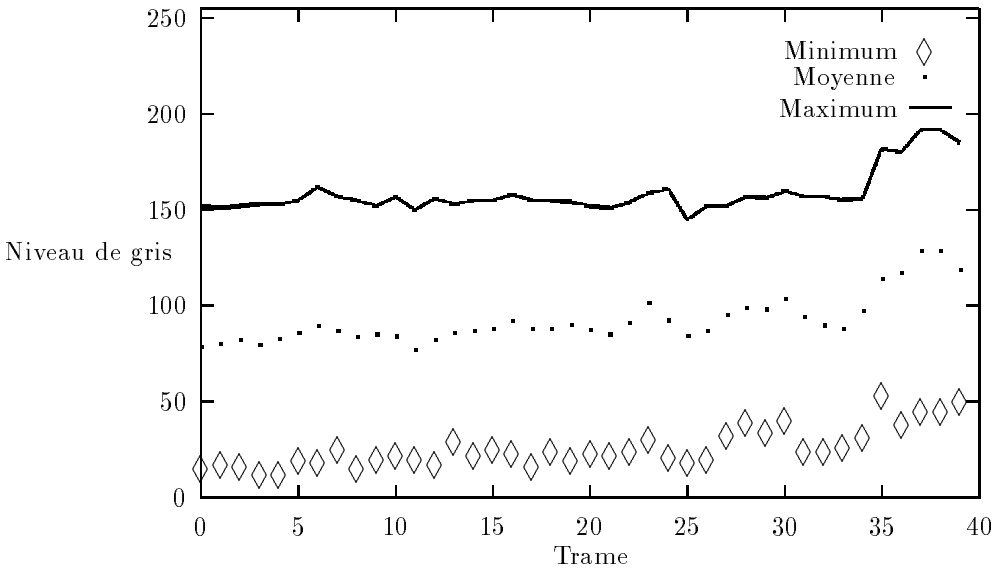


FIG. 3.2 - *Minimum, moyenne et maximum des niveaux de gris en fonction de la trame pour la séquence Train*

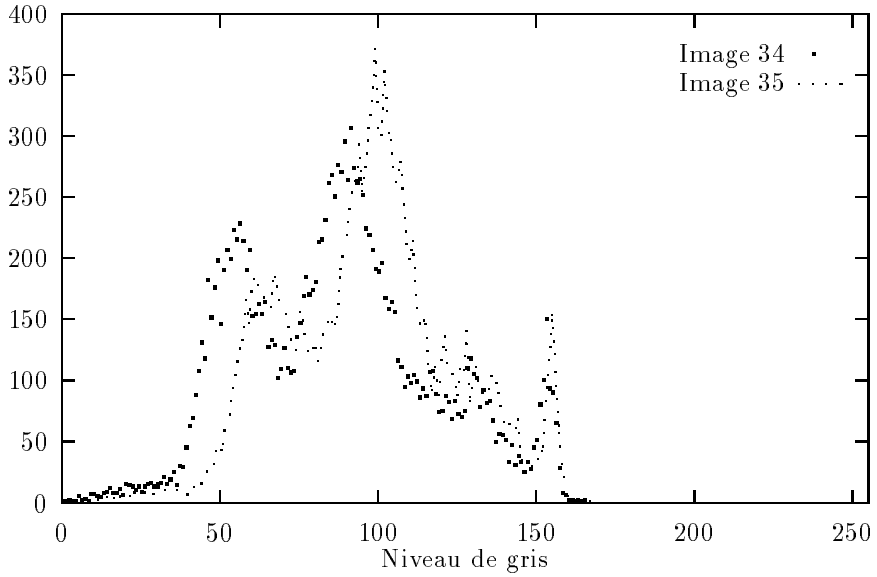


FIG. 3.3 - *Comparaison des histogrammes des images 34 et 35 de la séquence Train*

Ce problème devient encore plus sérieux lorsque l'image a perdu de l'information, soit à cause d'un effet de saturation, soit parce que le contraste a beaucoup diminué ("écrasement" latéral de l'histogramme). Ainsi, si nous comptons le nombre de niveaux de gris effectivement utilisés par image dans le cas de la séquence "Max" (cf. figure 3.4) nous constatons que celui-ci diminue de façon dramatique pour certaines trames. Par exemple la trame 60 comporte seulement 127 niveaux de gris, alors que la plupart des autres trames en comptent plus de 200. D'ailleurs les trames les plus pauvres sont aussi celles pour lesquelles la moyenne des niveaux de gris est la plus basse (cf. 3.1).

Enfin, souvent l'effet de pompage n'affecte pas de façon homogène toute l'image. Dans certains cas il est plus marqué près des bords.

Dans ce chapitre nous développons des méthodes de correction d'histogramme dans une séquence qui permettent de réduire l'effet du pompage.

Dans la suite nous effectuerons nos tests avec la séquence "Max". Nous prendrons comme référence quatre images (cf. figure 3.5). Remarquons que la dégradation entre l'image 0 et l'image 3 n'est pas très sévère, alors que l'image 60 est sérieusement endommagée.

3.3 État de l'art

Nous n'avons pas trouvé dans la littérature des méthodes de correction de l'effet de pompage. Cependant P. Richardson et D. Suter proposent dans [69] une méthode de régularisation d'histogramme destinée à améliorer les résultats d'un algorithme de compensation

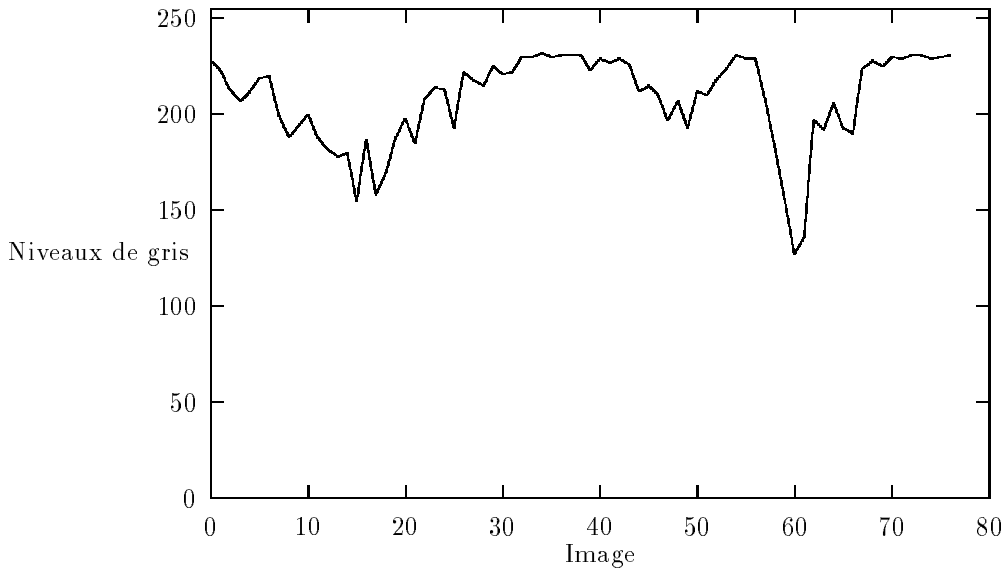


FIG. 3.4 - *Nombre de niveaux de gris utilisés par image de la séquence Max.*

Image 0 Image 3 Image 54 Image 60

FIG. 3.5 - *Images test de la séquence "Max"*

de mouvement. Pour chaque trame I_t indépendamment, ils calculent la valeur minimale m_b et maximale M_b des niveaux de gris, qu'ils utilisent pour normaliser I_t et obtenir ainsi une nouvelle image I_t^* définie par (P est un pixel quelconque):

$$\begin{array}{rcl} I_t(P) \leq m_b & \Rightarrow & I_t^*(P) = 0 \\ m_b < I_t(P) < M_b & \Rightarrow & I_t^*(P) = (I_t(P) - m_b)/(M_b - m_b) \times (G-1) \\ M_b \leq I_t(P) & \Rightarrow & I_t^*(P) = G \end{array}$$

Afin de réduire l'effet du bruit, pour calculer m_b et M_b les auteurs enlèvent de l'histogramme les $b \times N$ pixels de plus petite valeur de gris, et les $b \times N$ pixels de plus grande valeur de gris, où N est le nombre total de pixels de l'image. En d'autres termes, m_b est le mode $N \times b$ et M_b le mode $N - (N \times b)$ de l'histogramme des valeurs de gris. En pratique ils utilisent $b = 0,01$.

Nous nous sommes inspirés de cette méthode pour construire une de nos méthodes de correction du pompage.

3.4 Recalage des histogrammes

Les méthodes de correction du pompage que nous présentons dans la suite sont toutes basées sur l'analyse des histogrammes des images. De plus elles sont récursives: pour corriger la trame t nous utilisons la trame $t-1$. Dans la suite nous supposons donc que la trame I_{t-1} a été restaurée, récupérant ainsi la trame originale I_{t-1}^o. Nous utilisons celle-ci pour corriger l'effet de pompage dans la trame I_t grâce à des hypothèses de faible variation de l'histogramme au cours du temps.

3.4.1 Première méthode : utilisation de la moyenne

La première méthode que nous avons développée est très simple. Elle est basée sur l'hypothèse que la trame courante a été endommagée par une variation linéaire de la luminance:

$$I_t = \alpha_t I_t^o \tag{3.1}$$

où I_t^o est l'image originale, avant dégradation, donc inconnue, et α_t est un réel qui dépend uniquement de la trame t et que nous cherchons à estimer. En faisant la moyenne sur toute l'image nous obtenons:

$$\bar{I}_t = \alpha_t \bar{I}_t^o \tag{3.2}$$

et **en supposant que la moyenne de la trame originale I_t^o est identique à la moyenne de la trame originale précédente I_{t-1}^o**, nous obtenons:

$$\bar{I}_t = \alpha_t \bar{I}_{t-1}^o \tag{3.3}$$

d'où:

$$\alpha_t = \frac{\bar{I}_t}{\bar{I}_{t-1}^o} \tag{3.4}$$

Ce qui nous permet de calculer α_t. Ensuite il suffit de diviser I_t par α_t pour obtenir l'image I_t^o.

Évidemment nous ne pouvons pas appliquer cette procédure à la première image de la séquence. Nous supposerons ici qu'elle est correcte, hypothèse qui est risquée, car si elle s'avérait fausse toute la séquence serait du coup détériorée par la restauration! Nous verrons dans les sections suivantes comment contourner ce problème.

Nous avons appliqué cette méthode à la séquence "Max". L'analyse des résultats apparaît dans la figure 3.6. On remarquera tout de suite que la plupart des images sont saturées. La moyenne n'est pas tout à fait constante à cause justement des problèmes de saturation.

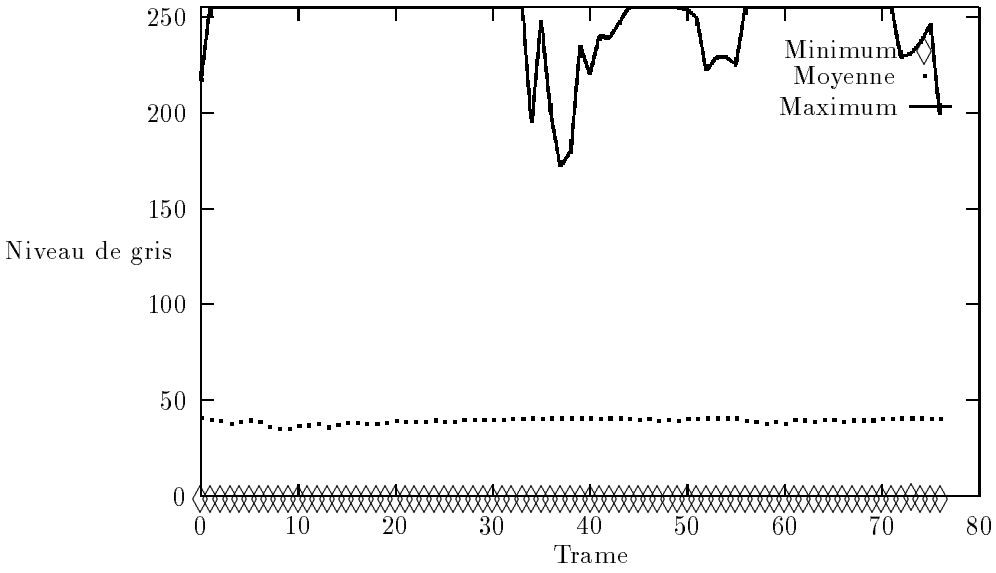


FIG. 3.6 - *Minimum, moyenne et maximum des valeurs de gris pour la séquence Max restaurée avec la méthode de la moyenne.*

Regardons le résultat sur nos images de référence (figure 3.7). L'image 3 est trop saturée, l'image 54 est inchangée, et l'image 60 est légèrement meilleure, mais reste très mauvaise. Et pourtant, elles ont toutes pratiquement la même moyenne de niveaux de gris!

Cette méthode n'est satisfaisante que dans les cas les plus simples. La raison principale qui explique cette limitation est la non vérification de l'hypothèse de base, c'est à dire le caractère linéaire de la dégradation. Cette constatation nous a amenés à élaborer un modèle de dégradation affine, inspiré de la méthode que P. Richardson et D. Suter décrivent dans [69].

Image 0 Image 3 Image 54 Image 60

FIG. 3.7 - *Images test de la séquence "Max" dont l'effet de pompage a été corrigé avec la méthode de la moyenne*

3.4.2 Recalage d'histogrammes à partir de leurs valeurs extrémales

3.4.2.1 Description de la méthode

Nous supposons maintenant que la dégradation de l'histogramme est affine:

$$I_t = \alpha_t I_t^o + \beta_t = \lambda_t(I_t^o) \tag{3.5}$$

Pour calculer les deux paramètres de la transformation affine, nous ferons l'hypothèse que **les valeurs extrémales des histogrammes originaux sont constantes au cours du temps**.

Soit $M()$ la fonction qui à une trame I_t associe sa $(bN+1)$-ième valeur de gris maximale (nous ne prenons pas en compte les $b \times N$ valeurs les plus grandes pour limiter l'effet du bruit). Alors l'équation 3.5 nous permet d'écrire:

$$M(I_t) = M(\alpha_t I_t^o + \beta_t) = \alpha_t M(I_t^o) + \beta_t \tag{3.6}$$

or d'après l'hypothèse de conservation des valeurs extrémales au cours du temps:

$$M(I_t^o) = M(I_{t-1}^o) \tag{3.7}$$

d'où:

$$M(I_t) = \alpha_t M(I_{t-1}^o) + \beta_t \tag{3.8}$$

De même, si $m()$ est la fonction qui à une trame associe son minimum sans prendre en compte les $b \times N$ valeurs les plus petites, nous obtenons:

$$m(I_t) = \alpha_t m(I_{t-1}^o) + \beta_t \tag{3.9}$$

Les équations 3.8 et 3.9 nous permettent de calculer nos deux inconnues α_t et β_t.

Vu que α_t ne peut pas être nul, nous pouvons inverser la fonction affine $\lambda_t(I) = \alpha_t I + \beta_t$, qui nous servira à calculer la trame restaurée I_t^o. Pour tout pixel P de I_t:

$$\begin{aligned} 0 \leq \lambda_t^{-1}(I_t(P)) \leq G-1 &\Rightarrow I_t^o(P) = \lambda_t^{-1}(I_t(P)) \\ \lambda_t^{-1}(I_t(P)) < 0 &\Rightarrow I_t^o(P) = 0 \\ G-1 < \lambda_t^{-1}(I_t(P)) &\Rightarrow I_t^o(P) = G-1 \end{aligned}$$

3.4.2.2 Assouplissement de la méthode présentée

La méthode précédente donne un moyen simple et rapide de corriger le problème du pompage dans une séquence d'images, mais en même temps elle suppose que le niveau de gris moyen de chaque trame est constant au cours du temps. Cette hypothèse n'est pas toujours vrai: l'éclairage de la séquence peut varier et de nouveaux objets apparaître.

Afin d'écarter cet inconvénient, au lieu d'imposer l'égalité des valeurs extrémales de l'histogramme entre deux trames consécutives, nous allons autoriser une certaine variation δ entre les deux trames:

- Si $M(I_{t-1}^o) - \delta \leq M(I_t) \leq M(I_{t-1}^o) + \delta$ alors $M(I_t^o) = M(I_t)$
- Si $M(I_t) < M(I_{t-1}^o) - \delta$ alors $M(I_t^o) = M(I_{t-1}^o) - \delta$
- Si $M(I_{t-1}^o) + \delta < M(I_t)$ alors $M(I_t^o) = M(I_{t-1}^o) + \delta$

Nous calculerons de façon analogue le minimum de la trame courante restaurée:

- Si $m(I_{t-1}^o) - \delta \leq m(I_t) \leq m(I_{t-1}^o) + \delta$ alors $m(I_t^o) = m(I_t)$
- Si $m(I_t) < m(I_{t-1}^o) - \delta$ alors $m(I_t^o) = m(I_{t-1}^o) - \delta$
- Si $m(I_{t-1}^o) + \delta < m(I_t)$ alors $m(I_t^o) = m(I_{t-1}^o) + \delta$

Reste à savoir comment choisir δ. Si nous traitons une séquence où nous savons que la luminance moyenne ne varie pratiquement pas, alors nous pouvons choisir $\delta = 0$ et nous ramener ainsi au cas non souple. Dans les autres cas, et en particulier si nous voulons que le système soit entièrement automatique, nous conseillons de prendre une valeur faible mais non nulle de δ. Dans nos applications nous avons choisi $\delta = 1$.

Par ailleurs, rappelons que nous avions supposé que la première trame de la séquence n'avait pas subi d'effet de pompage. Nous avions précisé que si cette hypothèse n'était pas vérifiée alors nous risquions d'endommager tout la séquence. Maintenant, grâce à la dérive de l'histogramme que nous autorisons, une non vérification de cette hypothèse entraînera beaucoup moins de problèmes.

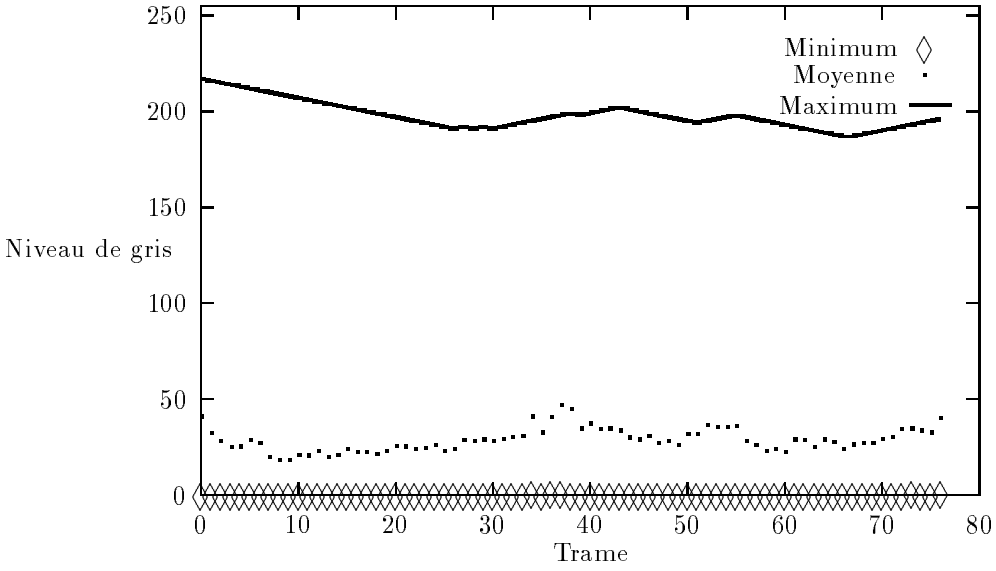


FIG. 3.8 - *Minimum, moyenne et maximum des valeurs de gris pour la séquence Max restaurée avec la méthode basée sur le minimum et le maximum de l'histogramme.*

3.4.2.3 Résultats; avantages et défauts de la méthode

Dans la figure 3.8 nous montrons les résultats obtenus pour la séquence "Max", avec une dérive $\delta = 1$. Nous pouvons constater que maintenant les maxima et minima des images sont beaucoup plus réguliers, mais que par contre l'évolution de la moyenne est irrégulière.

Qualitativement (voir figure 3.9) nous obtenons des résultats meilleurs que dans le cas de la méthode de la moyenne, mais la correction de l'image 3 n'est pas satisfaisante, l'image 54 a été légèrement assombrie, et l'image 60 reste dégradée.

Image 0 Image 3 Image 54 Image 60

FIG. 3.9 - *Images test de la séquence "Max" restaurées avec la méthode utilisant le maximum et le minimum des valeurs de gris*

Cette méthode a l'avantage d'être plus générale que la précédente car le modèle de dégradation de l'histogramme est plus riche. En particulier elle permet de gérer assez bien

les changements de contraste. Mais d'un autre côté les maxima et minima de l'histogramme, malgré l'utilisation des modes, sont peu robustes.

3.4.3 Méthode mixte basée sur la moyenne et les extréma de l'histogramme

Finalement nous avons produit une troisième méthode qui allie la robustesse de la première avec la richesse de la deuxième.

3.4.3.1 Modèle

Nous supposons toujours que l'effet de pompage se traduit dans une image par une application affine:

$$I_t = \alpha_t I_t^o + \beta = \lambda_t(I_t^o) \tag{3.10}$$

Mais cette fois **nous supposons que la moyenne des niveaux de gris est constante au cours du temps (ou, dans une version assouplie, qu'elle varie au plus de δ_1), et que l'écart entre les extréma se conserve aussi au cours du temps (ou qu'elle varie au plus de δ_2)**. Ceci nous permet de calculer $\bar{I}_t^o$ et $M(I_t^o) - m(I_t^o)$. Or l'équation 3.10 nous permet d'écrire:

$$\bar{I}_t = \alpha_t \bar{I}_t^o + \beta_t \tag{3.11}$$

$$M(I_t) = \alpha_t M(I_t^o) + \beta_t \tag{3.12}$$

$$m(I_t) = \alpha_t m(I_t^o) + \beta_t \tag{3.13}$$

d'où:

$$\begin{cases} \bar{I}_t = \alpha_t \bar{I}_t^o + \beta_t \\ M(I_t) - m(I_t) = \alpha_t (M(I_t^o) - m(I_t^o)) \end{cases} \tag{3.14}$$

Ce système nous donne une solution unique pour α_t et β_t (avec α_t non nul) donc nous pouvons calculer I_t^o:

$$I_t^o = \frac{I_t - \beta_t}{\alpha_t} \tag{3.15}$$

(en n'oubliant pas de ramener les valeurs inférieures à 0, à 0, et les valeur supérieures à 255, à 255).

3.4.3.2 Résultats et commentaires

Nous avons appliqué cette méthode de restauration à la séquence Max, avec $\delta_1 = 0,5$ et $\delta_2 = 1$. L'analyse des résultats apparaît sur la figure 3.10. La moyenne est maintenant stable au cours du temps, et les valeurs extrémales de l'histogramme varient de façon plus régulière.

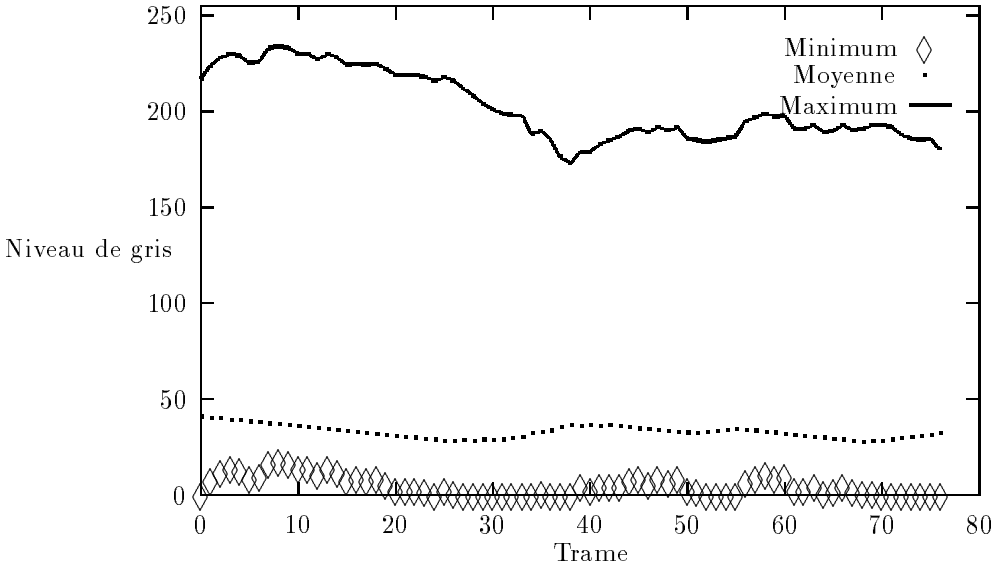


FIG. 3.10 - *Minimum, moyenne et maximum des valeurs de gris pour la séquence Max restaurée avec la méthode mixte.*

Les résultats sur les images test (figure 3.11) sont à peine meilleurs que ceux obtenus avec la méthode précédente, mais lorsqu'on visualise la séquence en mouvement l'amélioration est plus nette grâce à la stabilité accrue de la moyenne.

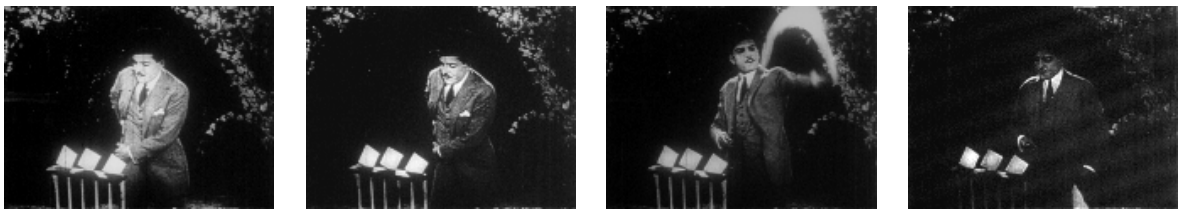


Image 0 Image 3 Image 54 Image 60

FIG. 3.11 - *Images de la séquence "Max" restaurées avec la méthode mixte*

Enfin pour finir revenons à la séquence "Train". L'application de la méthode mixte à cette séquence, certes moins touchée par l'effet de pompage que la séquence "Max", régularise de façon notable les valeurs de l'histogramme au cours du temps, comme nous pouvons le voir dans la figure 3.12 (à comparer avec les valeurs initiales apparaissant dans la figure 3.2). Là encore, lorsque nous visualisons la séquence en mouvement l'amélioration apparaît beaucoup plus clairement.

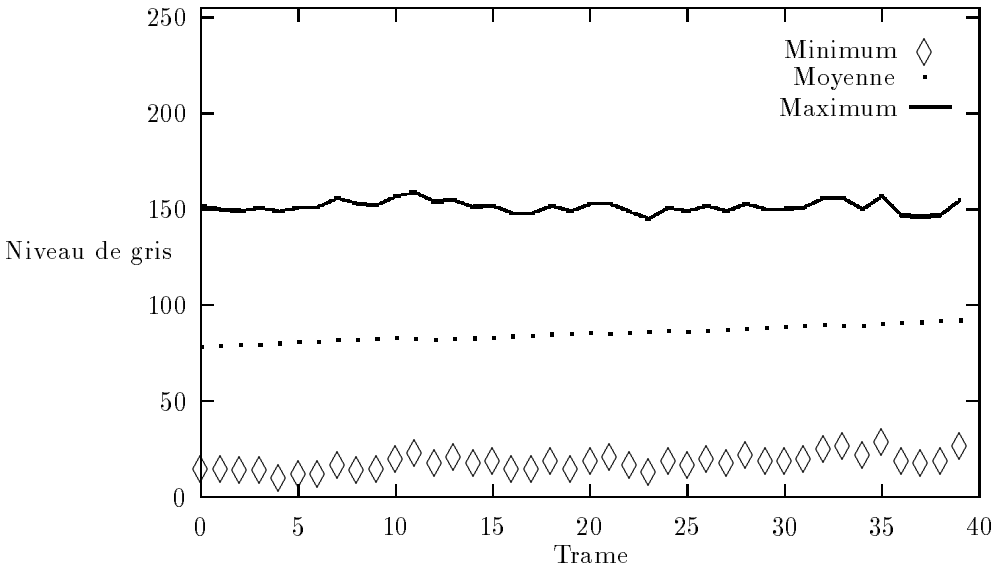


FIG. 3.12 - *Minimum, moyenne et maximum des valeurs de gris pour la séquence "Train" restaurée avec la méthode mixte.*

3.5 Conclusion: limites de la méthode et développements futurs

La méthode de correction du pompage que nous avons développée est basée sur deux critères : recalage des moyennes des niveaux de gris des différentes trames de la séquence et conservation de la dynamique des images. Elle est simple, extrêmement rapide, et donne des résultats satisfaisants tant que le pompage n'est pas trop sévère. Par ailleurs, grâce à la dérive de la moyenne et de la dynamique que nous autorisons, les changements pouvant survenir dans la scène (tels que l'apparition de nouveaux objets) sont pris en compte.

Dans les cas où il a perte d'information par saturation de l'image cette méthode s'avère insuffisante. Dans ce cas un outil de reconstruction des niveaux de gris manquants est indispensable. C'est une voie possible pour des développement futurs.

L'effet de pompage sur certaines séquences n'est pas régulier sur toute l'image. Il arrive qu'il soit plus sévère près des bords. Une méthode de correction avec des paramètres de correction α_t et β_t dépendant de la coordonnée spatiale serait mieux adaptée pour ces cas là.

Chapitre 4

Correction des vibrations

4.1 Introduction

Dans ce chapitre nous étudions le problème des vibrations parasites qui apparaissent dans les films. Celles-ci sont introduites essentiellement lors du transfert du film sur un autre support, digital ou vidéo. Elles sont provoquées donc par un défaut mécanique ou de réglage dans le scanner ou dans l'appareil de transfert vers la vidéo (appelé *téléciné*).

Les méthodes de correction des vibrations présentées sont simples, cependant elles fonctionnent correctement et elles facilitent le travail des algorithmes de restauration suivants.

Nous n'avons trouvé que deux références dans la littérature traitant de ce type de problème. Dans [88] T. Vlachos et G. Thomas traitent le problème du non-alignement des trames entrelacées dans une séquence vidéo produite par un téléciné à lentilles jumelles ("twin-lens telecine"). Le modèle de déplacement utilisé est une translation ajoutée à des légers effets de zoom et de rotation. Nous ne traiterons pas ce cas particulier de défaut. Dans [87] T. Vlachos propose une méthode pour corriger les translations parasites: pour un couple d'images consécutives, il partitionne la première image en blocs réguliers pour lesquels il calcule un vecteur de mouvement. Ensuite par une procédure de vote, qui utilise l'histogramme des vecteurs calculés, il en déduit le mouvement de translation global entre les deux images. On ne voit pas clairement comment il s'y prend pour distinguer entre le mouvement naturel de la séquence et les déplacements parasites.

Notre méthode est similaire sous plusieurs aspects à la méthode présentée dans [87]. En premier lieu nous adopterons aussi un modèle de déplacement translationnel pour les vibrations. Deuxièmement nous estimerons aussi la translation globale entre trames consécutives, mais avec une méthode beaucoup plus simple. Par contre nous accorderons beaucoup d'importance à la distinction entre le mouvement naturel de la séquence et les translations parasites que nous voulons corriger.

Dans la section qui suit nous expliquerons notre modèle d'estimation du mouvement de translation global de la séquence et nous l'appliquerons à une séquence dont le mouvement est faible. Dans la section suivante nous montrerons comment faire pour filtrer les vibrations et ne garder que le mouvement original.

4.2 Estimation d'une translation globale entre deux images d'une séquence

4.2.1 Méthode

Soient $I(t-1)$ et $I(t)$ deux trames d'une séquence. Soit $\vec{\tau}_{u,v}$ le vecteur de coordonnées (u,v). Nous notons $I(t) \oplus \vec{\tau}_{u,v}$, le résultat de l'application de la translation de vecteur $\vec{\tau}_{u,v}$ à la trame $I(t)$. Etant donné que le support $\mathcal{D}_t = \{0, \ldots X-1\} \times \{0, \ldots Y-1\}$ de chaque trame est borné, cette définition pose des problèmes. Nous plongerons donc $\mathcal{D}_t$ dans $\mathbb{R}^2$ et nous associerons une valeur quelconque aux points se trouvant à l'extérieur du support.

Nous supposons que le fond de la scène subit un mouvement de translation global $\vec{\tau}_{u_0,v_0}$. Pour l'instant nous ne nous occupons pas de savoir si cette translation est normale ou si elle est due à des problèmes de vibration. Nous voulons l'estimer en sachant qu'éventuellement la scène filmée peut comporter d'autres objets dont le mouvement est quelconque.

Nous supposons que:

$$\begin{cases} -U \leq u_0 \leq U \\ -V \leq v_0 \leq V \end{cases} \tag{4.1}$$

où U et V sont des entiers naturels.

Nous prenons comme estimation de $\vec{\tau}_{u_0,v_0}$ la valeur $\vec{\tau}^* = \vec{\tau}_{u^*,v^*}$ de $\vec{\tau}$ qui minimise la fonction:

$$D(I(t-1) \oplus \vec{\tau}_{u,v}, I(t)) \tag{4.2}$$

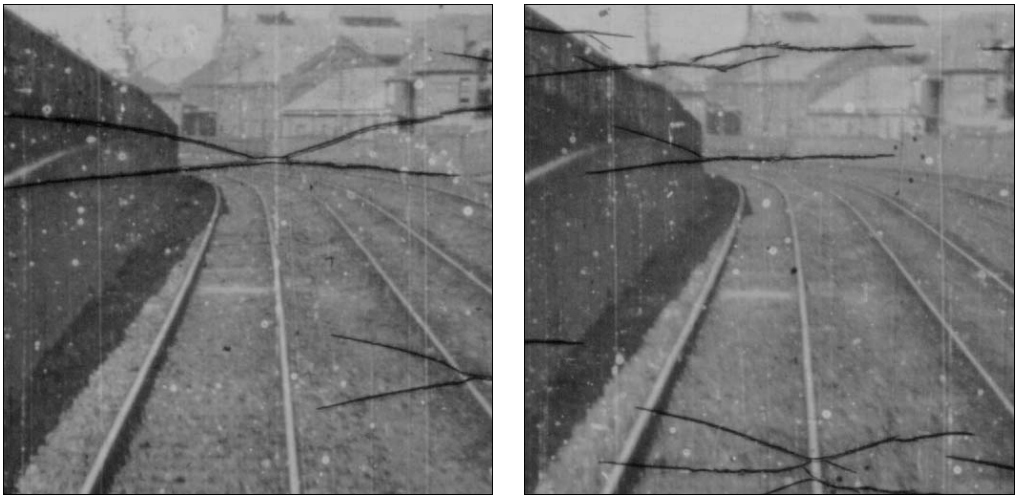
lorsque (u,v) balayent l'espace de définition $[-U,U] \times [-V,V]$. La fonction D est une mesure de la différence entre deux trames. Dans notre cas nous avons choisi la différence absolue pixel à pixel, qui pour cette application donne de bons résultats. Afin d'éviter les problèmes de bords, nous ne considérerons dans le calcul de cette diffèrence que les points appartenant au sous-ensemble de $\mathcal{D}_t$: $\{U, \ldots X-U-1\} \times \{V, \ldots Y-V-1\}$. Si la fonction présente plusieurs minima globaux, nous prendrons alors celui qui correspond au vecteur le plus petit.

Voyons quelques exemples.

4.2.2 Exemples

Dans la séquence "Train" le mouvement de la scène est relativement simple: il s'agit d'un léger zoom avant dû au déplacement de la caméra. En plus de ce mouvement, la séquence présente des translations parasites provoquées par des problèmes de digitalisation.

Soient deux images de cette séquence (voir figure 4.1). Nous avons calculé les différentes valeurs de $D(I(t-1) \oplus \vec{\tau}, I(t))$ pour $U = V = 10$ (c'est à dire pour $-10 \leq u,v \leq 10$) que nous avons représentées sous forme d'une image de dimensions 21×21 (figure 4.2). Remarquons que nous obtenons un minimum global bien marqué, qui correspond à une translation de vecteur $(2,0)$.



$I(t-1)$ $I(t)$

FIG. 4.1 - *Deux images consécutives de la séquence "Train"*

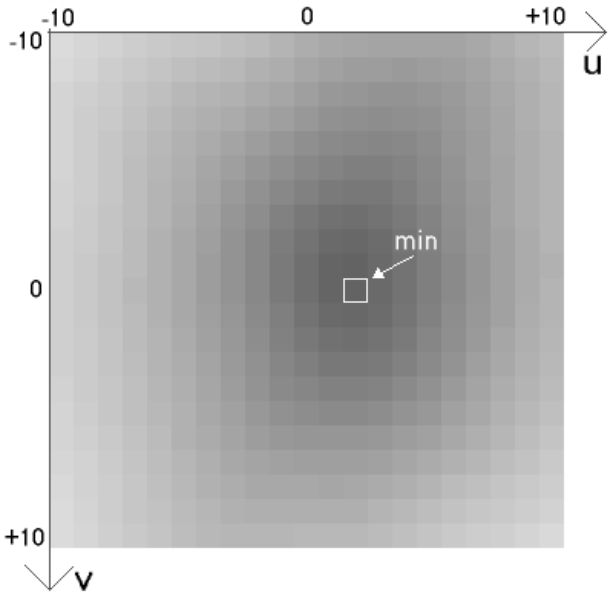


FIG. 4.2 - *Carte de la mesure de la différence entre* $I(t-1) \oplus \vec{\tau}_{u,v}$ *et* $I(t)$ *en fonction de* u *et de* v*. Le niveau de gris est proportionnel à cette mesure.*

Regardons maintenant un cas plus complexe, où les images comportent des objets en mouvement. Nous avons choisi deux images de la séquence "Coastguard". Cette fois nous n'avons pas choisi des images consécutives, mais plutôt des images séparées par 4 trames: $I(t-5)$ et $I(t)$ (figure 4.3). Nous pouvons constater que le fond de la scène subit une translation horizontale, alors que le bateau au centre est suivi par la caméra et par conséquent a un mouvement relatif nul. Comme précédemment nous avons calculé les différentes valeurs de $D(I(t-5) \oplus \vec{\tau}, I(t))$ pour $U = V = 10$, que nous avons représentées dans la figure 4.4. Cette fois-ci nous avons deux minima locaux, que nous avons identifiés par les lettres A et B. Le minimum B donne une translation de vecteur $(0,0)$, ce qui correspond au bateau, et le minimum A correspond au mouvement du fond, qui est de $(-5,0)$. Nous voyons donc apparaître clairement les deux mouvements principaux de la séquence. Ce qui est agréable c'est que le minimum le plus profond, c'est à dire le minimum global, est A: le déplacement du fond l'emporte sur le mouvement du bateau, et ceci malgré le fait que le bateau est bien contrasté, riche en détails et de taille importante.

$I(t-5)$ $I(t)$

FIG. 4.3 - *Deux images de la séquence "Coastguard"*

En conclusion cette méthode nous permet de mesurer de façon assez robuste la translation principale entre deux images. Dans des cas extrêmes, par exemple si nous nous retrouvons en présence d'une rotation d'angle important ou d'un zoom très fort, voire d'un mouvement plus complexe, cette démarche risque d'échouer. Mais en pratique, étant donnée la fréquence d'échantillonnage temporel des films et de la vidéo, notre méthode suffit.

La translation globale que nous venons de calculer peut comporter deux composantes: une naturelle due au mouvement de la scène, et l'autre introduite par les translations parasites que nous cherchons à détecter. Nous allons voir maintenant comment les séparer.

4.3 Calcul des translations parasites

Soit I la séquence originale, avant l'apparition des vibrations. Elle nous est inconnue et nous voulons la retrouver. Nous supposons que chaque image subit un mouvement de

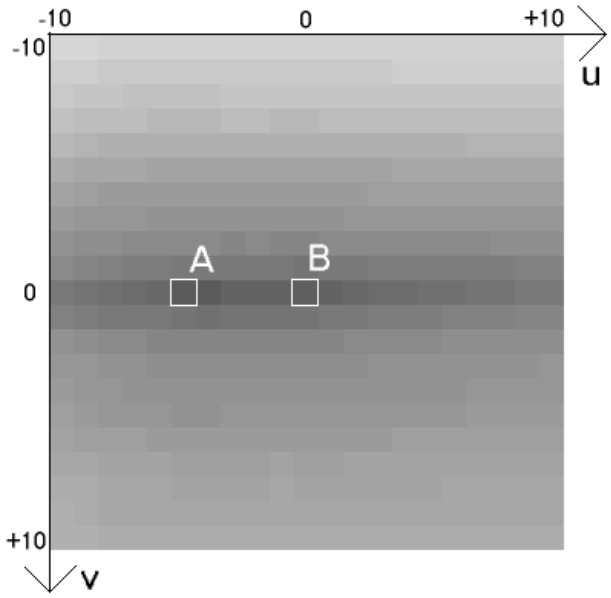


FIG. 4.4 - *Carte de la mesure de la différence entre $I(t-5) \oplus \vec{\tau}_{u,v}$ et $I(t)$ en fonction de u et de v. La valeur de gris est proportionnelle à cette mesure.*

translation global donné par le vecteur $\vec{m}(t)$:

$$I(t-1) \oplus \vec{m}(t-1) = I(t) \tag{4.3}$$

Soit I' la séquence perturbée par les vibrations. C'est notre séquence d'entrée, elle est donc connue. Par contre nous ne connaissons pas les vibrations parasites $\vec{v}(t)$. Elles sont telles que:

$$I'(t) = I(t) \oplus \vec{v}(t) \tag{4.4}$$

La méthode présentée dans la section précédente nous permet de calculer la translation globale $\vec{\tau}(t)$ entre deux trames de I':

$$I'(t-1) \oplus \vec{\tau}(t-1) = I'(t) \tag{4.5}$$

En remplaçant $I'(t-1)$ et $I'(t)$ par l'expression donnée par l'équation 4.4 nous obtenons:

$$(I(t-1) \oplus \vec{v}(t-1)) \oplus \vec{\tau}(t-1) = I(t) \oplus \vec{v}(t) \tag{4.6}$$

Donc, d'après l'équation 4.3:

$$(I(t-1) \oplus \vec{v}(t-1)) \oplus \vec{\tau}(t-1) = (I(t-1) \oplus \vec{m}(t-1)) \oplus \vec{v}(t) \tag{4.7}$$

D'où finalement:

$$\vec{\tau}(t-1) = \vec{v}(t) - \vec{v}(t-1) + \vec{m}(t-1) \tag{4.8}$$

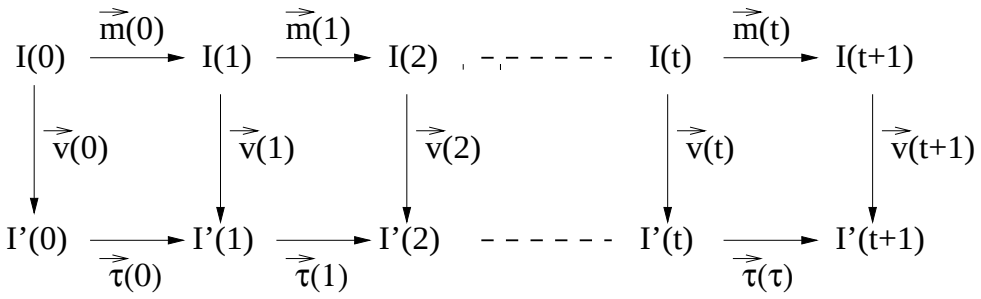


FIG. 4.5 - *Illustration de la relation entre les vecteurs $\vec{m}$, $\vec{\tau}$ et $\vec{v}$.*

Cette équation est illustrée par la figure 4.5.

L'équation 4.8 nous donne une relation dans laquelle nous ne connaissons qu'une estimation $\vec{\tau}^*$ de $\vec{\tau}$, obtenue grâce à la méthode présentée dans la section précédente. Les vecteurs $\vec{v}$ et $\vec{m}$ sont inconnus. Il nous faut des hypothèses supplémentaires pour pouvoir résoudre le problème.

On rencontre ce genre de problèmes en automatique dans les applications de filtrage de trajectoires. L'équation 4.8 peut s'écrire de la façon suivante:

$$\vec{\tau}(t) \quad = \quad \vec{m}(t) + \vec{v}(t+1) - \vec{v}(t) \tag{4.9}$$

La différence $\vec{w}(t) = \vec{v}(t+1) - \vec{v}(t)$ peut alors être interprétée comme un bruit de lecture.

Si par ailleurs nous supposons que le déplacement du fond est constant au cours du temps nous avons en plus:

$$\vec{m}(t+1) = \vec{m}(t) \tag{4.10}$$

Le système résultant correspond à un problème de filtrage discret linéaire. La méthode de choix pour estimer le vrai déplacement est le filtrage de Kalman [37].

Cependant, pour pouvoir appliquer cette méthode il faut que le bruit $\vec{w}(t)$ soit blanc et qu'il soit correctement modélisé par une loi normale de moyenne nulle. Dans notre cas $\vec{w}(t)$ et $\vec{w}(t+1)$ sont correlés, donc nous ne pouvons pas accepter la première hypothèse; et nous ne connaissons pas la loi de distribution du bruit, donc la deuxième hypothèse pose aussi des problèmes. De plus, l'équation 4.10 est un peu restrictive.

Nous avons donc préféré développer une méthode mieux adapté à notre cas. Nous essaierons de limiter le nombre d'hypothèses pour que notre algorithme reste aussi général que possible. En contrepartie, nous ne disposerons d'aucune preuve d'optimalité de la solution que nous proposerons.

La seule hypothèse qui nous sera indispensable dans la suite sera que la moyenne des vibrations $\vec{v}(t)$ au cours du temps soit nulle.

Dans la suite nous appelons $\vec{\tau}$, $\vec{m}$ et $\vec{v}$ les vecteurs réels, qui nous sont inconnus, et $\vec{\tau}^*$, $\vec{m}^*$ et $\vec{v}^*$ l'estimation de ces vecteurs.

4.3.1 Fond fixe

Dans de nombreuses scènes le fond est fixe, et par conséquent on peut considérer que $\vec{m}(t) = \vec{0}$. L'équation 4.8 devient alors:

$$\vec{\tau}(t-1) = \vec{v}(t) - \vec{v}(t-1) \tag{4.11}$$

En intégrant nous obtenons donc $\vec{v}(t)$ pour tout temps t, à une constante près:

$$\begin{aligned} \vec{v}(t) &= \vec{v}(0) + \vec{\tau}(0) + \vec{\tau}(1) + \ldots + \vec{\tau}(t-1) \\ \vec{v}(t) &= \vec{v}(0) + \sum_{i=0}^{t-1} \vec{\tau}(i) \end{aligned} \tag{4.12}$$

Comment faire pour calculer $\vec{v}(0)$?

On peut émettre l'hypothèse que la première trame n'a pas subi de vibration et donc prendre $\vec{v}(0) = \vec{0}$. Si l'hypothèse est incorrecte, toute la séquence se trouvera décalée par la translation de vecteur $\vec{v}(0)$, ce qui en général n'est pas très grave.

Ceci dit, on peut faire mieux avec une hypothèse plus vraisemblable: celle qui dit qu'en moyenne les vibrations sont nulles. Reprenons l'expression des n premières vibrations:

$$\begin{aligned} \vec{v}(0) &= \vec{v}(0) \\ \vec{v}(1) &= \vec{v}(0) + \vec{\tau}(0) \\ \vec{v}(2) &= \vec{v}(0) + \vec{\tau}(0) + \vec{\tau}(1) \\ \ldots &= \ldots \\ \vec{v}(n-1) &= \vec{v}(0) + \vec{\tau}(0) + \ldots + \vec{\tau}(n-2) \end{aligned}$$

D'où:

$$\begin{aligned} \vec{v}(0) + \vec{v}(1) + \ldots + \vec{v}(n-1) &= n\vec{v}(0) + (n-1)\vec{\tau}(0) + \\ &\quad (n-2)\vec{\tau}(1) + \ldots + \vec{\tau}(n-2) \end{aligned} \tag{4.13}$$

En divisant cette équation par n nous obtenons la moyenne des n premières vibrations, qui par hypothèse (à condition de prendre n suffisamment grand) est nulle:

$$\vec{0} = \frac{n\vec{v}(0) + (n-1)\vec{\tau}(0) + (n-2)\vec{\tau}(1) + \ldots + \vec{\tau}(n-2)}{n}$$

Finalement:

$$\begin{aligned} \vec{v}(0) &= -\frac{(n-1)\vec{\tau}(0) + (n-2)\vec{\tau}(1) + \ldots + \vec{\tau}(n-2)}{n} \\ \vec{v}(0) &= -\frac{1}{n}\sum_{j=1}^{n-1} j\,\vec{\tau}(n-1-j) \end{aligned} \tag{4.14}$$

En injectant $\vec{v}(0)$ dans l'équation 4.12 nous obtenons l'expression finale de $\vec{v}(t)$:

$$\vec{v}(t) = -\frac{1}{n}\sum_{j=1}^{n-1} j\vec{\tau}(n-1-j) + \sum_{i=0}^{t-1}\vec{\tau}(i) \tag{4.15}$$

Or nous savons calculer une estimation des vecteurs $\vec{\tau}$, notée $\vec{\tau}^*$, par conséquent nous pouvons donner l'expression de l'estimation de $\vec{v}^*(t)$:

$$\vec{v}^*(t) = -\frac{1}{n}\sum_{j=1}^{n-1} j\vec{\tau}^*(n-1-j) + \sum_{i=0}^{t-1}\vec{\tau}^*(i) \tag{4.16}$$

Ce raffinement, le calcul de $\vec{v}(0)$, peut être considéré dans de nombreux cas comme superflu. On pourra alors se limiter à prendre, comme nous l'avons déjà dit, $\vec{v}(0) = \vec{0}$.

4.3.2 Mouvement constant

Dans la section précédente nous avons résolu l'équation 4.8 en supposant que le mouvement réel du fond était nul. Cette hypothèse est très souvent vraie, mais pas toujours. Nous l'assouplissons dans cette section. Nous supposons que la séquence subit un mouvement global de translation constant au cours du temps:

$$\forall t \quad \vec{m}(t) = \vec{m}(0) \tag{4.17}$$

Désormais dans cette section nous notons $\vec{m} = \vec{m}(t)$

En plus des inconnues que représentent les vibrations, nous en avons une autre: le mouvement $\vec{m}$. L'équation 4.8 devient:

$$\vec{\tau}(t-1) = \vec{v}(t) - \vec{v}(t-1) + \vec{m} \tag{4.18}$$

Donc maintenant nous obtenons:

$$\begin{aligned}
\vec{v}(t) &= \vec{v}(0) + (\vec{\tau}(0) - \vec{m}) + (\vec{\tau}(1) - \vec{m}) + \ldots + (\vec{\tau}(t-1) - \vec{m}) \\
\vec{v}(t) &= \vec{v}(0) - t\vec{m} + \sum_{i=0}^{t-1}\vec{\tau}(i)
\end{aligned} \tag{4.19}$$

Additionnons les n premiers $\vec{v}(t)$:

$$\begin{aligned}
\sum_{j=0}^{n-1}\vec{v}(j) &= n\vec{v}(0) - \sum_{j=0}^{n-1} j\vec{m} + (n-1)\vec{\tau}(0) + (n-2)\vec{\tau}(1) + \ldots + \vec{\tau}(n-2) \\
\sum_{j=0}^{n-1}\vec{v}(j) &= n\vec{v}(0) - \frac{n(n-1)}{2}\vec{m} + \sum_{j=1}^{n-1} j\vec{\tau}(n-1-j)
\end{aligned} \tag{4.20}$$

Afin de calculer $\vec{m}$ nous supposons que la moyenne des vibrations est nulle et que $\vec{v}(0) = \vec{0}$ (dans la section précédente nous avions utilisé soit l'une de ces hypothèses,

soit l'autre; maintenant nous avons besoin des deux en même temps). Par conséquent, à condition que n soit suffisamment grand, nous avons:

$$\vec{m} = \frac{2}{n(n-1)} \sum_{j=1}^{n-1} j\vec{\tau}(n-1-j) \tag{4.21}$$

Donc les équations 4.19 et 4.21, ajoutées au fait que $\vec{v}(0) = \vec{0}$, nous permettent d'écrire:

$$\vec{v}(t) = \sum_{i=0}^{t-1} \vec{\tau}(i) - \frac{2t}{n(n-1)} \sum_{j=1}^{n-1} j\vec{\tau}(n-1-j) \tag{4.22}$$

Par conséquent l'estimation de $\vec{v}(t)$ est:

$$\vec{v}^*(t) = \sum_{i=0}^{t-1} \vec{\tau}^*(i) - \frac{2t}{n(n-1)} \sum_{j=1}^{n-1} j\vec{\tau}^*(n-1-j) \tag{4.23}$$

Nous avons donc réussi à séparer la composante constante du mouvement global de la séquence de la composante à moyenne nulle, correspondant aux vibrations parasites. Nous avons aussi supposé que $\vec{v}(0)$ est nul. Si cette hypothèse ne se vérifie pas toute la séquence se trouvera décalée par la translation de vecteur $\vec{v}(0)$, ce qui en général ne se remarque pas. Malheureusement l'autre hypothèse ($\vec{m}(t)$ constant au cours du temps) est plus dangereuse. Si jamais elle n'est pas vraie et si le vecteur $\vec{m}(t)$ n'est pas tout à fait constant au cours du temps, une dérive importante de la séquence peut apparaître. On pourrait pallier ce risque en recalculant $\vec{m}$ régulièrement. Dans la section suivante nous allons plus loin: nous calculons $\vec{m}(t)$ pour chaque temps t en supposant que le vecteur varie lentement au cours du temps.

4.3.3 Accélération faible

Dans la section précédente nous avons supposé que le mouvement global de la séquence était une translation constante au cours du temps. Maintenant nous supposons que cette translation varie lentement au cours du temps, c'est à dire que l'accélération est faible.

L'équation 4.8 nous donne:

$$\vec{v}(t) = \vec{\tau}(t-1) + \vec{v}(t-1) - \vec{m}(t-1) \tag{4.24}$$

Cette équation, à condition de connaître $\vec{v}(0)$ et $\vec{m}(t)$ pour tout temps t, nous permet de calculer récursivement $\vec{v}(t)$. Nous supposons que $\vec{v}(0)$ est nul. Il nous reste donc à calculer $\vec{m}(t)$. Soient n_1 et n_2 deux entiers naturels. L'équation 4.8 nous permet d'écrire:

$$\vec{m}(t-n_1)+\ldots+\vec{m}(t)+\ldots+\vec{m}(t+n_2)+\vec{v}(t+n_2+1) = \vec{v}(t-n_1)+\vec{\tau}(t-n_1)+\ldots+\vec{\tau}(t)+\ldots+\vec{\tau}(t+n_2) \tag{4.25}$$

En divisant par $n_1 + n_2 + 1$ nous obtenons:

$$\frac{\vec{m}(t-n_1) + \ldots + \vec{m}(t) + \ldots + \vec{m}(t+n_2)}{n_1+n_2+1} = \frac{\vec{v}(t-n_1) - \vec{v}(t+n_2+1)}{n_1+n_2+1} + \frac{\vec{\tau}(t-n_1) + \ldots + \vec{\tau}(t) + \ldots + \vec{\tau}(t+n_2)}{n_1+n_2+1} \quad (4.26)$$

Simplifions maintenant cette expression. A gauche nous reconnaissons la moyenne de $\vec{m}$ entre le temps $t - n_1$ et le temps $t + n_2$. Étant donné que nous avons supposé que ce vecteur varie peu au cours du temps, nous remplaçons cette moyenne par $\vec{m}(t)$. Ensuite, à condition de prendre n_1 et n_2 suffisamment grands, nous négligeons le terme $(\vec{v}(t-n_1) - \vec{v}(t+n_2+1))/(n_1+n_2+1)$. Nous obtenons donc:

$$\vec{m}(t) = \frac{\vec{\tau}(t-n_1) + \ldots + \vec{\tau}(t) + \ldots + \vec{\tau}(t+n_2)}{n_1+n_2+1} \quad (4.27)$$

c'est à dire, en considérant les estimations de ces vecteurs:

$$\vec{m}^*(t) = \frac{\vec{\tau}^*(t-n_1) + \ldots + \vec{\tau}^*(t) + \ldots + \vec{\tau}^*(t+n_2)}{n_1+n_2+1} \quad (4.28)$$

Donc, grâce à l'équation 4.8, nous pouvons maintenant estimer récursivement $\vec{v}(1)$, $\vec{v}(2)$, $\vec{v}(3)$...

4.4 Application et résultats

Supposons que nous devons restaurer une séquence d'images qui présente des vibrations parasites. Quelle méthode adopter pour la restaurer? Est-ce qu'on peut la choisir automatiquement? Comment mettre en œuvre les méthodes décrites plus haut?

4.4.1 Choix de la méthode

Nous n'avons pas mis au point de détecteur automatique de la présence de vibrations dans une séquence. Dans l'état actuel des choses, soit un opérateur devra choisir de traiter ou non les vibrations pour une séquence donnée, ou alors lancer l'algorithme sur toutes les séquences, en sachant que théoriquement les séquences qui ne présentent pas de vibrations devraient sortir pratiquement inchangées du filtrage.

La même méthode se pose pour le choix d'une méthode de correction des vibrations. Si on ne veut pas faire ce choix, alors on peut prendre la technique basée sur l'hypothèse d'accélération faible, car ce cas comprend ceux de vitesse constante et de vitesse nulle.

4.4.2 Mise en œuvre pratique

Nous n'avons retenu que la première et la troisième méthode pour une mise en œuvre effective. La deuxième, basée sur l'hypothèse d'une vitesse constante, nous paraît moins sûre.

Comme précédemment nous notons I et I' respectivement la séquence d'images originale, sans vibrations, et la séquence d'images avec les vibrations, que nous connaissons. Nous appelons I^* la séquence d'images obtenue après la correction des vibrations. Idéalement I^* et I devraient être égales, mais en pratique ce n'est pas toujours le cas.

Voici l'algorithme de mise en œuvre de la méthode pour les séquences à fond fixe:

1. Calcul de $\vec{v}^*(0)$. Soit on choisit arbitrairement $\vec{v}^*(0) = \vec{0}$, soit on calcule les $n-2$ premiers vecteurs $\vec{\tau}^*$ et on utilise la formule 4.14.
2. Si $\vec{v}^*(0)$ n'est pas égal à $\vec{0}$ alors $I^*(0) = I(0) - \vec{v}^*(0)$.
3. Pour t allant de 1 à la dernière trame de la séquence on fait:
   - Calcul de $\vec{\tau}^*(t-1)$. On en déduit $\vec{v}^*(t)$ (équation 4.11). Le résultat en pratique est rarement entier, donc nous prenons l'entier le plus proche.
   - $I^*(t) = I(t) - \vec{v}^*(t)$

L'implémentation de la méthode basée sur l'hypothèse d'accélération faible est plus délicate. Le problème principal provient du choix de n_1 et de n_2. Les simplifications que nous avons opérées pour obtenir l'équation 4.28 ne sont valables que si la somme $n_1 + n_2$ est grande, mais alors l'hypothèse de "faible variation" de la vitesse est plus forte. Sinon la procédure de calcul de $\vec{v}^*(t)$ est similaire à la précédente:

1. Initialisation:
   - $t = 0$
   - Calcul de $\vec{\tau}^*(0)$, $\vec{\tau}^*(1)$, ... $\vec{\tau}^*(n_2)$
2. Pour t allant de 1 à la dernière trame de la séquence on fait:
   - Calcul de $\vec{\tau}^*(t+n_2)$ (si $t+n_2$ ne dépasse pas la dernière trame de la séquence!)
   - Calcul de $\vec{m}^*(t)$ (équation 4.28)
   - Calcul de $\vec{v}^*(t)$ (équation 4.24)
   - $I^*(t) = I(t) - \vec{v}^*(t)$

Cette méthode est plus lourde à mettre en œuvre: d'une part il faut garder en mémoire $n_1 + n_2 + 1$ valeurs de $\vec{\tau}^*$ et d'autre part, et c'est l'inconvénient principal, il faut aller lire les trames $I(t+n_2)$ et $I(t+n_2+1)$ pour calculer $\vec{\tau}^*(t+n_2)$.

4.4.3 Résultats

4.4.3.1 Application du premier algorithme

Intéressons nous à la séquence "Train". Dans cette séquence le déplacement global correspond à un léger zoom avant, provoqué par le mouvement du train, sur lequel viennent

se superposer les vibrations. Étant donné que le mouvement global ne comporte pas de composante de translation, et malgré la présence du zoom, nous avons appliqué la méthode destinée aux séquences à fond fixe à cette série d'images. Et en effet, après traitement la séquence ne vibre plus. Ceci est un bon point pour la robustesse de notre algorithme.

4.4.3.2 Application de l'algorithme basé sur l'hypothèse d'une accélération faible

Pour tester le deuxième algorithme nous avons fabriqué une séquence test. Nous avons pris comme point de départ la fin de la séquence "Foreman", où la scène est balayée de gauche à droite avec une vitesse qui varie au cours du temps. Dans la figure 4.6 nous montrons la première, la vingt-et-unième et la quarante-et-unième (et dernière) image de cette séquence. Chaque trame a une dimension de 166×134 pixels. Cette séquence constitue donc notre séquence "avant vibrations" I.

FIG. 4.6 - *Trois images de la séquence utilisée pour tester la méthode de correction des vibrations.*

Nous avons estimé la translation entre trames consécutives de cette séquence grâce à la méthode présentée dans la section 4.2. Dans le tableau 4.1 nous montrons les abscisses des vecteurs de translation correspondants. Remarquons qu'elles varient lentement.

Ensuite nous avons rajoutée des vibrations parasites à cette séquence. Pour chaque trame indépendamment nous avons calculé un vecteur de vibration. Chaque coordonnée a été tirée au hasard, avec une probabilité égale pour chaque valeur, dans l'ensemble $\{-5, \ldots, 0, \ldots 5\}$. Seule exception: la première trame, pour laquelle nous avons pris un vecteur de translation nul. Les abscisses ainsi obtenues apparaissent sur le tableau 4.2. Cette séquence correspond à notre image à restaurer $I'(t)$.

Avant de continuer remarquons que la séquence ainsi produite correspond à un cas difficile pour la restauration de vibrations. En effet, le mouvement naturel de la séquence est important, comme le montre le tableau 4.1, surtout quand on prend en compte les dimensions de l'image, et la sévérité des vibrations. De plus, dans les premières trames apparaît un personnage en mouvement.

Trame t	0	1	2	3	4	5	6	7	8	9
Abscisse de $\vec{m}(t)$	-4	-5	-5	-6	-6	-7	-8	-9	-10	-10
Trame t	10	11	12	13	14	15	16	17	18	19
Abscisse de $\vec{m}(t)$	-10	-9	-8	-7	-7	-6	-5	-5	-5	-5
Trame t	20	21	22	23	24	25	26	27	28	29
Abscisse de $\vec{m}(t)$	-5	-6	-6	-6	-6	-6	-6	-7	-7	-7
Trame t	30	31	32	33	34	35	36	37	38	39
Abscisse de $\vec{m}(t)$	-6	-5	-4	-3	-3	-2	-2	-2	-2	-2

TAB. 4.1 -

Trame t	1	2	3	4	5	6	7	8	9	10
Abscisse de $\vec{v}(t)$	-3	-1	-5	1	-5	-4	3	5	0	5
Trame t	11	12	13	14	15	16	17	18	19	20
Abscisse de $\vec{v}(t)$	-3	2	1	-4	-3	-5	0	-2	1	-3
Trame t	21	22	23	24	25	26	27	28	29	30
Abscisse de $\vec{v}(t)$	3	-3	-5	-4	5	4	-5	-3	-4	-2
Trame t	31	32	33	34	35	36	37	38	39	40
Abscisse de $\vec{v}(t)$	0	2	4	1	0	-2	-3	1	-1	-3

TAB. 4.2 -

Le calcul des vecteurs de translation $\vec{\tau}^*$ à partir des images soumises aux vibrations donne le tableau 4.3.

Trame t	0	1	2	3	4	5	6	7	8	9
Abscisse de $\vec{\tau}^*(t)$	-7	-3	-9	1	-12	-6	-1	-7	-15	-5
Trame t	10	11	12	13	14	15	16	17	18	19
Abscisse de $\vec{\tau}^*(t)$	-18	-4	-9	-12	-6	-8	0	-7	-2	-9
Trame t	20	21	22	23	24	25	26	27	28	29
Abscisse de $\vec{\tau}^*(t)$	1	-12	-8	-5	3	-7	-15	-5	-8	-5
Trame t	30	31	32	33	34	35	36	37	38	39
Abscisse de $\vec{\tau}^*(t)$	-4	-3	-2	-6	-4	-4	-3	2	-4	-4

TAB. 4.3 -

Remarquons que les valeurs des trois tableaux que nous venons de présenter sont théoriquement liées par l'équation 4.8. Et en effet, ces valeurs vérifient notre équation. Ce qui veut dire encore une fois que notre méthode de calcul de la translation entre deux images fonctionne plutôt bien!

Maintenant oublions que nous connaissons les vecteurs de mouvement ainsi que les vecteurs de vibration. Comme dans un cas réel nous nous limitons à la connaissance des vecteurs $\vec{\tau}^*$. En utilisant l'algorithme de recalage pour les cas où l'accélération est faible nous calculons $\vec{v}^*$ et $\vec{m}^*$.

Comment choisir n_1 et n_2? Idéalement il faudrait qu'ils soient aussi grands que possible tout en respectant l'hypothèse de base, *i.e.* que $\vec{m}(t)$ soit égal à la moyenne de $\vec{m}(t - n_1), \ldots, \vec{m}(t + n_2)$. Dans le cas qui nous concerne nous connaissons les vrais vecteurs de mouvement (tableau 4.1). Nous constatons alors qu'avec $n_1 = n_2 = 1$ l'hypothèse est vraie pour les 40 vecteurs, pour $n_1 = n_2 = 2$ elle est vraie pour 38 vecteurs, et pour $n_1 = n_2 = 3$ elle est vraie pour 29 vecteurs. Nous retiendrons donc $n_1 = n_2 = 2$. Mais dans les problèmes réels nous ne connaissons pas les vrais vecteurs de translation globale, nous cherchons à les calculer. Ceci dit, étant donné que l'exemple que nous avons choisi correspond à un cas assez sévère, où la vitesse varie de façon importante, nous retiendrons ces paramètres pour les autres cas.

L'application de la formule 4.28 nous donne les valeurs de $\vec{m}^*$ à partir de $\vec{\tau}^*$ (voire tableau 4.4). Remarquons d'abord que ces valeurs varient doucement au cours du temps. Ceci se traduit par un défilement agréable de la séquence. Nous avons indiqué sur le même tableau la différence entre le mouvement ainsi calculé et le mouvement "vrai" de la séquence. Cette différence est faible.

Nous pouvons maintenant calculer récursivement les valeurs des vecteurs de vibration en utilisant l'équation 4.8. Le résultat, ainsi que la différence avec les valeurs réelles, apparaissent sur le tableau 4.5.

Le résultat du calcul de vibrations est de prime abord un peu décevant: la moyenne de l'erreur absolue, pour l'abscisse, est de $1,5$. Cependant, comme nous l'avons dit au

Trame t	0	1	2	3	4	5	6	7	8	9
Abscisse de $\vec{m}^*(t)$	-6	-4	-6	-6	-5	-5	-8	-7	-9	-10
Différence absolue avec la vraie valeur	2	1	1	0	1	2	0	2	1	0
Trame t	10	11	12	13	14	15	16	17	18	19
Abscisse de $\vec{m}^*(t)$	-10	-10	-10	-8	-7	-7	-5	-5	-3	-6
Différence absolue avec la vraie valeur	0	1	2	1	0	1	0	0	2	1
Trame t	20	21	22	23	24	25	26	27	28	29
Abscisse de $\vec{m}^*(t)$	-6	-7	-4	-6	-6	-6	-6	-8	-7	-5
Différence absolue avec la vraie valeur	1	1	2	0	0	0	0	1	0	2
Trame t	30	31	32	33	34	35	36	37	38	39
Abscisse de $\vec{m}^*(t)$	-4	-4	-4	-4	-4	-3	-3	-3	-2	-2
Différence absolue avec la vraie valeur	2	1	0	1	1	1	1	1	0	0

TAB. 4.4 -

Trame t	1	2	3	4	5	6	7	8	9	10
Abscisse de $\vec{v}^*(t)$	-1	-0	-3	4	-3	-4	3	3	-3	2
Différence absolue avec la vraie valeur	2	1	2	3	2	0	0	2	3	3
Trame t	11	12	13	14	15	16	17	18	19	20
Abscisse de $\vec{v}^*(t)$	-6	0	1	-3	-2	-3	2	0	1	-2
Différence absolue avec la vraie valeur	3	2	0	1	1	2	2	2	0	1
Trame t	21	22	23	24	25	26	27	28	29	30
Abscisse de $\vec{v}^*(t)$	5	0	-4	-3	6	5	-4	-1	-2	-2
Différence absolue avec la vraie valeur	2	3	1	1	1	1	1	2	2	0
Trame t	31	32	33	34	35	36	37	38	39	40
Abscisse de $\vec{v}^*(t)$	-2	-1	1	-1	-1	-2	-2	3	1	-1
Différence absolue avec la vraie valeur	2	3	3	2	1	0	1	2	2	2

TAB. 4.5 -

début, nous avons choisi un cas délicat. Par ailleurs, lorsque nous visualisons la séquence en mouvement l'amélioration est nette.

4.5 Conclusion

Dans ce chapitre nous avons présenté une méthode de calcul d'une translation globale entre deux trames qui est simple et efficace. Nous avons ensuite décrit deux algorithmes qui utilisent cette méthode pour estimer les vibrations parasites dans des séquences d'images. Le premier algorithme est destiné aux cas où le mouvement global de la scène est nul, et le deuxième aux cas où le mouvement du fond est une translation qui varie peu au cours du temps.

Ces algorithmes sont rapides. Le temps le plus long correspond à l'estimation de la translation globale; la durée du calcul des vibrations parasites est négligeable par rapport à celui-là. Ces algorithmes sont aussi robustes. Ils donnent aussi de bons résultats lorsque les mouvement de la scène n'est pas purement translationnel ou lorsqu'il y a des objets qui bougent différemment.

Chapitre 5

Restauration de défauts locaux immobiles

5.1 Introduction

Une classe de défauts assez courante dans les anciens films est celle constituée par des rayures verticales blanches ou noires apparaissant au même endroit sur plusieurs trames consécutives. Ces défauts sont produits par le frottement de particules étrangères sur la pellicule. Non seulement ce sont des défauts courants, mais en plus ils sont très gênants, le regard du spectateur étant souvent attiré par ces artéfacts qui peuvent rester présents à l'écran pendant plusieurs secondes.

Ces rayures verticales rentrent dans la catégorie des défauts locaux immobiles, telle que nous l'avons définie dans le chapitre 1. En effet, d'une part ce sont des défauts locaux car ils affectent une région de taille restreinte de chaque trame, et d'autre part, comme nous venons de le dire, ils apparaissent à la même position sur plusieurs images consécutives.

Intuitivement on aurait pu penser que ces défauts sont plus faciles à détecter et à corriger que les défauts locaux aléatoires. En effet, puisqu'ils ne bougent pas, puisqu'ils restent au même endroit plutôt que disparaître et apparaître de façon aléatoire, ils devraient être faciles à traquer et localiser. Mais en pratique ce n'est pas le cas. Vu qu'ils ne bougent pas, il se confondent avec leur environnement. Les analyses basées sur le mouvement dans la scène se révèlent incapables de les détecter. En fait ceci n'est pas tout à fait exact : théoriquement, lorsqu'il y a du mouvement dans la scène, et en particulier quand la caméra bouge, on devrait pouvoir identifier les défauts immobiles parce que justement, eux ils ne bougent pas. Mais en pratique ça serait très difficile.

En contrepartie, ces défauts sont beaucoup moins variés que les défauts locaux aléatoires. Nous n'en avons observé que deux sortes : des rayures noires verticales et des rayures blanches verticales. Donc nous utiliserons leurs caractéristiques géométriques dans la phase de détection.

La bibliographie sur ce sujet est très restreinte. Morris présente dans sa thèse [62] une méthode de restauration de rayures verticales basée sur des techniques Bayesiennes. L'algo-

rithme résultant est apparemment très lourd. Dans [42] Kokaram propose une amélioration: dans une phase de prétraitement il cherche les lignes candidates à rayures; ensuite, par un raffinement Bayesien, il garde les meilleurs candidats. Cette phase de prétraitement présente des analogies avec les méthodes de détection que nous présentons dans ce chapitre. L'interpolation se fait à l'aide d'un filtre auto-régressif 2D. Dans ces deux articles la restauration est divisée en deux phases bien distinctes: la première de détection, la deuxième d'interpolation. Ainsi on ne modifie que les parties de l'image considérées comme étant des défauts. Nous adopterons aussi cette division du processus de restauration.

Dans la suite nous considérerons uniquement les rayures blanches verticales. Les méthodes proposées pour les effacer pourront sans aucune difficulté être appliquées aux rayures noires verticales en inversant l'image.

Nous commençons par présenter deux méthodes de détection et par les comparer. Ensuite nous décrivons rapidement comment faire pour, une fois les rayures localisées, interpoler l'information perdue.

5.2 Détection

Dans cette section nous présentons deux méthodes de détection de rayures verticales. Nous utilisons deux séquences pour effectuer les tests. La première, dont nous pouvons voir une trame sur la figure 5.1(a), montre un cas de rayure qui ne traverse pas toute l'image mais qui autrement ne pose pas de problème particulier. Sur la deuxième séquence (cf. figure 5.1(b)) nous avons bien une rayure, mais elle n'est pas la seule structure verticale de la scène: on a d'autres lignes blanches et étroites, cependant elles sont légèrement inclinées. Dans les deux cas la caméra est fixe et les personnages bougent peu.

FIG. 5.1 - *Deux trames extraites des séquences utilisées pour illustrer les algorithmes de restauration des rayures.*

5.2.1 Première méthode

5.2.1.1 Description et modèle

Comme nous l'avons dit dans l'introduction, les rayures blanches présentent les caractéristiques suivantes:

1. Verticales. De plus elles sont toujours longues et souvent elles traversent toute l'image de haut en bas.
2. Étroites.
3. Plus claires que leur voisinage.
4. Immobiles dans le temps.

Le modèle mathématique reprend exactement les termes de cette description. On dira qu'un parallélépipède dont les arêtes sont parallèles aux axes de $\mathbb{Z}^3$, de hauteur d_y, de largeur d_x et de profondeur d_t est une rayure de la séquence I si et seulement si:

1. $d_x < D_x$,
2. $d_y \geq D_y$,
3. tous les voisins du parallélépipède ont un niveau de gris plus bas que celui-ci et
4. $d_t < D_t$.

Les paramètres D_x, D_y et D_t sont choisis en fonction du format de la séquence et de la puissance de calcul disponible. D_x sera d'autant plus grand que la résolution sera élevée. Dans le cas de nos deux séquences test nous avons pris $D_x = 5$, alors que les trames ont une dimension de 768×576 pixels. Théoriquement on pourrait prendre un D_y élevé, mais c'est risqué en présence de bruit (ce qui est souvent le cas, vue notre application!). Nous avons utilisé $D_y = 22$. De même, l'idéal serait de prendre D_t très grand, mais les temps de calcul peuvent devenir alors prohibitifs. Nous nous sommes limités à $D_t = 3$.

5.2.1.2 Mise en œuvre

La suite d'opérations morphologiques pour identifier les rayures décrites par ce modèle mathématique est très simple:

1. Chapeau haut de forme horizontal de taille D_x. Ceci permet de détecter les structures claires, verticales et étroites de l'image.
2. Ouverture verticale de taille D_y. Cette opération efface toutes les structures qui ne sont pas suffisamment longues suivant la direction (Oy).

3. Ouverture temporelle de taille D_t, suivie d'une reconstruction spatio-temporelle. Nous ne gardons ainsi que les objets qui sont immobiles.

4. Seuillage.

Nous avons appliqué cet algorithme à nos deux séquences test. Les résultats apparaissent sur la figure 5.2.

Remarquons que dans le cas de la première séquence la rayure n'a pas été complètement détectée. Ceci est dû au fait qu'elle n'est pas continue. Lorsqu'on regarde de près on peut constater que la ligne blanche est coupée à certains endroits. Dans le deuxième cas la rayure a été détectée de façon plus ou moins satisfaisante mais on peut observer de nombreuses autres détections, et qui plus est, apparaissant de façon beaucoup plus claire.

Donc dans l'ensemble le résultat n'est pas très satisfaisant. Nous analysons les causes de cet échec relatif dans la section suivante.

5.2.2 Intermède critique

Pourquoi est-ce que la méthode que nous venons de présenter n'est pas aussi satisfaisante que nous l'aurions voulu, alors que l'utilisation du chapeau haut-de-forme horizontal réussit à mettre en évidence les structures verticales de l'image? Que pouvons-nous faire pour l'améliorer?

Cet échec relatif a deux causes. Premièrement, l'utilisation de l'ouverture par un élément structurant vertical n'est pas suffisamment discriminante: une ligne d'une certaine épaisseur qui ne serait pas parfaitement verticale peut contenir des segments verticaux de longueur D_y, et par conséquent passer entre les mailles du filet de notre algorithme de détection. C'est ce qui arrive dans le cas de la deuxième séquence (cf. image (f) de la figure 5.2). Deuxièmement, à cause de sa nature même et de la présence de bruit, la rayure correspond mal au modèle mathématique. Elle n'apparaît pas toujours sous la forme d'une ligne continue; souvent elle est coupée en plusieurs segments plus petits.

Par ailleurs, notre critère temporel n'est pas très utile. Il ne sert que lorsque le mouvement dans la scène est important, ce qui le plus souvent n'est pas vrai. Or l'ouverture par reconstruction est une opération assez coûteuse en temps de calcul, donc nous ne l'utilisons plus dans la section suivante.

Forts de cette expérience, nous développons dans la section suivante un algorithme qui reprend les points forts de l'algorithme précédent, c'est à dire le chapeau haut-de-forme horizontal, tout en utilisant des méthodes différentes pour exploiter le caractère vertical des rayures.

5.2.3 Deuxième méthode

Comme précédemment, nous commençons par appliquer à notre séquence le chapeau haut de forme avec un élément structurant horizontal de taille D_x. Le résultat, pour nos deux trames illustratives, avec $D_x = 5$, apparaît sur la figure 5.3 (a) et (b).

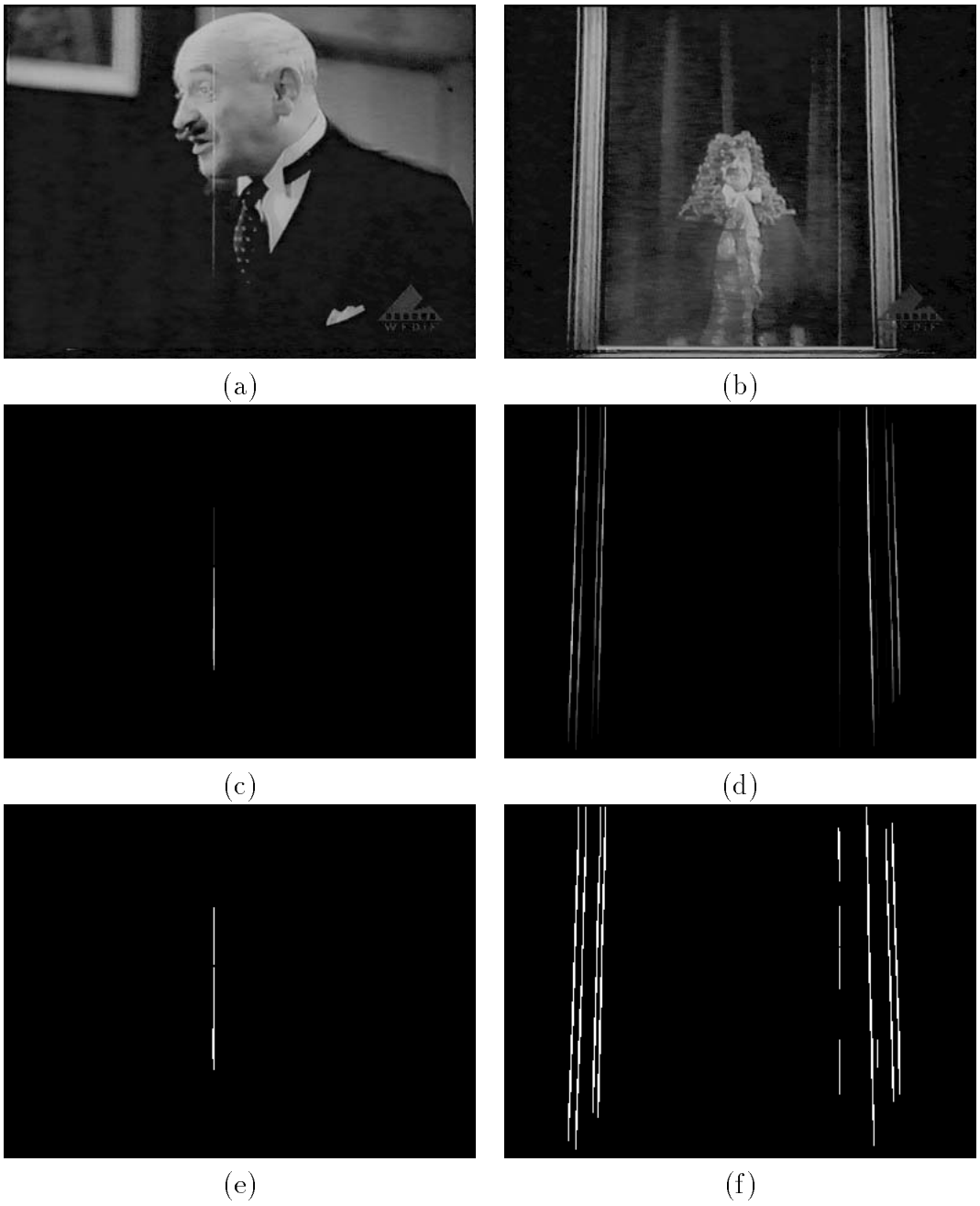


FIG. 5.2 - *Résultat de la détection des rayures avec le premier algorithme. (a) et (b) sont deux trames originales des séquences test. (c) et (d) montrent le résultat de l'application de l'algorithme pour les trames considérées (les histogrammes de ces images sont normalisés pour la visualisation). (e) et (f) correspondent aux images (c) et (d) seuillées respectivement avec des seuils de 6 et 2.*

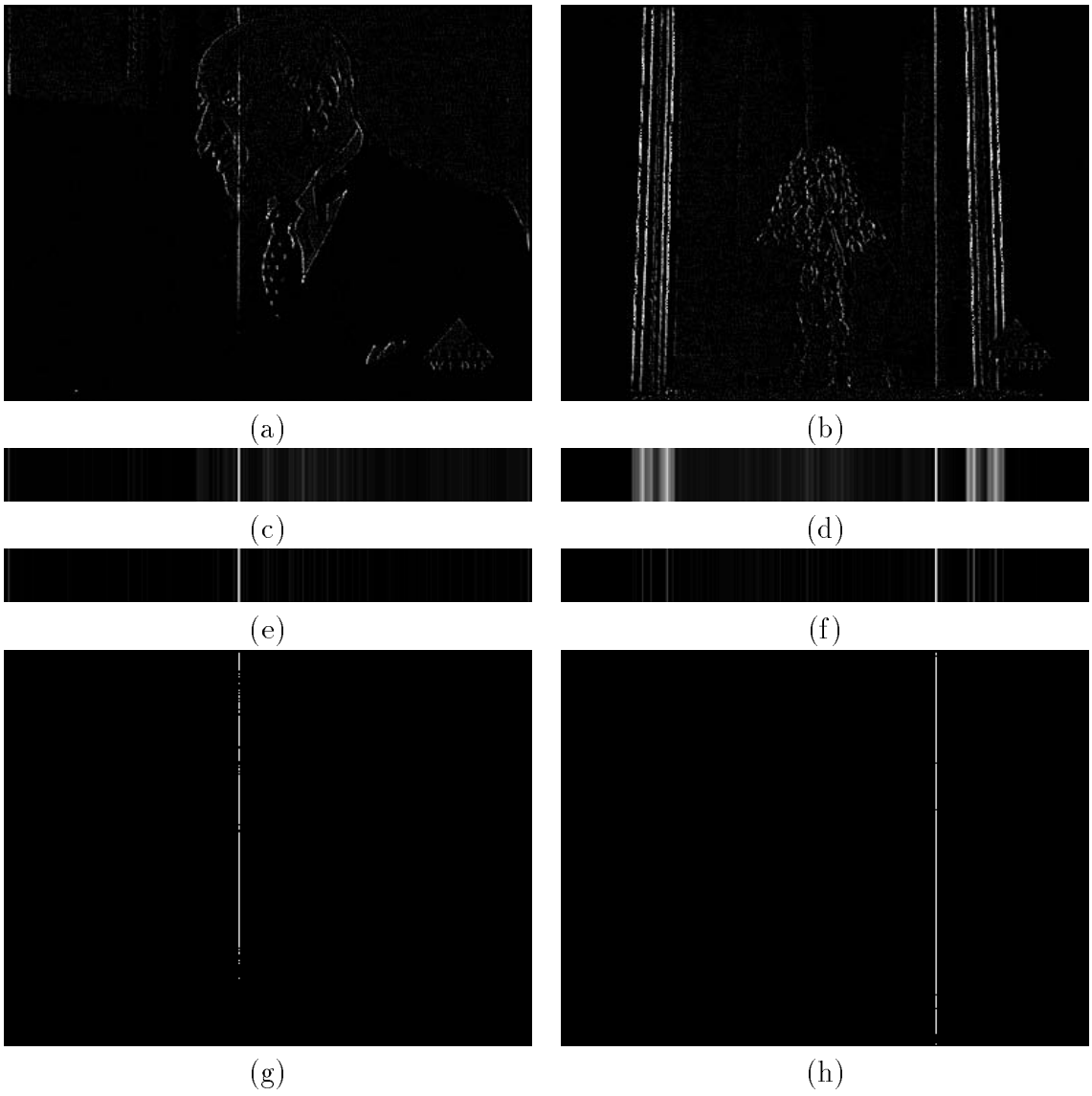


FIG. 5.3 - *(a) et (b) : Résultat de l'application du chapeau haut-de-forme avec un élément structurant horizontal de taille 5 à nos deux images de référence. (c) et (d) : Moyenne des colonnes des images précédentes. (e) et (f) Résultat de l'application du même chapeau haut-de-forme que précédemment aux images (c) et (d) respectivement. (g) et (h) : Colonnes de (a) et (b) retenues, après seuillage. Toutes les images présentées dans cette figure sont normalisées.*

Ensuite, pour interpréter le résultat, nous calculons la moyenne sur chaque colonne de l'image résultante. Mais cette moyenne est entière et notre image d'entrée contient de nombreux pixels de niveau de gris nul, donc l'ensemble des valeurs obtenues risque d'avoir une dynamique assez faible. Afin de réduire ce problème nous multiplions notre image d'entrée par une constante, que nous avons choisie égale à 5. L'image ainsi obtenue, que nous appelons "accumulateur", est réduite à une seule ligne. Nous présentons ceux correspondant aux images (a) et (b) dans la figure 5.3 (images (c) et (d)). Remarquons que les rayures apparaissent clairement, mais qu'éventuellement d'autres structures verticales peuvent aussi y laisser une trace (en particulier voir l'image (d)).

Vu que les rayures sont étroites et parfaitement verticales, l'application du même chapeau haut-de-forme que précédemment à l'accumulateur permet d'enlever une bonne partie des fausses détections (cf. images 5.3 (e) et (f)). Ceci apparaît clairement dans le cas de la deuxième séquence: la trace de la rayure ressort clairement sur l'image (f), alors que sur l'image (d) on voyait clairement aussi les marques laissées par les autres structures verticales de la scène. Le pouvoir de discrimination de notre algorithme est grand. Ceci dit, si la scène avait comporté des lignes blanches parfaitement verticales et étroites, elles auraient été prises pour des rayures blanches.

En seuillant ce résultat avec une valeur g_1, nous obtenons des marqueurs pour les colonnes de la trame qui présentent des rayures. Cependant, certaines rayures ne couvrent pas toute la hauteur de l'image, donc la dernière étape consiste à considérer comme appartenant aux rayures les pixels du premier chapeau haut-de-forme (images (a) et (b)) dont le niveau de gris est supérieur à un certain seuil g_2 et qui appartiennent aux colonnes marquées. Nous obtenons ainsi finalement les rayures qui abîment nos images (figure 5.3 (g) et (h)).

Dans les exemples présentés nous avons utilisé comme seuils $g_1 = 30$ et $g_2 = 4$.

Les étapes principales de cet algorithme sont résumées sur la figure 5.4.

5.3 Récupération de l'information perdue

Nous avons réussi à détecter de façon précise les rayures qui abîment nos séquences d'images. Maintenant nous devons reconstruire l'image pour faire disparaître le défaut.

Dans cette section nous traitons le problème de la récupération d'information dans le cas particulier des rayures. La méthode présentée est donc très spécifique et assez simple. Dans le chapitre 6 nous verrons des méthodes plus générales.

Les images originales de la figure 5.1 montrent que les rayures détruisent l'information qui se trouvait à l'emplacement qu'elles occupent. Nous devons donc utiliser l'information qui se trouve autour du défaut pour interpoler la texture manquante. Quand nous disons "autour" nous nous plaçons dans la structure tridimensionnelle, donc a priori nous pouvons utiliser aussi bien l'information se trouvant sur la trame courante que l'information des trames voisines. Cependant, dans le cas qui nous occupe les défauts sont immobiles, donc la plupart du temps l'information qui a été perdue sur la trame courante a disparu aussi sur les trames voisines. Par conséquent nous nous limitons à une interpolation spatiale.

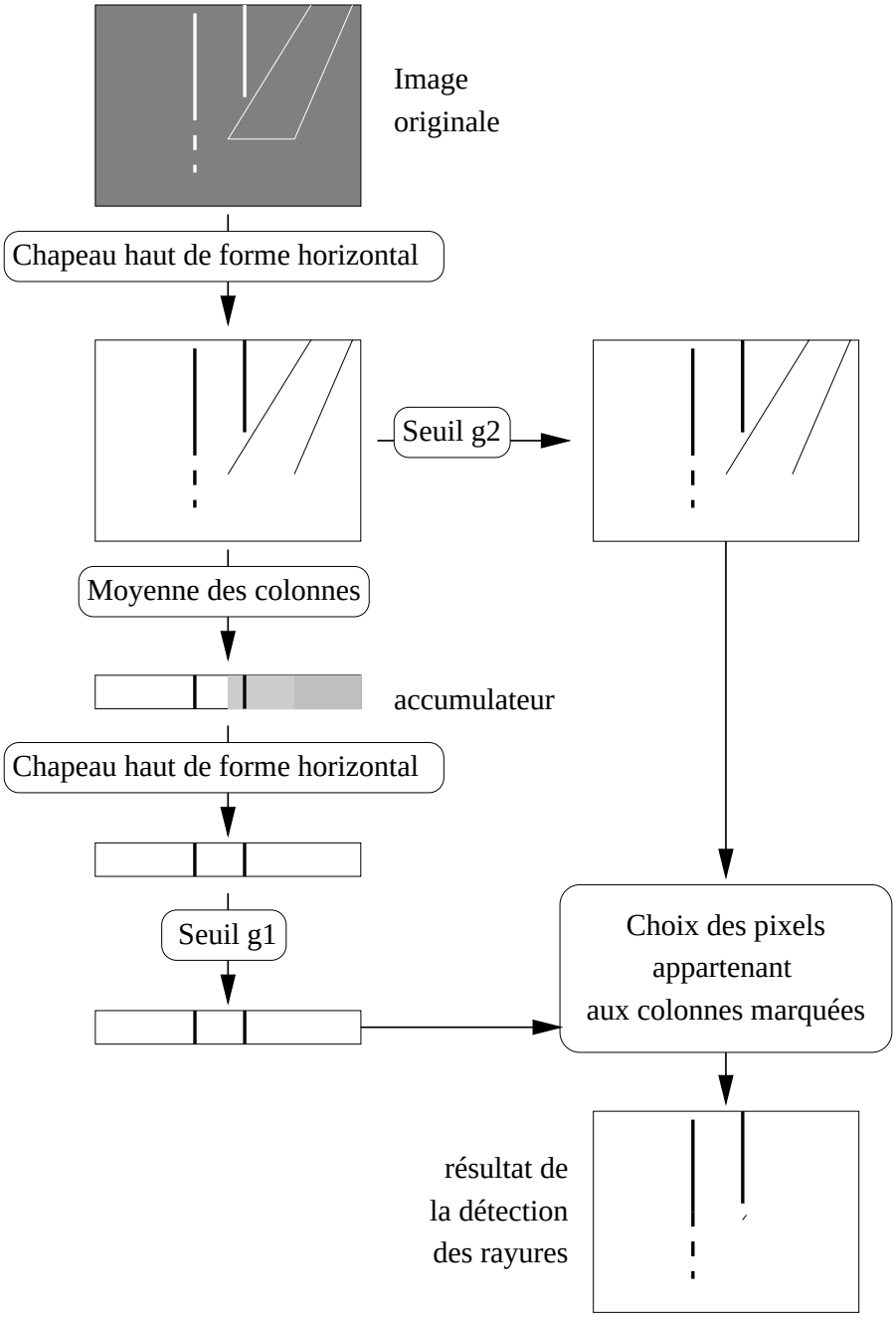


FIG. 5.4 - *Algorithme de détection de rayures*

5.3.1 Prétraitement

Avant de faire l'interpolation proprement dite nous procédons à une phase de prétraitement. En effet, le voisinage autour de la zone détectée comme endommagée est souvent perturbé par le défaut, ce qui peut fausser l'interpolation. Afin de palier ce problème, nous dilatons le marqueur des rayures en utilisant un élément structurant carré 3×3 (voir figure 5.5).

(a) Marqueur (b) Marqueur dilaté

FIG. 5.5 - *Détail du marqueur d'une rayure. (a) Marqueur en blanc, superposé sur l'image originale. (b) Dilatation de ce marqueur.*

5.3.2 Interpolation

Étant donné que les rayures sont étroites nous nous contentons d'une méthode d'interpolation très simple: une ouverture avec le même élément structurant utilisé dans le chapeau haut-de-forme horizontal lors de la phase de détection. Rappelons que le résultat du filtrage est seulement appliqué aux régions qui ont été identifiées comme étant des rayures. Partout ailleurs l'image reste inchangée. L'image de la figure 5.6 montre un exemple d'application de cette méthode.

(a) (b)

FIG. 5.6 - *Détail de la trame originale (a) et interpolation de la région qui avait été détruite par la rayure grâce à une ouverture morphologique avec un élément structurant horizontal (b).*

5.4 Conclusion

Dans ce chapitre nous avons présenté deux méthodes de détection de rayures dans une image. La première ne donne pas de très bons résultats, mais elle nous a inspirés pour construire la deuxième, qui s'avère bien meilleure.

Cependant, si la scène originale comporte des fines lignes blanches verticales (par exemple si une grille peinte en blanc apparaît sur le film) elle seront aussi détectées par notre algorithme. Comment résoudre ce problème? Nous pourrions préciser encore davantage notre modèle de rayure, mais nous tomberons toujours sur des cas de films où nous trouverons des objets qui auront exactement les mêmes caractéristiques que les défauts que nous cherchons à détecter. Une autre solution, plus conservatrice, est de faire contrôler le processus de restauration par un opérateur humain, qui choisira les séquences auxquelles il veut appliquer l'algorithme.

La méthode de récupération d'information que nous proposons pour les rayures est simple et rapide. Elle peut être améliorée, mais nous ne développons pas le sujet davantage dans ce chapitre car nous verrons à la fin du chapitre suivant des méthodes d'interpolation plus puissantes.

Chapitre 6

Restauration des défauts locaux aléatoires

6.1 Introduction

Dans le chapitre précédent nous avons étudié la restauration des défauts locaux immobiles. Nous allons maintenant nous intéresser au deuxième type de défauts locaux: les défauts locaux aléatoires.

Celui qui a vu un vieux film aura sûrement remarqué des taches noires ou blanches, de tailles et formes variées, qui apparaissent et disparaissent fugitivement un peu partout sur l'image, si rapidement qu'on ne les voit que du coin de l'œil. Elles ont des origines diverses: poussière, champignons, éclats de la pellicule... Ces défauts sont très gênants et, malheureusement, très courants. Dans le chapitre 1 nous les avons classés dans la catégorie des **défauts locaux aléatoires**, qui, rappelons-le, sont caractérisés par les propriétés suivantes:

- Ils sont locaux, c'est à dire que chaque défaut affecte une région relativement restreinte de chaque trame.
- La probabilité qu'ils se superposent entre deux trames consécutives est faible.

6.1.1 État de l'art

La réduction de bruit dans les séquences vidéo a été l'une des premières tentatives de restauration de défauts locaux. Dès le début les chercheurs ont développé des algorithmes basés sur une première phase de compensation de mouvement, suivie d'un traitement le long des trajectoires [35, 21]. Les recherches dans ce domaine continuent aujourd'hui en utilisant des méthodes plus modernes et qui restent basées sur une première phase de compensation de mouvement (voir par exemple [64, 39, 83]).

Gonzalo Arce, dans [1], montre que le filtrage spatio-temporel peut être effectué de façon satisfaisante sans utiliser de compensation de mouvement. Pour cela il utilise une

combinaison de filtres médians (*Multistage Order Statistic Filters*). Les résultats sont intéressants mais, étant donnée la taille des supports des filtres médians, en présence de mouvement le filtrage résultant est purement spatial.

Dans [66] Pardàs, Serra et Torres appliquent pour la première fois les ouvertures et fermetures par reconstruction aux séquences d'images obtenant des résultats encourageants. L'application qui les motivait était la réduction de bruit impulsif, mais ils précisent que cette méthode peut être appliquée aussi à la restauration de films anciens.

Les publications spécifiques à la restauration de défauts dans les films anciens sont beaucoup moins nombreuses. A notre connaissance un des premiers articles à ce sujet est celui de Richard Storey [81]); il utilise des critères de connexité temporelle simples pour détecter les taches. Le résultat est satisfaisant dans les zones immobiles de la séquence, mais ailleurs la méthode fonctionne mal.

Stuart Geman, Donald E. McClure et Donald Geman ont été les premiers à appliquer les techniques de restauration Bayesiennes (introduites dans [25]) aux séquences d'images (voir [26]).

Des chercheurs de l'Université de Cambridge s'occupent de restauration de films anciens depuis le début des années 90. Ils ont continué à développer les techniques Bayesiennes [62], et ont mis au point d'autres méthodes de restauration telles que "l'index de détection de pointes" (*spike detection index* [41, 43]) et le modèle auto-régressif 3D (*3D Auto-Regressif Model* [46]). Les publications résultantes sont nombreuses: [41, 62, 59, 60, 61, 44, 45, 43, 29, 30]. Dans tous les cas les méthodes présentées respectent la même structure: une première phase de compensation de mouvement, pour laquelle ils ont choisi un algorithme de block-matching hiérarchique [3], une seconde phase de détection des défauts et finalement une phase de récupération de l'information. Dans [44] on trouve un résumé intéressant de plusieurs méthodes de détection, anciennes et nouvelles, et dans [45] un panorama des méthodes d'interpolation.

Des modèles auto-régressifs 3D ont aussi été étudiés par des chercheurs de l'Université de Singapour [31, 38, 9]. La structure globale de leurs méthodes est similaire à celle des chercheurs de Cambridge.

Suite aux travaux présentés dans [66], des chercheurs du centre de Morphologie Mathématique de l'École des Mines de Paris se sont intéressés à l'application des fermetures et ouvertures par reconstruction à la détection des défauts locaux aléatoires [13, 14, 17, 16]. Ces travaux sont à la base du matériel présenté dans ce chapitre.

6.1.2 Notre approche

Notre approche, comme la plupart de celles que nous venons de citer, comporte deux parties principales. D'abord une détection des défauts, et ensuite une phase de récupération de l'information perdue.

Cependant, dans ce qui suit nous n'utilisons pas de phase de compensation de mouvement avant la détection. Nous avons fait ce choix pour plusieurs raisons, que nous expliquons dans la section 6.2.2.

Nous commencerons par analyser le problème de la détection des défauts locaux aléatoires. Cette étude nous dévoilera deux voies possibles pour la détection de ces défauts, voies que nous emprunterons dans les deux sections suivantes. Finalement, une fois les défauts détectés, nous nous occuperons de la récupération de l'information perdue.

6.2 Analyse du problème

6.2.1 Commentaire rapide sur le choix du nom "défaut aléatoire"

Le choix de l'adjectif "aléatoire" pour qualifier les défauts que nous traitons dans ce chapitre n'est pas très approprié. En effet, la caractéristique essentielle que nous utiliserons pour leur détection est la non-superposition entre trames consécutives, ou plus précisément, la faible probabilité de superposition, et non pas leur caractère aléatoire. Nous pouvons très bien imaginer des défauts parfaitement aléatoires et qui se superposent toujours entre trames consécutives. Supposons par exemple que toutes les trames paires sont dégradées par une rayure blanche verticale, dont la position est aléatoire, et que les trames impaires présentent une rayure blanche horizontale, aléatoire aussi. C'est certes un défaut peu vraisemblable, mais qui malgré son caractère aléatoire se superpose toujours à lui même entre trames consécutives et qui est par conséquent indétectable pas les méthodes présentées dans ce chapitre. Réciproquement, un défaut périodique qui effacerait la même région de toutes les trames paires n'aurait rien d'aléatoire et pourtant les méthodes présentées le restaureraient facilement.

Nous aurions donc dû appeler ces défauts "locaux à faible probabilité de superposition entre trames voisines", mais nous avons préféré le premier nom car tous les défauts de ce type que nous avons rencontrés sont bien décrits par des modèles aléatoires. De plus, le nom "local aléatoire" (que nous résumerons souvent à "aléatoire") est plus pratique à manipuler que celui de "locaux à faible probabilité de superposition entre trames voisines".

Nous verrons dans la suite que nous exploiterons cette faible probabilité de superposition pour détecter les défauts locaux aléatoires.

6.2.2 Compensation de mouvement

La plupart des techniques de détection de défauts locaux aléatoires exploitent la corrélation temporelle élevée propre aux séquences d'images de films ou vidéo (voir par exemple [1, 41, 62, 44]). Cette corrélation serait parfaite (abstraction faite des perturbations introduites par l'acquisition des images) si la scène était éclairée de façon uniforme et surtout s'il n'y avait pas de mouvement. Par conséquent une idée naturelle et séduisante pour améliorer la corrélation temporelle d'une séquence d'images et d'effectuer une compensation de mouvement. Cette opération consiste à prendre une image comme référence et à modifier les autres de façon à annuler le mouvement qui a eu lieu entre l'image de référence et les autres images. Pour cela il faut estimer le mouvement.

Cependant ce problème est très compliqué. En effet, non seulement les déplacements

peuvent être complexes, mais en plus des objets peuvent apparaître ou disparaître, ou être partiellement cachés, et l'éclairage de la scène peut varier au cours du temps. De nombreux chercheurs travaillent sur le sujet, comme en témoigne une littérature abondante. Depuis les techniques différentielles ([6, 34]), en passant par les méthodes basées sur des blocs , dites de *block-matching* ([36]), sur des nœuds ([48, 22]) ou sur une segmentation de la séquence ([65]), jusqu'aux plus récentes méthodes stochastiques ([47, 33, 89]) des centaines d'articles ont été publiés, mais il n'existe pas encore de solution parfaite.

De plus, dans notre cas, les images sont de mauvaise qualité et présentent des défauts, ce qui rend le fonctionnement des algorithmes de compensation de mouvement encore plus délicat.

Finalement, nous voulons étudier à fond les possibilités offertes par notre approche, basée uniquement sur des critères de connexité de la séquence, considérée comme une structure tridimensionnelle. Le choix d'un algorithme particulier de compensation de mouvement aurait risqué de biaiser notre étude. Nous verrons que dans la plupart des cas les méthodes de détection de défauts que nous proposons sont suffisantes, et que même en présence de mouvement important les résultats restent corrects.

C'est pour toutes ces raisons que nous avons choisi pour l'instant de ne pas utiliser de compensation de mouvement pour la détection des défauts aléatoires. Nous nous limiterons à une approche uniquement basée sur la connexité de la séquence, considérée en tant que structure tridimensionnelle.

Ceci dit, on peut toujours envisager d'utiliser un algorithme de compensation de mouvement dans une phase de prétraitement. S'il fonctionne bien, il ne pourra que faciliter les étapes suivantes.

Par ailleurs, lorsque nous aurons détecté avec précision l'emplacement des défauts, nous utiliserons des algorithmes d'estimation de mouvement pour aller chercher l'information manquante dans les autres trames de la séquence.

6.2.3 Étude de la connexité

Nous avons choisi une approche uniquement basée sur l'étude de la connexité de la séquence pour détecter les défauts locaux aléatoires. L'idée de base, que nous détaillons plus loin, consiste à formuler l'hypothèse que les défauts aléatoires sont les objets de la séquence qui sont isolés aussi bien d'un point de vue spatial que d'un point de vue temporel.

Qu'est-ce que cette approche implique?

Avant tout cette méthode nous permet d'exploiter de façon naturelle la définition des défauts aléatoires. D'une part le fait qu'ils soient locaux implique qu'ils resteront la plupart du temps isolés des vrais objets de la scène, et d'autre part le fait que la probabilité de superposition entre trames contigües soit faible implique que la plupart du temps ils seront isolés dans le temps.

Ceci dit, cette approche pose aussi quelques problèmes.

Premièrement, même si les défauts aléatoires sont "la plupart du temps isolés" des autres objets de la scène, il peut arriver qu'ils se collent à ceux-ci. Par exemple, si une

tache blanche apparaît sur la limite entre un mur blanc et un fond noir, alors il nous sera difficile de l'identifier en tant que défaut.

Deuxièmement, malgré le fait que "la probabilité de superposition entre trames contigües soit faible" il est toujours possible qu'un défaut apparaisse à la même position pendant plusieurs trames consécutives. Dans ce cas nous aurons aussi beaucoup de mal à le reconnaître. Nous essaierons de maintenir cette probabilité aussi basse que possible en prenant une connexité temporelle faible.

Troisièmement et dernièrement, les objets de la scène qui bougent si vite que leurs positions dans des trames consécutives ne se superposent pas risquent d'être pris pour de défauts (voir figure 6.1). Une solution possible, comme nous l'avons déjà dit, serait d'utiliser un algorithme de compensation de mouvement mais nous avons choisi de laisser cette méthode de côté au vu de tous les problèmes pratiques (problèmes qui ne font que se compliquer en présence de défauts) que pose encore l'estimation de mouvement dans les séquences d'images. Nous verrons dans la suite comment pallier cet inconvénient.

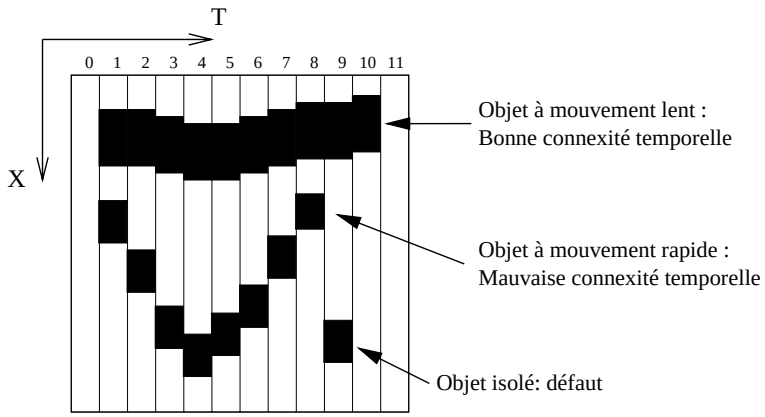


FIG. 6.1 - *Illustration des problèmes de connexité temporelle provoqués par le mouvement. La figure représente une séquence d'images binaire vue en coupe suivant l'axe Y (le plan de la coupe est donné par les axes X et T).*

6.3 Comment détecter les défauts aléatoires?

Afin d'analyser une séquence d'images et d'y repérer les défauts, nous avons besoin de nouveaux outils. Ces outils travailleront sur la séquence d'images en tant que structure tridimensionnelle. Nous commençons par les définir dans le cas d'une séquence binaire B. Chaque outil est en fait une mesure des composantes connexes maximales de B.

6.3.1 Différentes mesures des composantes connexes maximales d'une séquence binaire

Soit donc B une séquence d'images binaires, munie d'une connexité spatiale et d'une connexité temporelle données respectivement par les dilatations δ_s et δ_t. Rappelons que la connexité de la section est alors donnée par $\delta_{st} = \delta_s \bigvee \delta_t$ (voir section 2.4). Dans la suite C est un sous-ensemble de B.

Nous aurons aussi besoin de la définition suivante:

Définition 6.1 *Portée temporelle:* $\Delta_t(C)$
La portée temporelle (ou portée tout simplement) de C*, notée* $\Delta_t(C)$ *est donnée par:*

$$\Delta_t(C) = max\{t_j - t_i + 1 \mid (x_i, y_i, t_i), (x_j, y_j, t_j) \in C\}$$

Nous utiliserons la figure 6.2 pour illustrer le fonctionnement des différentes mesures que nous allons construire. Cette figure représente une coupe parallèle au plan (Oxt) d'une séquence d'images binaires. Observez que, avec la connexité choisie (représentée en haut à droite de la figure), nous avons 3 composantes connexes maximales sur cette séquence d'images. La composante (A) est un objet qui bouge rapidement, la composante (B) correspond probablement à un défaut, et (C) cache plusieurs objets, dont les trajectoires se croisent.

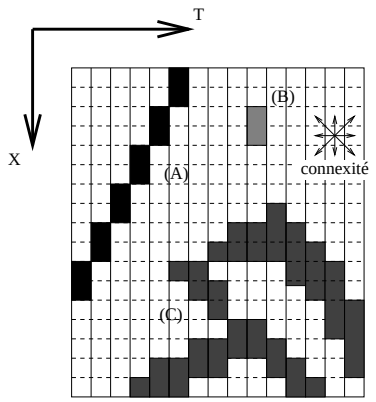


FIG. 6.2 - *Séquence illustrative*

6.3.1.1 Longueur temporelle

Définition 6.2 *Chemin.*
Un chemin L *de* n *pixels est une suite de pixels* $(P_i)_{0 \leq i \leq n-1}$ *telle que:*

$$\forall i,\ 0 \leq i \leq n-2 \quad P_{i+1} \in \delta_{st}(\{P_i\})$$

Définition 6.3 *Longueur temporelle: $lt(C)$.*
La longueur temporelle de C, notée $lt(C)$ est le maximum des portées des chemins contenus dans C.

$$lt(C) = max\{\Delta_t(L),\ L\ chemin\ inclus\ dans\ C\} \tag{6.1}$$

Proposition 6.1

$$\Delta_t(C) \geq lt(C) \tag{6.2}$$

Si C est connexe, alors $\Delta_t(C) = lt(C)$

En effet, l'ensemble des couples de points appartenant à C et qui sont reliés par un chemin inclus dans C est inclus dans l'ensemble des couples de points appartenant à C. Et ces deux ensembles sont égaux si C est connexe.

La figure 6.3 montre quelques exemples de mesures de longueur temporelle pour les composantes connexes maximales d'une séquence.

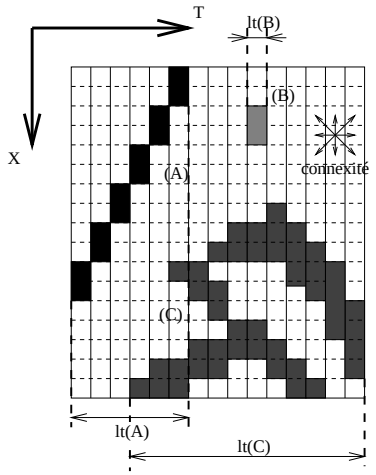


FIG. 6.3 - *Exemple de mesures de longueur temporelle*

6.3.1.2 Permanence temporelle

Définition 6.4 *Chemin temporel.*
Un chemin temporel L de n pixels est une suite de pixels $(P_i)_{0\leq i\leq n-1}$ de coordonnées respectives $(x_i, y_i, t_i)_{0\leq i\leq n-1}$ tels que $\forall i,\ 0 \leq i \leq n-2$ ou bien $P_{i+1} \in \delta_s(P_i)$ ou bien $t_{i+1} = t_i + 1$ et $P_{i+1} \in \delta_t(P_i)$.

En d'autres termes, un chemin temporel est un chemin dont les coordonnées temporelles des points qui le constituent sont croissantes: il va forcément vers le futur (mais de façon non stricte).

Définition 6.5 *Permanence temporelle: $p(C)$.*
Soit C un sous-ensemble de la séquence binaire B. La permanence temporelle de C, notée $p(C)$, est le maximum des portées des chemins temporels contenus dans C.

$$p(C) = max\{\Delta_t(L),\ L \text{ chemin temporel inclus dans } C\} \tag{6.3}$$

Proposition 6.2

$$lt(C) \geq p(C). \tag{6.4}$$

Cette relation découle du fait que l'ensemble des chemins temporels inclus dans C est compris dans l'ensemble des chemins inclus dans C.

La figure 6.4 montre quelques exemples de mesures de permanence temporelle pour les composantes connexes maximales de notre séquence test.

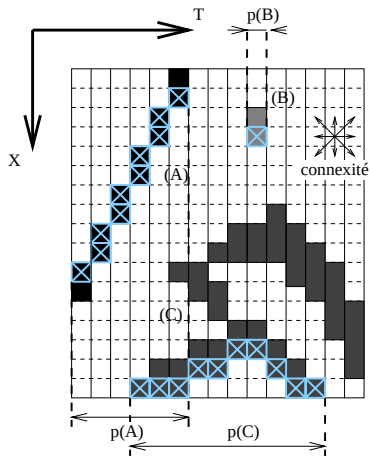


FIG. 6.4 - *Exemple de mesures de permanence temporelle. Nous avons marqué par des croix le chemin temporel maximal qui dans chaque cas a permis de mesurer la permanence temporelle.*

6.3.1.3 Permanence temporelle stricte

Définition 6.6 *Chemin temporel strict.*
Un chemin temporel strict L de longueur n commençant au temps t_0 est une suite de n pixels $(P_i)_{0\leq i\leq n-1}$ de coordonnées respectives $(x_i, y_i, t_0 + i)$ tels que:

$$\forall i,\ 0 \leq i \leq n-2 \quad P_{i+1} \in \delta_t(P_i)$$

Dit autrement, un chemin temporel strict est un chemin dont les coordonnées temporelles des points qui le constituent sont strictement croissantes.

Remarquons que la longueur d'un chemin temporel strict est égale à sa portée.

Définition 6.7 *Permanence temporelle stricte: $ps(C)$*
La permanence temporelle stricte de C, notée $ps(C)$, est le maximum des portées des chemins temporels stricts contenus dans C.

$$ps(C) = max\{\Delta_t(L),\ L \text{ chemin temporel strict inclus dans } C\} \tag{6.5}$$

Proposition 6.3

$$p(C) \geq ps(C) \tag{6.6}$$

La figure 6.5 montre quelques exemples de mesures de permanence temporelle stricte pour les composantes connexes maximales d'une séquence.

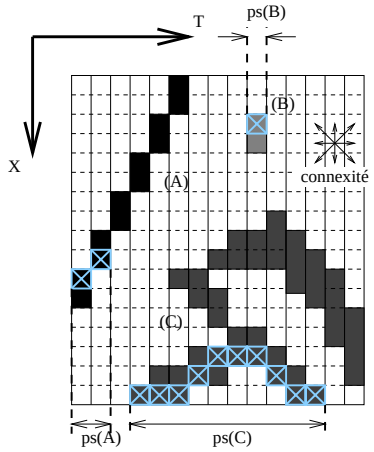


FIG. 6.5 - *Exemple de mesures de permanence temporelle stricte*

6.3.1.4 Épaisseur temporelle

Définition 6.8 *Segment temporel.*
Un segment temporel S de longueur n est une suite de n pixels $(P_i)_{0\leq i\leq n-1}$ de coordonnées respectives $(x, y, t_0 + i)_{0\leq i\leq n-1}$.

En d'autres termes, un segment temporel est un segment parallèle à l'axe du temps.

Définition 6.9 *Épaisseur temporelle:* $e(C)$
L'épaisseur temporelle d'un sous-ensemble C de B, que nous noterons $e(C)$, est le maximum des portées des segments temporels inclus dans C.

$$e(C) = max\{\Delta_t(L),\ L\ segement\ temporel\ inclus\ dans\ C\} \tag{6.7}$$

Remarquons que cette définition est indépendante de la connexité spatio-temporelle de B.

Étant donné que les connexités temporelles que nous utiliserons seront supérieures à la connexité temporelle simple (définie dans la section 2.4), l'ensemble des segments temporels stricts contenus dans C est inclus dans l'ensemble des chemins temporels contenus dans C. Par ailleurs, si nous choisissons la connexité temporelle simple, alors les chemins temporels stricts sont les segments temporels. Par conséquent:

Proposition 6.4

$$ps(C) \geq e(C) \tag{6.8}$$

Si la connexité temporelle est simple alors $ps(C) = e(C)$.

La figure 6.6 montre quelques exemples de mesures d'épaisseur temporelle pour les composantes connexes maximales d'une séquence.

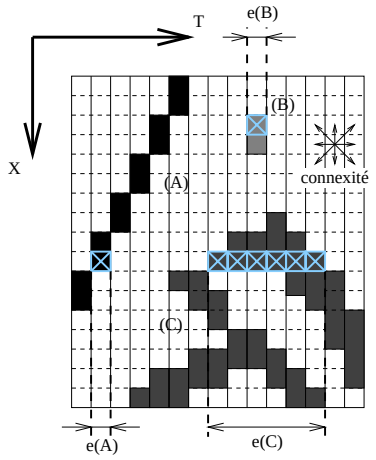


FIG. 6.6 - *Exemple de mesures d'épaisseur temporelle*

6.3.1.5 Des mesures hiérarchisées

Les différentes mesures que nous venons de présenter sont ordonnées. En effet, d'après les propositions 6.1, 6.2, 6.3 et 6.4, pour tout sous-ensemble C de B nous avons la relation:

$$lt(C) \geq p(C) \geq ps(C) \geq e(C) \tag{6.9}$$

Toutes ces mesures donnent des valeurs pratiquement identiques, et grandes , dans le cas d'objets immobiles ou bougeant très lentement. La différence entre les mesures grandira lorsqu'il s'agira d'objets qui bougent, le cas extrême étant celui où le mouvement entre deux trames sera plus grand que la taille de l'objet; on aura alors (à condition que la connexité temporelle soit suffisamment forte pour que l'objet constitue une seule composante connexe) une longueur temporelle égale au nombre de trames où l'objet est présent, et une épaisseur temporelle égale à 1. Enfin, s'il s'agit d'un objet visible uniquement sur une trame alors toutes les mesures seront à nouveau d'accord: elles vaudront toutes 1.

Ceci dit, nous n'avons pas encore expliqué comment les calculer en pratique.

6.3.2 Mise en œuvre des mesures: algorithmes

Les algorithmes que nous décrivons dans cette section ont pour but de faire le tri en fonction d'une valeur n parmi les CCM d'une séquence binaire B. Avec chaque mesure m (m peut être égale à lt, p, ps ou e) nous construirons une ouverture par reconstruction γ de la façon suivante: les CCM C telles que $m(C) > n$ seront préservées, et les autres seront entièrement effacées.

$$\gamma(B) = \bigcup \{C \ / \ C \in CCM(B),\ m(C) > n\} \tag{6.10}$$

Nous commençons par le cas de l'épaisseur temporelle, car l'algorithme correspondant est très connu en morphologie mathématique, puis nous traiterons le cas des autres mesures dans l'ordre inverse de la section précédente.

6.3.2.1 Ouverture par reconstruction avec l'épaisseur temporelle

Les composantes connexes maximales de B d'épaisseur temporelle inférieure ou égale à n ne contiennent pas de segment temporel de longueur $n+1$. Donc elles sont effacées par une ouverture temporelle de taille n. Réciproquement, les autres composantes connexes maximales de la séquence contiennent un segment temporel de longueur $n+1$, si bien qu'elles ne sont pas entièrement effacées par cette même ouverture temporelle. Une reconstruction permet alors de récupérer la forme initiale des objets qui ont survécu à l'ouverture. Le résultat est **l'ouverture par reconstruction** de la séquence initiale par un élément structurant linéaire parallèle à l'axe du temps de longueur n. La procédure complète est illustrée par la figure 6.7 dans le cas où $n = 2$.

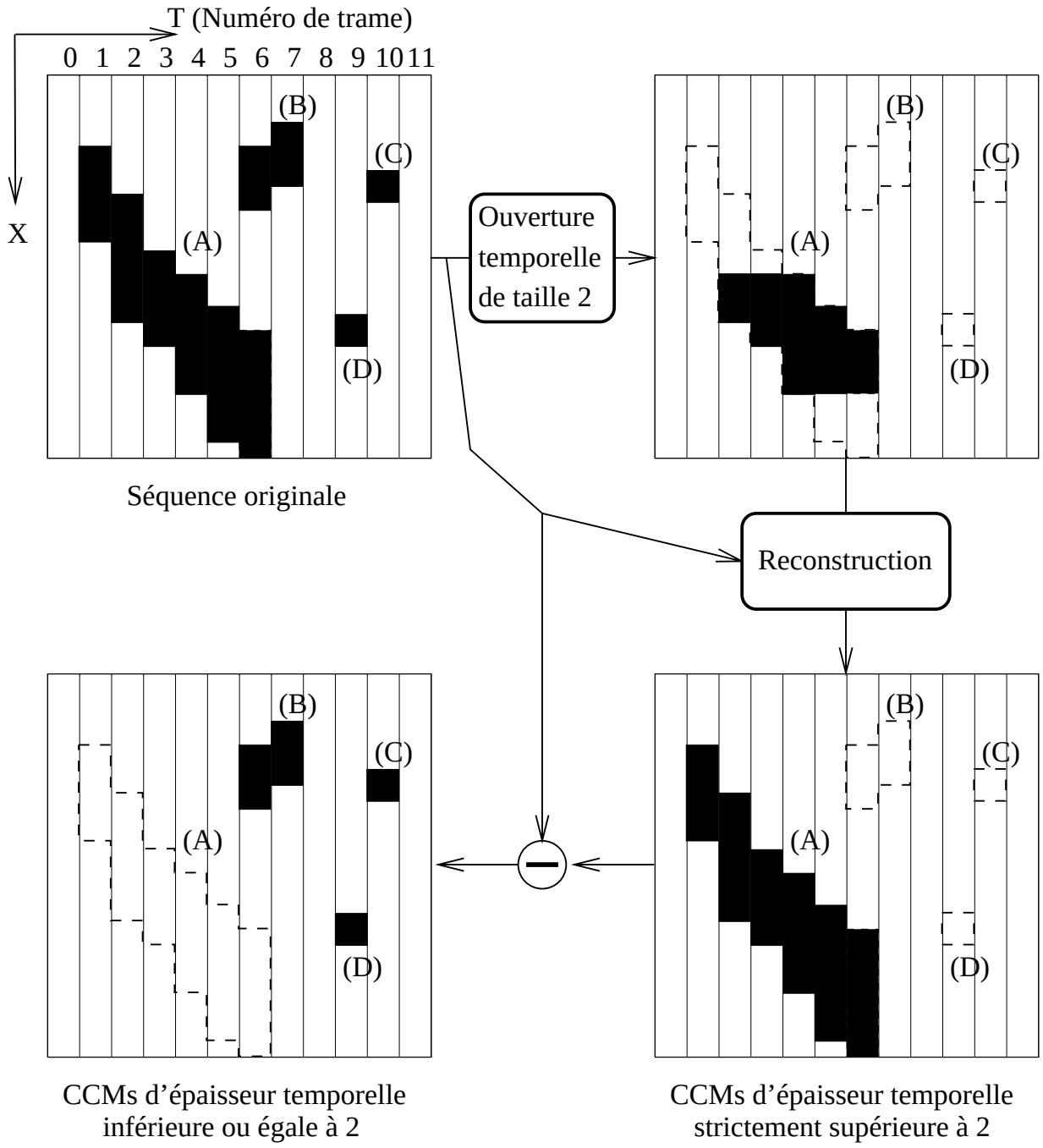


FIG. 6.7 - *Détection des défauts dont l'épaisseur temporelle est inférieure ou égale à 2.*

6.3.2.2 Ouverture par reconstruction avec la permanence temporelle stricte

Nous voulons repérer toutes les composantes connexes maximales contenant au moins un chemin temporel strict de longueur $n+1$. Pour ce faire, nous pouvons adopter une approche analogue à la précédente: utiliser une ouverture qui effacera complètement seulement les composantes connexes maximales de permanence temporelle stricte inférieure à n, et ensuite appliquer une reconstruction pour récupérer la forme initiale des objets qui n'ont pas été complètement effacés. Le résultat sera l'ouverture par reconstruction γ que nous cherchons.

Soit $\mathcal{C}_{n+1} = \{c_i\}_{i \in I}$ l'ensemble fini des chemins temporels stricts de longueur $n+1$ et γ_i l'ouverture morphologique avec le chemin c_i. Le sup $\bigvee \gamma_i$ de toutes ces ouvertures est une ouverture et elle rend exactement tous les points de B par lesquels passe un chemin temporel strict de longueur $n+1$ contenu dans B. Il suffit donc ensuite d'appliquer une reconstruction, comme précédemment.

Malheureusement, construire $\bigvee \gamma_i$ est peu pratique, le nombre d'éléments dans $\mathcal{C}_{n+1}$ pouvant être assez important. Au lieu de cela nous calculerons pour chaque pixel $P(x,y,t)$ de B la longueur maximale des chemins temporels stricts inclus dans B qui finissent en P. Nous noterons cette valeur $ps_f(P)$. Si dans une composante connexe maximale C de B il existe un point P tel que $ps_f(P) \geq n+1$ alors on est sûr que C contient un chemin temporel strict de longueur au moins égale à $n+1$. Réciproquement, si C contient un chemin temporel strict de longueur supérieur à $n+1$, alors il contient forcément un point P, contenu dans C, tel que $ps_f(P) \geq n+1$. Donc les pixels P tels que $ps_f(P) \geq n+1$ constituent des marqueurs des objets dont la permanence temporelle stricte est supérieure à n, si bien qu'après il suffira d'opérer une reconstruction à l'intérieur de B pour obtenir le résultat de l'ouverture par reconstruction.

Décrivons l'algorithme qui nous permet de calculer $ps_f(P)$ pour tous les pixels de B. Soit I une séquence d'images à niveaux de gris de mêmes dimensions que B et t un compteur entier. Soit S_t l'élément structurant de la dilatation δ_t associée à la connexité temporelle de la séquence. Soit alors S_t^+ égal à S_t privé des points dont la coordonnée temporelle est négative ($S_t^+ = \{P(x,y,t) \in S_t, t \geq 0\}$), et δ_t^+ la dilatation d'élément structurant S_t^+.

Initialisation: I=0 et t=0

1. $I = \delta_t^+(I) \cap B$
2. $I(t) = I(t) + B(t)$
3. $t = t + 1$
4. Si $t \leq t_{max}$ alors retour au (1)

Le fonctionnement de cet algorithme est illustré par la figure 6.8.

En sortie de cet algorithme, nous obtenons pour tout pixel P de B: $I(P) = ps_f(P)$. Il suffit ensuite de seuiller I au niveau $n+1$. L'image résultante $X_{n+1}(I)$ constitue le marqueur que nous cherchons pour les composantes connexes maximales de permanence

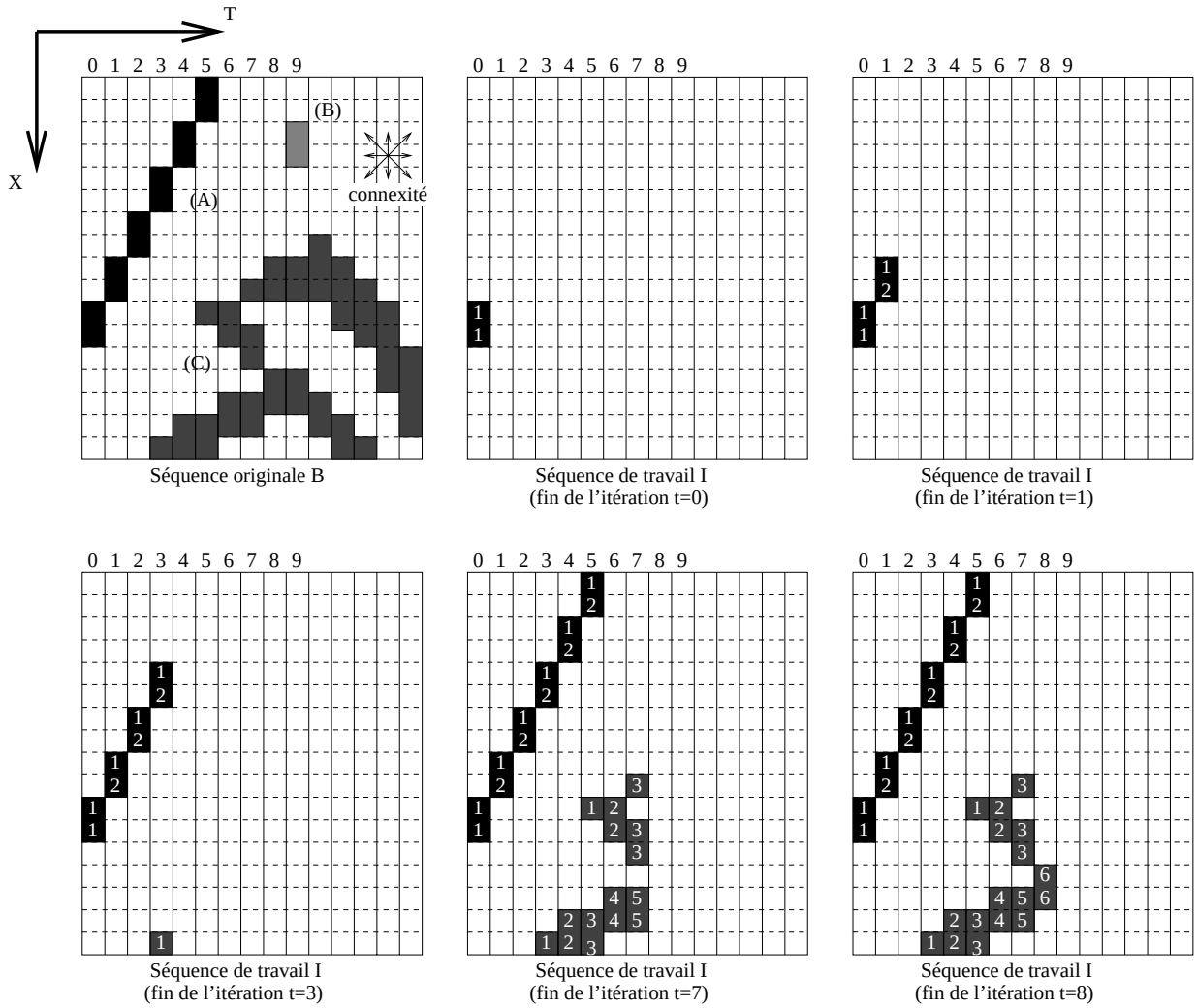


FIG. 6.8 - *Algorithme de calcul de ps_f.*

temporelle stricte supérieure ou égale à $n+1$. Il ne reste plus alors qu'à reconstruire $X_{n+1}(I)$ à l'intérieur de la séquence originale B: $\gamma(B) = Rec(X_{n+1}(I), B)$.

Cet algorithme a l'avantage d'être itératif. Il suffit de connaître $I(t)$ pour calculer $I(t+1)$.

6.3.2.3 Ouverture par reconstruction avec la permanence temporelle

De façon analogue à la précédente, nous définissons $p_f(P)$ comme étant le maximum des longueurs des chemins temporels inclus dans B finissant en P.

L'algorithme que nous avons développé pour le calcul de p_f est très similaire à celui que nous venons de présenter servant au calcul de ps_f. Il comporte une étape supplémentaire de reconstruction géodésique pour prendre en compte le fait que les chemins temporels ne sont plus stricts.

Initialisation: I=0 et t=0

1. $I = \delta_t^+(I) \cap B$
2. $I(t) = I(t) + B(t)$
3. $I(t) = Rec(I(t), B(t))$
4. $t = t + 1$
5. Si $t \leq t_{max}$ alors retour au (1)

Le fonctionnement de cet algorithme est illustré par la figure 6.9.

6.3.2.4 Ouverture par reconstruction en utilisant la longueur temporelle

Pour calculer la longueur temporelle il suffit d'utiliser un algorithme de labelisation parmi ceux qu'on trouve dans la littérature et de retenir, pour chaque label, les valeurs extrêmes des coordonnées temporelles des pixels correspondants (un algorithme de labelisation est une procédure qui permet d'associer à chaque composante connexe maximale un label différent).

6.3.3 Quelle mesure choisir?

Nous utiliserons les mesures présentées ci-dessus pour analyser la séquence binaire B. Les valeurs obtenues pour chaque composante connexe de la séquence nous permettront de décider lesquelles correspondent à des défauts. Mais laquelle devons-nous utiliser pour reconnaître au mieux les défauts?

Avant de répondre à cette question remarquons que dans certains cas la réponse n'est pas nécessaire... D'une part, si nous nous limitons à la connexité temporelle simple alors l'épaisseur temporelle et la permanence temporelle stricte donnent le même résultat (voir proposition 6.4). D'autre part remarquons que si une composante connexe maximale est

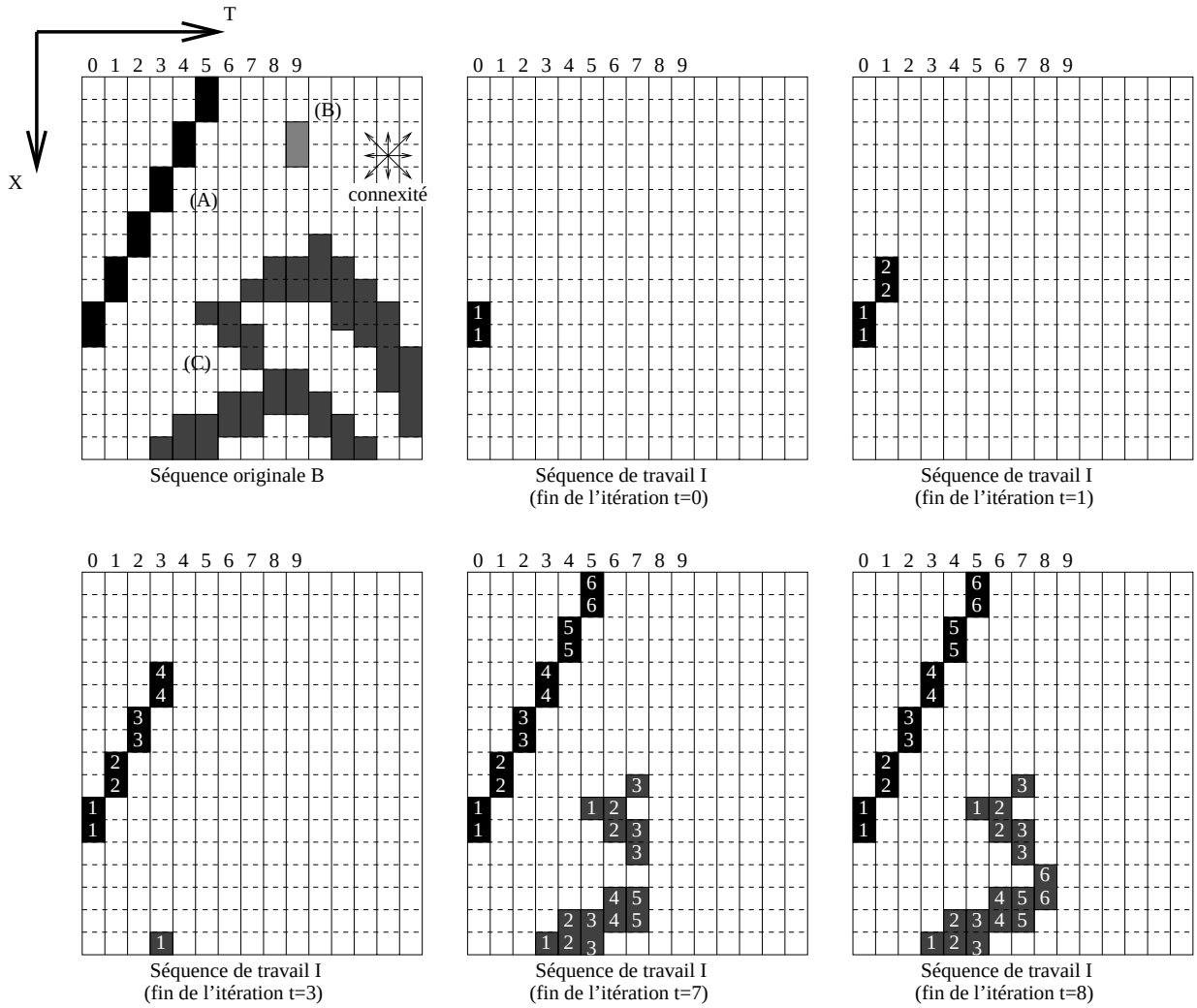


FIG. 6.9 - *Algorithme de calcul de p_f.*

telle que $lt(C) = 1$, alors, en vertu de la relation 6.9, les autres mesures seront aussi égales à 1. Et si $lt(C) = 2$ alors $p(C) = ps(C) = 2$ (mais pas nécessairement $e(C) = 2$).

Jetons un coup d'œil au tableau 6.1.

Composante connexe maximale	(A)	(B)	(C)
Épaisseur	1	1	6
Permanence stricte	2	1	10
Permanence	6	1	10
Longueur	6	1	12

TAB. 6.1 - *Résultats de l'application des différentes mesures présentées aux composantes connexes maximales de notre séquence illustrative.*

Nous remarquons que la permanence temporelle et la longueur temporelle sont les mesures qui caractérisent le mieux les défauts (ici la composante connexe (B)), ceci grâce à leur plus grande robustesse vis à vis des objets en mouvement. Parmi ces deux mesures nous choisirons la permanence temporelle parce qu'elle impose une certaine cohérence aux objets en mouvement et empêche en particulier des "retours en arrière".

6.3.4 Application à la détection de défauts

Supposons qu'une séquence d'images I présente un seul type de défaut aléatoire, et que nous disposons d'un critère qui nous permet de détecter, trame par trame, des régions qui sont soupçonnées d'être des défauts (nous dirons que ce sont des régions qui sont des **défauts potentiels**). Nous obtenons ainsi une séquence binaire B que nous munissons d'une connexité spatio-temporelle. Ses composante connexes maximales correspondent soit à des défauts, soit à des objets de la scène qui ont été détectés de façon erronée à cause de leur ressemblance avec les premiers. Étant donné que la probabilité que les défauts aléatoires se superposent entre deux trames successives est faible, ceux-ci apparaîtront généralement sous la forme de composantes connexes maximales de faible permanence temporelle. Nous construirons donc notre modèle à partir de cette remarque:

Définition 6.10 *Modèle de défaut local aléatoire dans une séquence binaire.*
Nous appellerons ***défaut local aléatoire d'épaisseur*** n *les composantes connexes maximales de* B *dont la permanence temporelle est inférieure ou égale à* n*.*

Par exemple nous pouvons décider de considérer comme défauts toutes les régions étant des défauts potentiels dont la permanence temporelle est inférieure ou égale à 2; dans ce cas les objets (B), (C) et (D) de la figure 6.10 seront considérés comme étant des défauts, ce qui semble vraisemblable.

Il nous reste donc deux paramètres à fixer: l'épaisseur des défauts que nous cherchons n et la connexité spatio-temporelle de la séquence.

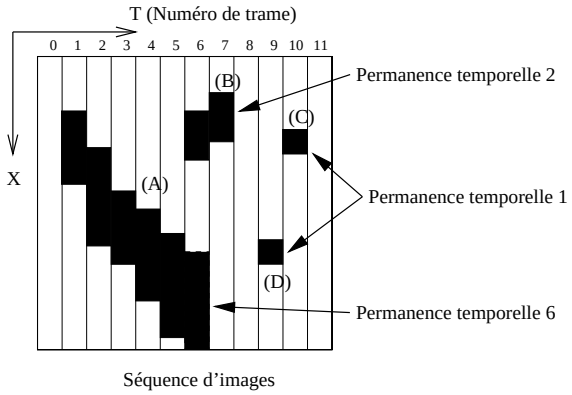


FIG. 6.10 - *Mesures de permanence temporelle.*

N'oublions pas que, au vu des contraintes de qualité que nous nous sommes imposées, nous choisirons de ne pas détecter certains défauts plutôt que d'abîmer des zones non-dégradées de l'image.

L'épaisseur des défauts que nous choisirons dépendra essentiellement de la densité de ceux-ci. Si la densité est faible (moins de 1% de l'image dégradée) alors on prendra $n = 1$. Avec une densité moyenne (entre 1% et 5% de l'image) on prendra $n = 2$. Finalement, pour les cas extrêmes, on choisira $n = 3$.

Par ailleurs nous avons choisi la 8-connexité spatiale. Nous obtenons ainsi moins de composantes connexes dans chaque trame et les objets fins risquent moins de se voir coupés.

Le choix de la connexité temporelle est beaucoup plus délicat. Il dépend essentiellement de la présence ou non de mouvement important dans la scène.

6.4 Première méthode de détection

Nous disposons d'un modèle de défaut aléatoire dans le cas des séquences d'images binaires. Nous ne pouvons pas l'appliquer directement à des séquences à niveaux de gris. Par conséquent dans cette section nous commençons par proposer quelques algorithmes de détection qui nous permettent d'identifier trame par trame les défauts potentiels. Nous obtenons ainsi une séquence d'images binaires. Il ne nous restera alors plus qu'à appliquer les algorithmes décrits dans les sections précédentes pour reconnaître, parmi tous les candidats, les défauts aléatoires.

6.4.1 Reconnaissance trame par trame des défauts aléatoires

Chaque type de défaut local aléatoire présente certaines caractéristiques spatiales qui permettent de le reconnaître sur chaque trame de la séquence, indépendamment de toute caractéristique temporelle. Ainsi, les taches claires apparaissent comme des maxima lo-

caux de forme plus ou moins ronde; les craquelures sont des structures allongées, étroites et sombres globalement horizontales. Ceci dit, certains objets de la scène peuvent avoir les mêmes caractéristiques, et risquent donc d'être détectés aussi. Par exemple les fleurs blanches peuvent avoir une forme très similaire aux taches claires, et l'ombre d'un toit peut ressembler à une craquelure. Ce n'est pas grave. Une analyse temporelle postérieure permettra de distinguer les vrais défauts des fausses détections. Par conséquent la caractéristique essentielle des algorithmes de détection trame par trame est de ne pas laisser échapper des défauts. La présence de fausses détections est certes dommage mais moins grave.

Nous présenterons deux algorithmes de détection trame par trame basés sur des outils de morphologie mathématique. Le premier permet de détecter les taches d'une taille donnée et le deuxième permet de détecter les craquelures.

6.4.1.1 Détection trame par trame de taches claires

Nous modélisons les taches claires comme étant des maxima locaux de surface maximale S_{max}, de surface minimale S_{min} et de contraste minimal g.

Afin de les détecter nous utilisons des filtres aréolaires [85, 86], dont nous avons rappelé la définition dans le chapitre 2. La suite d'opérations que nous utilisons, et qui est illustrée dans la figure 6.11, est la suivante (la trame originale est $F = I(t)$):

1. Ouverture aréolaire de taille S_{max} de F, $\Rightarrow F_1$. On prive ainsi F de tous ses maxima de surface inférieure à S_{max}.

2. $F_2 = F - F_1$. F_2 est le résidu de l'opération précédente. Il contient donc les maxima de F de surface inférieure à S_{max}.

3. Seuillage au niveau g de $F_2 \Rightarrow F_3$. Nous éliminons ainsi les maxima de contraste inférieur à g.

4. Ouverture par reconstruction en utilisant comme masque F_2 et comme marqueur F_3; on obtient ainsi F_4 qui contient les maxima de F de surface inférieure à S_{max} et de contraste supérieur à g.

5. Binarisation de F_4, donnant F_5. On utilise pour cela un seuil bas g_{bas}.

6. Ouverture aréolaire de taille S_{min} de F_5. On obtient ainsi F_6, qui est égal à F_5 privé de ses composantes connexes de surface inférieure à S_{min}.

L'image binaire F_6 ainsi obtenue indique l'emplacement exact des objets décrits par notre modèle de taches claires. L'application de notre algorithme à la première image de notre séquence "Train" donne le résultat de la figure 6.12. Nous pouvons constater que la détection est satisfaisante. Toutes les taches claires ont été repérées. Les paramètres utilisés apparaissent dans le tableau 6.2.

Cet algorithme peut être perfectionné. Par exemple on peut appliquer des critères de forme pour ne pas détecter les taches qui sont trop allongées.

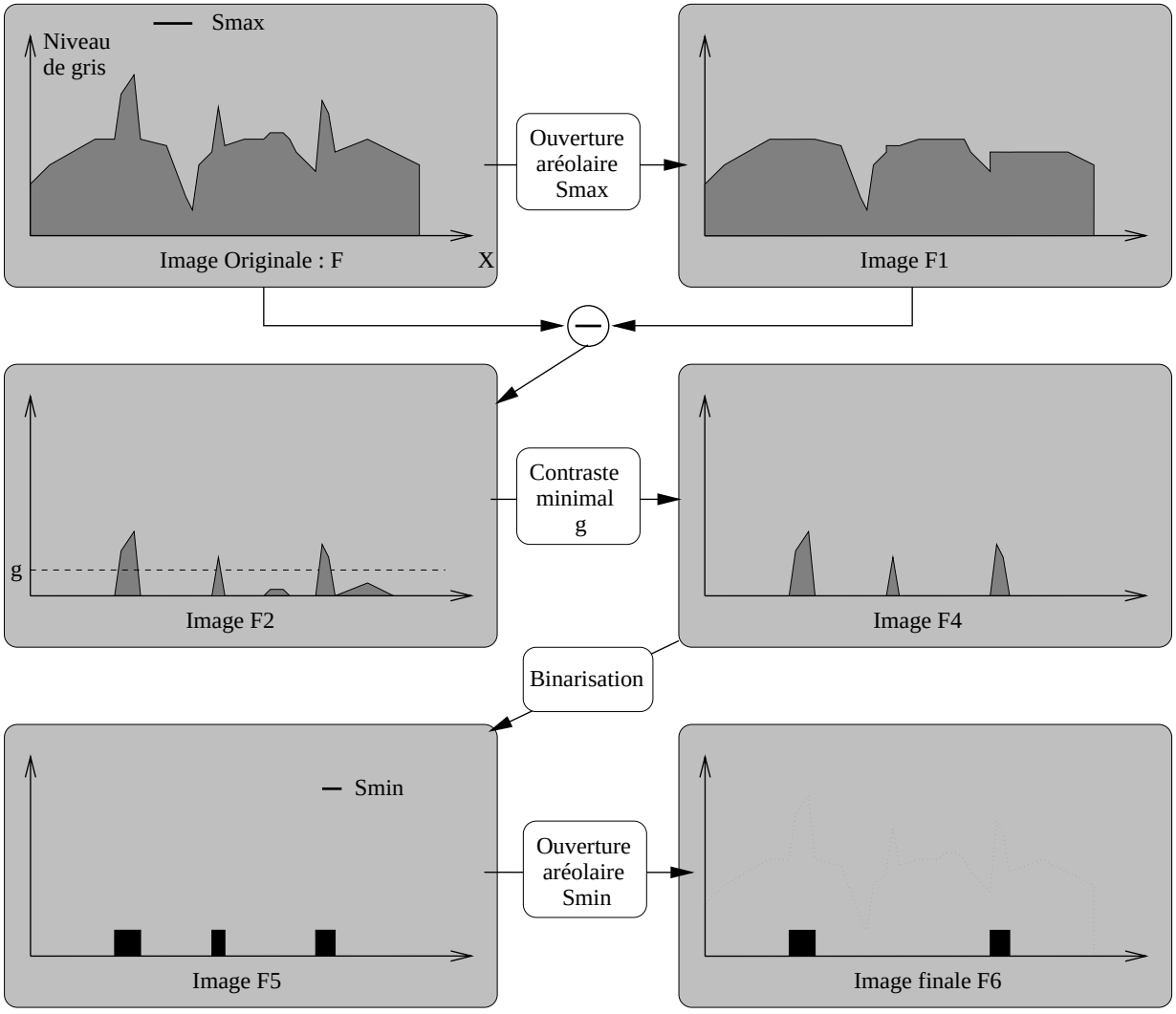
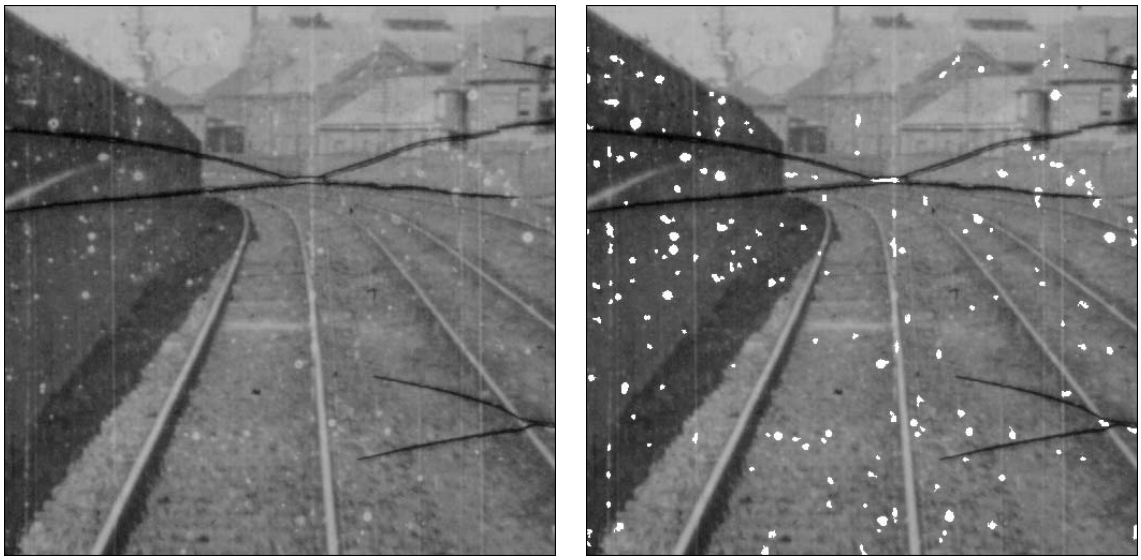


FIG. 6.11 - *Différentes étapes de l'algorithme de détection de taches blanches*

Paramètre	Valeur
S_{max}	100
S_{min}	16
g	25
g_{bas}	10

TAB. 6.2 - *Paramètres utilisés pour la détection des taches de la séquence "Train".*



Trame originale Détection de taches blanches

FIG. 6.12 - *Résultat de l'application de l'algorithme de détection de taches claires à la première trame de la séquence "Train"*

6.4.1.2 Détection trame par trame de craquelures noires

Comme vous avez pu le remarquer, les images de la séquence "Train" présentent des craquelures noires horizontales. Il s'agit là d'un défaut très rare est extrêmement difficile à traiter, mais nous verrons que nous obtenons des résultats encourageants.

Afin de les détecter nous adoptons un modèle analogue à celui que nous avons utilisé pour les taches blanches. Mais avant tout nous avons besoin d'une définition:

Définition 6.11 *Epaisseur d'un ensemble C de $\mathbb{R}^n$ suivant la direction $\vec{u}$.*
Soit C un sous-ensemble borné de $\mathbb{R}^n$ et $\vec{u}$ un vecteur de $\mathbb{R}^n$. L'épaisseur de C suivant la direction $\vec{u}$ est le maximum des longueurs des segments colinéaires à $\vec{u}$ contenus dans C. Remarquons que $\vec{u}$ peut être égal à $\vec{0}$; dans ce cas l'épaisseur correspondante est le maximum des longueurs de tous les segments contenus dans C.
Nous décrivons les craquelures comme étant des structures sombres de contraste au moins égal à g, d'épaisseur verticale inférieure à d_x et de surface supérieure à S_{min}.

Pour mettre en œuvre ce modèle nous utilisons la suite d'opérations suivante:

1. Chapeau haut de forme sombre avec un élément structurant linéaire vertical de longueur d_x. On obtient F_1.
2. Binarisation de F_1 avec le seuil g. Nous appellerons le résultat F_2.
3. Binarisation de F_1 avec un seuil bas g_{bas} (pour limiter le bruit): F_3.

4. Reconstruction en 4-connexité de F_3 en utilisant comme marqueur F_2. Le résultat est stocké dans F_4.

5. Ouverture aréolaire de taille S_{min} de F_4. On efface ainsi les composantes connexes maximales de F_4 de petite taille pour obtenir le résultat final.

Nous avons appliqué cet algorithme à la première image de notre séquence "Train" avec les paramètres donnés par le tableau 6.3. Le résultat apparaît sur la figure 6.13. Nous pouvons constater que toutes les craquelures ont été correctement détectées, mais certains objets de la scène dont les caractéristiques géométriques et les niveaux de gris correspondaient aussi à notre modèle ont été marqués aussi. En particulier l'ombre des toits a souvent été prise pour des craquelures. Ce problème sera résolu dans la section suivante.

Paramètre	Valeur
d_x	9
S_{min}	200
g	40
g_{bas}	5

TAB. 6.3 - *Paramètres utilisés pour la détection des craquelures de la séquence "Train".*

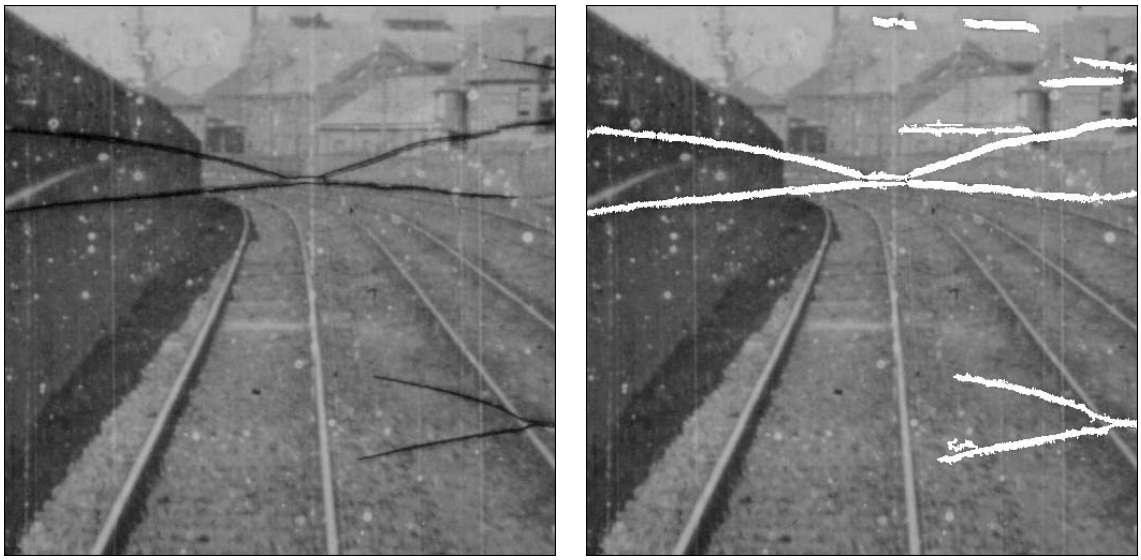
6.4.2 Analyse de la séquence binaire

A l'issue de la première phase de détection trame par trame, nous nous retrouvons avec une séquence binaire B dont les composantes connexes maximales correspondent à des défauts ou à des objets de la scène. Afin de reconnaître les vrais défauts nous utiliserons le modèle présenté au début de ce chapitre (cf. définition 6.10).

Nous munirons donc B d'une connexité spatio-temporelle. Il nous faut maintenant reconnaître les composantes connexes maximales de B dont la permanence temporelle est inférieure ou égale à n. Nous commencerons par étudier le cas où la connexité temporelle est simple, pour ensuite nous plonger dans le cas général.

6.4.2.1 Connexité temporelle simple

Dans le cas de la séquence "Train" le mouvement est suffisamment faible pour que nous puissions nous limiter à une connexité temporelle simple. Considérons les défauts d'épaisseur 2. D'après la section 6.3.3 nous pouvons nous contenter d'utiliser une ouverture temporelle pour identifier les défauts. C'est ce que nous montrons dans la figure 6.14. En haut de la figure (a) nous observons les 5 images originales. La ligne suivante (b) montre le résultat de la détection trame par trame, que nous avons décrite dans la section 6.4.1.2. Vu que nous avons effectué une ouverture avec un élément structurant linéaire de longueur 2



Trame originale Détection de craquelures

FIG. 6.13 - *Résultat de l'application de l'algorithme de détection de taches claires à la première trame de la séquence "Train"*

parallèle à l'axe du temps le résultat n'a de sens que pour la trame centrale (c). Remarquons que nous avons réussi à enlever les fausses détections (en particulier l'ombre des toits des maisons). Par contre nous avons aussi effacé une vraie craquelure. Pourquoi? Simplement parce que, avec d'autres craquelures dans les trames précédentes, elle constitue une CCM de longueur supérieure à 3.

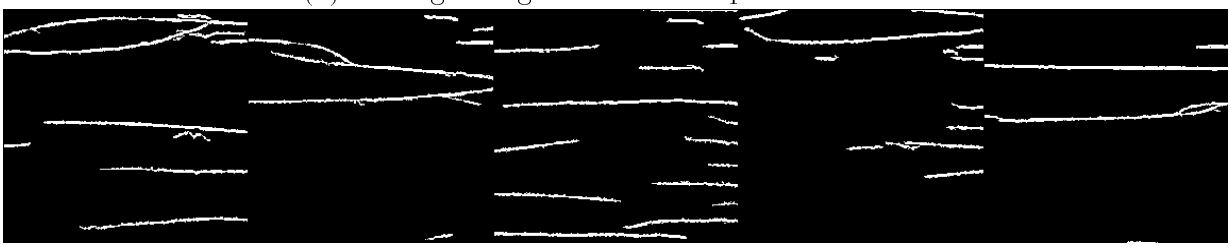
Afin de mieux comprendre la structure de cette séquence regardons le résultat du calcul de p_f (figure 6.15) (rappelons que pour un pixel P de B $p_f(P)$ donne le maximum des portées des chemins temporels finissant en P). Sur la première trame tous les pixels P appartenant à B sont tels que $p_f(P) = 1$. C'est normal car nous avons considéré qu'il s'agissait de la première trame de la séquence. Ce qui est intéressant c'est de regarder les régions où p_f atteint des valeurs importantes. On remarquera qu'il s'agit essentiellement des objets de la scène qui avaient été pris pour des craquelures. Ceci dit, on retrouve la craquelure qui nous a posé des problèmes précédemment: p_f prend des valeurs importantes sur elle.

En fait les craquelures, étant donnée leur grande taille, ne rentrent plus tout à fait dans le cadre des défauts locaux aléatoires car la probabilité de superposition est trop élevée. Cependant, les résultats obtenus sont intéressants.

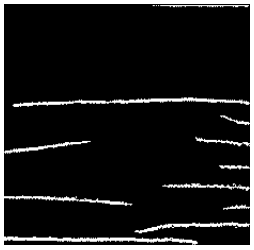
6.4.2.2 Connexité temporelle quelconque

Voyons maintenant ce qui se passe en présence de mouvement plus important. Nous avons choisi une séquence de bonne qualité où on observe de nombreux objets clairs qui

(a) 5 images originales de la séquence "Train"



(b) Détection de craquelures trame par trame



(c) Défauts d'épaisseur 2

FIG. 6.14 - *Détection de craquelures dans la séquence "Train"*

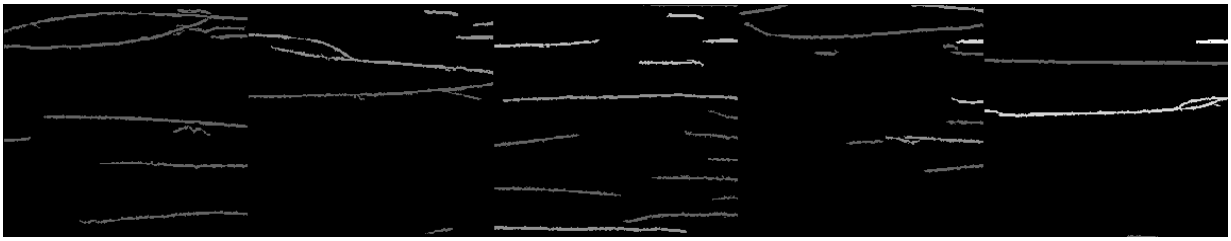


FIG. 6.15 - *Exemple de calcul de p_f. Le niveau de gris es proportionnel à la valeur au point considéré.*

peuvent être confondus avec des taches et où, en plus, le mouvement est rapide. Il s'agit de la séquence "football". Dans la figure 6.16 nous avons les trois premières images de cette séquence, et en dessous le résultat de la détection des taches blanches avec l'algorithme présenté dans la section 6.4.1.1, en prenant comme paramètres:

Paramètre	Valeur
S_{max}	100
S_{min}	16
g	40
g_{bas}	10

TAB. 6.4 - *Paramètres utilisés pour la détection des taches de la séquence "football".*

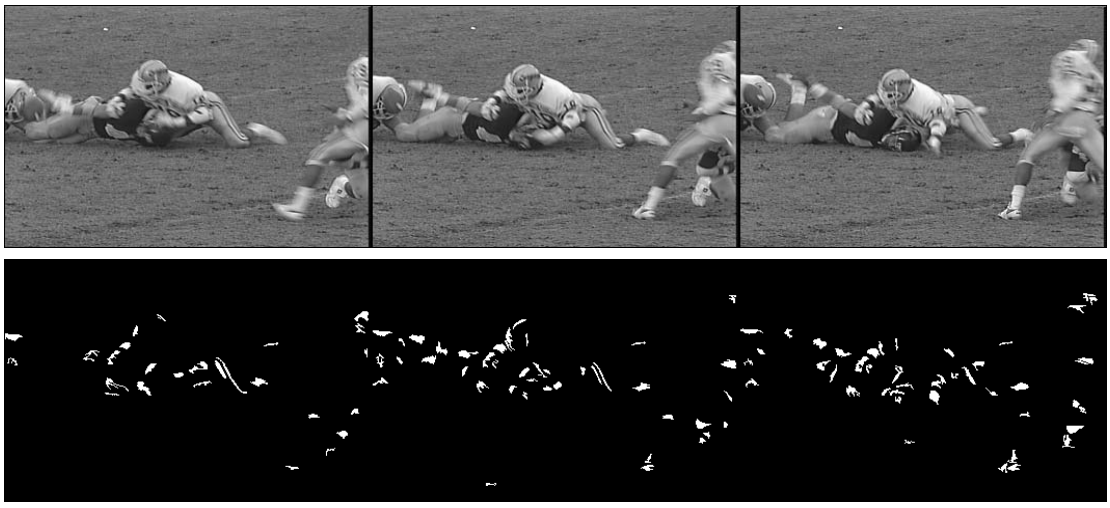


FIG. 6.16 - *Trois premières images de la séquence "football" et résultat de la détection trame par trame de taches blanches.*

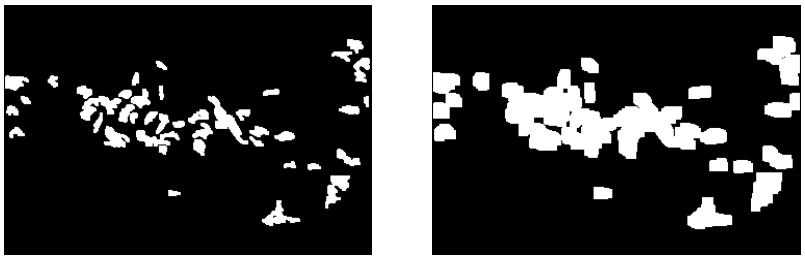
Si nous adoptons comme connexité temporelle $\delta_t = \delta_{t8}$ (voir section 2.4 pour les notations), alors les composantes connexes spatiales de la trame centrale considérées (de façon erronée) comme étant des défauts apparaissent sur la figure 6.17(a). Si nous choisissons une connexité temporelle plus forte, par exemple $\delta_t = \delta_{t8}^4$, le nombre de fausses détections diminue (voir la figure 6.17(b)).

Ceci dit, il y a un coût à payer. D'une part le temps de calcul est légèrement plus long, mais ce n'est là qu'un inconvénient mineur. D'autre part, et il s'agit du problème principal, des éventuels défauts risquent d'être connectés plus facilement à des objets de la scène lorsque la connexité spatiale est plus forte. Sur la figure 6.18 nous avons marqué tous les pixels de la trame centrale qui sont adjacents soit à un objet de cette même trame, soit

(a) Connexité temporelle: δ_{t8} (b) Connexité temporelle: δ_{t8}^4

FIG. 6.17 - *Composantes connexes maximales spatiales de la trame centrale de la séquence "football" dont la permanence temporelle est égale à 1.*

à un objet ou à un faux défaut des deux trames voisines. Si un vrai défaut apparaissant sur cette trame contenait un pixel de l'ensemble que nous venons de décrire, alors il viendrait à faire partie d'une composante connexe maximale de permanence supérieure strictement à 1, et par conséquent il échapperait à notre algorithme de détection. Remarquons que dans le cas de la connexité temporelle plus forte (6.18(b)) le risque de non-détection est plus élevé que dans l'autre cas.



(a) Connexité temporelle: δ_{t8} (b) Connexité temporelle: δ_{t8}^4

FIG. 6.18 - *Pixels adjacents aux objets et faux défauts des trames voisines, et aux objets de la trame courante.*

Par conséquent, en choisissant une connexité temporelle forte nous évitons de façon satisfaisante les fausses détections provoquées par les petits objets qui bougent rapidement. En contrepartie la détection de défauts est moins efficace.

6.4.2.3 Conclusion

La méthode de détection que nous venons de présenter est divisée en deux parties:

- la première, propre à chaque type de défaut, consiste à détecter trame par trame les objets qui, vues leurs caractéristiques géométriques, pourraient être des défauts.

– la deuxième consiste à analyser la séquence binaire obtenue pour distinguer les vrais défauts des fausses détections.

Cette méthode sera d'autant plus satisfaisante que les défauts seront faciles à caractériser. Si beaucoup d'objets de la scène sont pris pour des défauts l'algorithme aura du mal à faire le tri.

Par ailleurs il y aura toujours des objets de la scène qui seront à la limite d'être détectés en tant que défauts. Dans ce cas ils risquent d'être détectés dans certaines trames et pas sur d'autres, donnant ainsi l'impression qu'il s'agit d'accidents temporels et donc de défauts.

Dans la section suivante nous proposerons une deuxième approche où nous effectuons directement une analyse spatio-temporelle sur la séquence d'entrée. Pour cela il faudra généraliser le modèle de défauts locaux aléatoires à des images à niveaux de gris.

6.5 Deuxième méthode de détection

6.5.1 Modèle de défaut local aléatoire dans une séquence à niveaux de gris

Dans la section 6.3.4 nous avons donné un modèle de défaut aléatoire d'épaisseur n pour les séquences binaires: c'étaient les CCM de la séquence dont la permanence temporelle était inférieure ou égale à n. Désormais, nous les appellerons défauts locaux aléatoires binaires.

Comment pouvons-nous généraliser ce modèle aux images à niveaux de gris?

Nous adopterons la démarche déjà utilisée dans la section 2.3.4.2 qui consiste à considérer une image à niveaux de gris I comme un empilement d'images binaires B_g:

$$I = \sum_{g=1}^{G-1} B_g \tag{6.11}$$

où B_g est le seuil de I au niveau g: $B_g = X_g(I)$. Nous obtenons ainsi $G-1$ séquences binaires B_g, que nous pourrons analyser avec les outils déjà développés.

Définition 6.12 *Modèle de défaut local aléatoire clair dans une séquence à niveaux de gris.*
Avec les notations précédentes, nous dirons qu'un ensemble Ω est un défaut local aléatoire clair d'épaisseur n si et seulement si il existe un niveau de gris g tel que Ω soit un défaut local aléatoire binaire de même épaisseur de la séquence $B_g = X_g(I)$.

Définition 6.13 *Modèle de défaut local aléatoire obscur dans une séquence à niveaux de gris.*
Réciproquement Ω est un défaut obscur d'épaisseur n de la séquence I si et seulement si c'est un défaut clair de même épaisseur de la séquence inverse $G - I$.

6.5.2 Application

Supposons que nous cherchons à détecter les défauts clairs d'épaisseur n.

Soit γ l'ouverture par reconstruction qui efface toutes composantes connexes maximales C dont la permanence temporelle $p(C)$ est inférieure à n:

$$\gamma(B) = \bigcup\{C_i \in CCM(B) \ / \ p(C_i) > n\} \tag{6.12}$$

Comme toutes les ouvertures par reconstruction, γ est croissante.

L'ouverture par reconstruction résultante pour les images à niveaux de gris est:

$$\Gamma(I) = \sum_{g=1}^{G-1} \gamma(B_g) \tag{6.13}$$

Cette ouverture par reconstruction efface tous les défauts locaux aléatoires d'épaisseur n, et n'efface qu'eux.

La figure 6.19 montre un exemple. L'image (a) est la trame centrale de la série apparaissant sur la figure 6.14. Nous lui avons appliqué l'ouverture par reconstruction Γ en choisissant une permanence temporelle de 1. La connexité spatiale est de 4 et la connexité temporelle simple. Nous montrons sur l'image (b) la différence entre l'image originale et le résultat de l'ouverture. Remarquons que les défauts ressortent clairement, mais nous observons aussi des objets secondaires, qui ne nous intéressent pas.

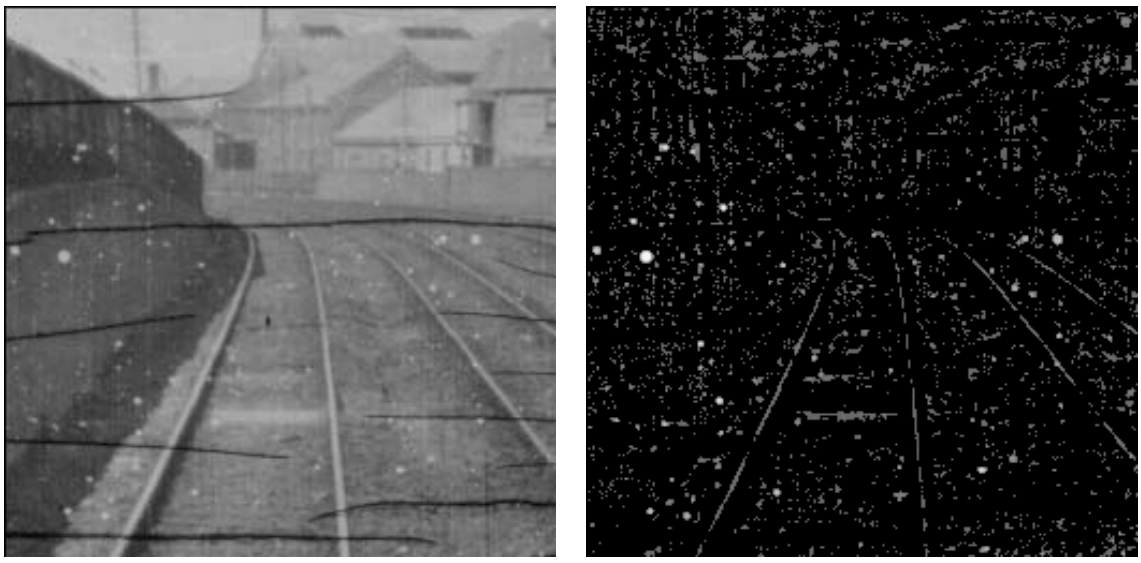
6.5.3 Post-traitement

Lorsque nous faisons la différence entre I et $\Gamma(I)$ nous obtenons une image à niveaux de gris I_2 indiquant tout ce qui a été détecté comme défaut aléatoire. En théorie il suffirait de la seuiller pour obtenir notre masque de défauts, mais en pratique elle contient de nombreuses détections sans intérêt, dues à du bruit ou à des déplacements.

Pour filtrer cette image nous utilisons un double seuillage [79]. Le procédé est le suivant. Nous choisissons un seuil bas g_1 et un seuil haut g_2. Nous seuillons I_2 avec ces deux seuils, obtenant respectivement $B_1 = X_{g_1}(I_2)$ et $B_2 = X_{g_2}(I_2)$, et nous reconstruisons B_1 en utilisant B_2 comme marqueur. L'image finale, B_3 correspond aux défauts que nous considérons comme importants.

Cet algorithme, appliqué à l'image 6.19(b) avec $g_1 = 2$ et $g_2 = 20$ donne le résultat de la figure 6.20. Nous avons réussi à faire ressortir les défaut, qui dans ce cas correspondent aux taches blanches. Mais deux bouts de rail ont aussi été détectés. Pourquoi? Pour deux raisons. Premièrement, à cause des vibrations de la séquence, les rails ont été effacés par l'ouverture par reconstruction Γ, et donc sont apparus sur l'image différence. D'autre part, des taches blanches bien contrastées touchaient ces bouts de rail, donc lors de la phase de reconstruction du double seuillage ils ont été mis en évidence.

Ce problème ne serait pas apparu si nous avions appliqué l'algorithme de correction des vibrations à cette séquence avant la détection des défauts aléatoires. Dans un cas plus général, pour éviter ces fausses détections lorsque la séquence est soumise à des mouvements importants, il faut une procédure de compensation de mouvement.



(a) Image originale (b) Détection de défauts clairs

FIG. 6.19 - *L'image (b) donne la différence entre l'image originale (a) et son ouverture par reconstruction* Γ *avec une permanence temporelle égale à 1. Afin de la rendre visible, nous avons rajouté 100 aux pixels dont le niveau de gris est compris entre 1 et le maximum.*

FIG. 6.20 - *Résultat du double seuillage*

6.5.4 Commentaires

Le modèle de défaut aléatoire que nous venons de présenter est très robuste tant que les déplacements dans la séquence ne sont pas plus grands que les objets qui se déplacent. Plus précisément, si un objet bouge si rapidement qu'il se déconnecte au cours du temps, alors il sera considéré comme un défaut aléatoire. Cette remarque n'est pas nouvelle: nous avons déjà eu l'occasion de la formuler dans les sections précédentes. Cependant, dans un grand nombre de cas ce critère est suffisant. Dans les autres cas nous pouvons réduire les fausses détections en utilisant un algorithme de compensation de mouvement, mais comme nous l'avons déjà mentionné, cette solution pose de nombreux autres problèmes.

Par ailleurs, l'utilisation des opérateurs par reconstruction dans les séquences d'image ouvre la porte à beaucoup d'autres applications. L'image différence est riche en information sur la dynamique de la séquence. Par exemple, dans les séquence où il n'y pas de défauts il permettent de détecter les mouvements importants.

Finalement, remarquons que le résultat de l'ouverture ou de la fermeture par reconstruction nous donne aussi une interpolation des régions abîmées par les défauts. Ceci dit, dans la section suivante nous verrons des méthodes de récupération de l'information perdue qui donnent des résultats plus réalistes.

6.6 Récupération de l'information perdue

Nous sommes arrivés au point où nous avons réussi à détecter avec précision les défauts locaux aléatoires d'une séquence d'images. Or, les défauts ont complètement effacé la texture qui se trouvait à leur emplacement. Il nous reste donc à récupérer cette information à partir d'interpolations. Comment pouvons-nous faire ceci?

Nous pouvons envisager deux approches. La première, et la plus simple, consiste à interpoler la texture qui se trouve autour du défaut à l'intérieur de celui-ci. La deuxième, plus complexe à mettre en œuvre mais au même temps plus intéressante, va chercher dans les autres trames de la séquence (en pratique la trame précédente) l'information perdue.

Nous avons étudié les deux possibilités.

6.6.1 Approche trame par trame

La première approche à la récupération d'information est l'interpolation purement spatiale. Plusieurs méthodes de ce type ont été développées dans le cadre de la restauration d'images 2D. Mais il ne faut pas oublier qu'ici nous considérons des défauts qui détruisent complètement l'information.

La difficulté du problème est très liée à la taille des défauts. Plus les défauts sont grands, plus l'information perdue est importante et plus les méthodes d'interpolation sont imprécises.

6.6.1.1 Très petits défauts: le bruit

Lorsque les défauts ne font que 1 ou 2 pixels la récupération est relativement simple. Ce type de défaut est rare dans les films, et de toute façon, étant donnée la résolution, très peu visible. Mais dans le cas des séquences vidéo il est courant, et assez gênant.

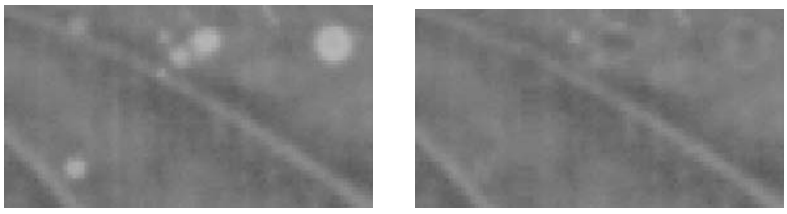
Un filtre médian classique avec un noyau de convolution 3×3 est largement suffisant pour la restauration. Sans oublier toutefois qu'il ne faut l'appliquer qu'aux pixels que nous avons identifiés comme étant des défauts!

6.6.1.2 Défauts moyens

Dès que la taille des défauts dépasse quelques pixels le besoin de méthodes d'interpolation plus fines se fait sentir.

Dans le cas des taches blanches, nous avons utilisé une méthode développée par Marc Van Droogenbroek [20]. Elle consiste à modéliser la texture à l'intérieur d'un rectangle entourant le défaut à partir de ses coefficients de Fourier, pour les interpoler ensuite à l'intérieur de la région endommagée.

La figure 6.21 montre un exemple d'application de cette méthode. Comme nous voyons, le résultat est très réaliste. Il est très difficile de retrouver la position des taches initiales à partir de l'image restaurée. Le seul inconvénient de cette méthode est son coût calculatoire.



(a) Image initiale (b) Image après restauration

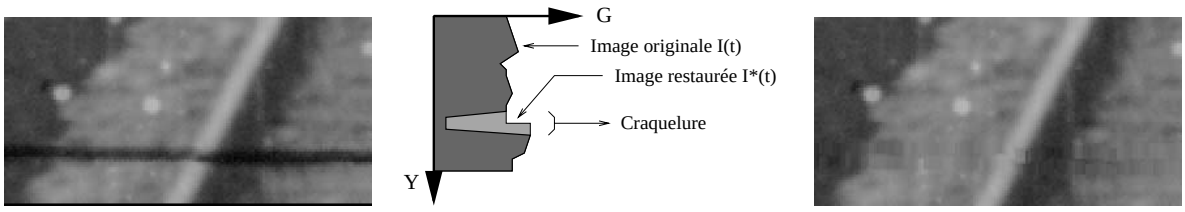
FIG. 6.21 - *Exemple de restauration de taches par interpolation de coefficients de Fourier*

6.6.1.3 Grands défauts

Lorsque les défauts ont touché des parties très importantes de l'image originale l'interpolation est très difficile. Dans certains cas on peut essayer d'appliquer des méthodes particulières à un type de défaut. Nous avons par exemple développé une méthode d'interpolation simple basée sur des dilatations verticales pour récupérer l'information perdue à cause des craquelures (cf. figure 6.22).

Le résultat n'est pas très satisfaisant: la texture résultante est lisse.

De toutes façons, lorsque le défaut est très grand, l'information perdue risque d'être irrécupérable par une approche purement spatiale. Par exemple si un détail a été complètement effacé il ne reste qu'un seul espoir: aller voir dans les autres trames de l'image si on arrive à retrouver les niveaux de gris manquants.



(a) Craquelure (b) Image après restauration

FIG. 6.22 - *Exemple de restauration d'une craquelure à partir de dilatations verticales*

6.6.2 Approche temporelle

Étant donné que nous travaillons avec des séquences d'images, il est assez naturel d'aller chercher dans les trames voisines ce qui manque sur la trame courante. En pratique, étant donné qu'au moment où nous restaurons la trame $I(t)$ nous supposons que la trame $I(t-1)$ a déjà été restaurée (que nous notons, rappelons-le, $I^*(t-1)$), nous irons chercher l'information manquante sur cette trame.

Remarquons que la méthode de détection donne un premier interpolateur temporel. C'est la différence entre l'image originale et cette interpolation qui nous avait permis de détecter les défauts. Cependant la restauration résultante n'est pas très belle.

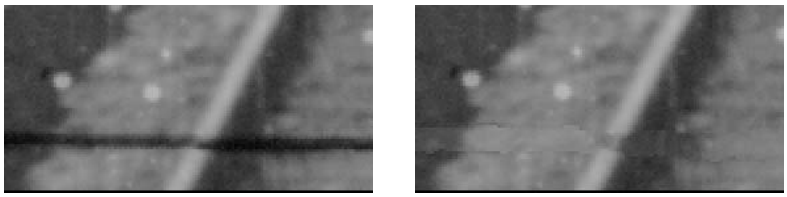
6.6.2.1 Le "couper-coller" de base

La première façon de faire, et la plus simple, est le "couper-coller" simple. Cette méthode peut être décrite ainsi: si nous ne connaissons pas le niveau de gris du pixel $P(x, y, t)$, nous allons chercher la valeur du pixel $P'(x, y, t-1)$, et nous la donnons à P. Cette méthode est basée sur l'hypothèse que la scène varie très peu au cours du temps.

Nous avons testé cette méthode dans le cas de la séquence "Train". Regardons ce que ça donne pour les craquelures (voir figure 6.23). Le résultat n'est pas idéal parce que l'hypothèse de non variation de la scène n'a pas été bien respectée. Ceci ressort d'une part au niveau de la frontière de la région restaurée: la discontinuité de la texture est très visible, et d'autre part au niveau de l'ombre du rail, qui a bougé.

6.6.2.2 Un "couper-coller" plus sophistiqué

Dans toute la phase de détection de défauts nous avons évité d'utiliser des algorithmes de compensation de mouvement. En effet, ces algorithmes sont délicats à mettre en œuvre et peu robustes. Or nous travaillons dans un environnement très bruité, raison pour laquelle nous avons choisi de nous baser uniquement sur la connexité temporelle pour la détection des défauts aléatoires. Mais maintenant que nous avons réussi à détecter les défauts de la trame $I(t)$, et que nous avons restauré la trame $I(t-1)$, nous pouvons appliquer la compensation de mouvement à la récupération d'information.



(a) Craquelure (b) Image après restauration

FIG. 6.23 - *Exemple de restauration d'une craquelure à partir d'une fermeture temporelle*

Afin d'améliorer la méthode précédente nous avons adapté la compensation par blocs (*blockmatching* en anglais [36]) à notre cas. Considérons le temps t. A partir de maintenant nous considérerons une connexité purement spatiale δ_s. Soit CC une CCM de $B(t)$ (c'est à dire un défaut de la trame t). Soit $R_{min}(CC)$ le plus petit rectangle contenant CC et $R_{min+m}(CC)$ le dilaté en 8-connexité de taille m de $R_{min}(CC)$:

$$R_{min+m}(CC) = \delta_{s8}^{m}(R_{min}(CC))$$

En d'autres mots $R_{min+m}(CC)$ est le plus petit rectangle contenant CC avec une marge d'épaisseur m (voir figure 6.24). Pour simplifier les notations nous poserons $R = R_{min+m}(CC)$.

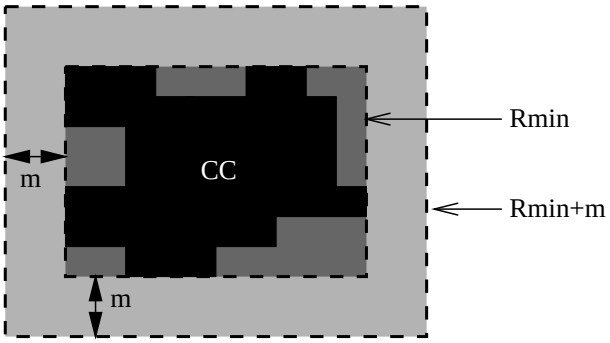


FIG. 6.24 - *Illustration de la définition du rectangle minimal contenant CC et du rectangle minimal avec une marge m contenant CC.*

Soit R' un rectangle appartenant à la trame $t-1$ de mêmes dimensions que R. Soit O_R (resp. $O_{R'}$) le premier pixel, dans l'ordre du balayage vidéo, de R (resp. de R'). Si nous changeons l'origine du repère et nous la mettons en O_R (resp. $O_{R'}$) nous obtenons une base locale à notre rectangle R (resp. R'). Nous appellerons pixel correspondant dans R' à un pixel P de R, le pixel dont les coordonnées spatiales dans le repère d'origine $O_{R'}$ sont les mêmes que les coordonnées spatiales de P dans le repère d'origine O_R. Nous noterons ce point P'.

Nous pouvons maintenant définir la différence quadratique entre les deux rectangles:

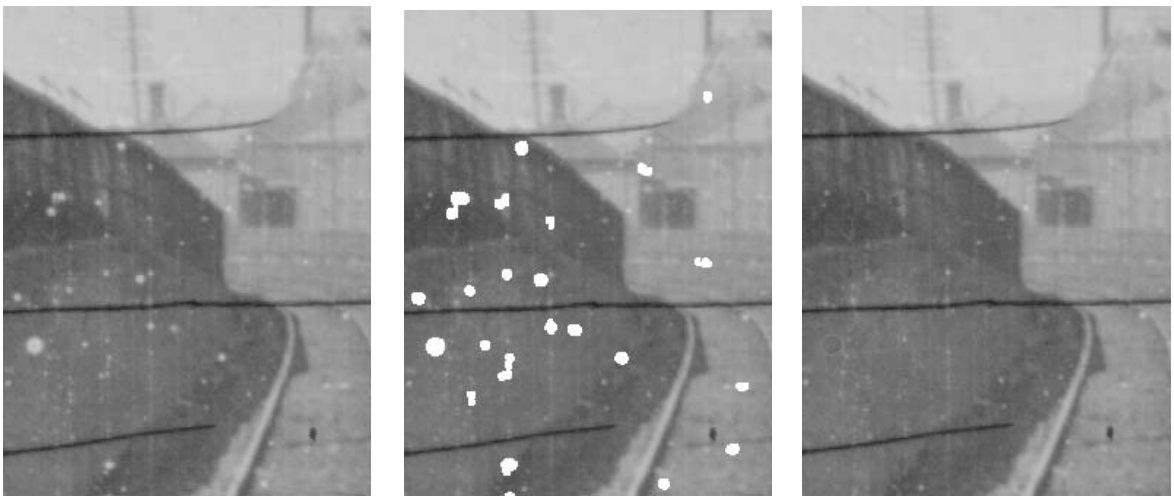
$$\Delta^2(R, R') = \sum_{P \in R-B} (I(P,t) - I^*(P', t-1))^2$$

Remarquons que nous ne prenons en compte que les pixels de R qui n'appartiennent pas à B dans le calcul de cette différence.

Forts de ces définitions nous pouvons maintenant décrire la nouvelle méthode de récupération d'information. Pour chaque composante connexe maximale CC de $B(t)$ (rappelons-nous que nous travaillons en connexité spatiale), nous calculons le rectangle minimal la contenant avec une marge m: $R = R_{min+m}(CC)$. Ensuite, nous cherchons dans le temps précédent $t-1$ le rectangle R' qui minimise $\Delta^2(R, R')$ (si nous en trouvons plusieurs nous prendrons le plus proche de R). Nous prenons le temps précédent parce que la version restaurée $I^*(t-1)$ de la trame originale $I(t-1)$ y est déjà connue. Finalement, à chaque pixel P de CC nous donnerons la valeur du pixel P' correspondant dans R': $I^*(P,t) = I^*(P', t-1)$.

En pratique il n'est pas nécessaire de parcourir toute la trame $t-1$ à la recherche du meilleur appariement pour R. Il suffit de tester des déplacements relativement petits.

Dans la figure 6.25 nous montrons un détail du résultat obtenu avec une trame de la séquence "Train". L'image (a) provient de la trame originale. L'image (b) correspond à la détection des taches blanches. L'image (c) montre le résultat de la récupération d'information avec le "couper-coller" sophistiqué. La marge utilisée est de 2 et le déplacement maximal permis pour R de 10.

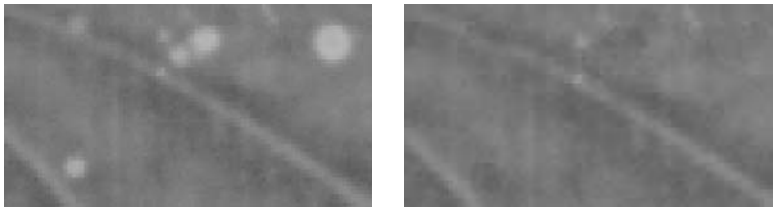


(a) Image originale (détail) (b) Détection de taches (c) Restauration

FIG. 6.25 - *Exemple de restauration de taches blanches avec un "couper-coller" sophistiqué.*

Dans la figure 6.26 nous avons un autre détail qui nous permet de comparer le résultat avec celui obtenu par interpolation de coefficients de Fourier (voir figure 6.21). La qualité subjective est comparable, tandis que le temps de calcul est bien inférieur. Ceci dit, comme

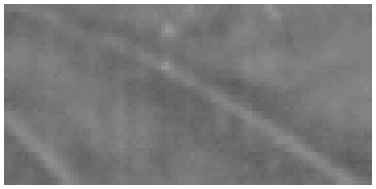
nous l'avons déjà noté, nous constatons une discontinuité de la texture aux bords des régions restaurées par la méthode du "couper-coller". Comment résoudre ce problème?



(a) Image originale (détail) (b) Restauration

FIG. 6.26 - *Deuxième exemple de restauration de taches blanches avec un "couper-coller" sophistiqué.*

La solution que nous proposons consiste à opérer une convolution avec un noyau 3×3 des points qui sont adjacents à la frontière des régions restaurées. La figure 6.27 montre le résultat. La discontinuité aux bords a disparu.



(a) Image restaurée

FIG. 6.27 - *Résultat de la restauration suivie d'un filtrage des bords pour rendre la texture continue.*

6.7 Conclusion

Dans ce chapitre nous avons appliqué les opérateurs par reconstruction aux séquences d'images, ce qui nous a permis de détecter de façon robuste les défauts locaux aléatoires.

Nous les avons mis en œuvre de deux façons distinctes, la première en les appliquant à des séquences binaires, et la deuxième à des séquences à niveaux de gris. Quelle méthode faut-il choisir? Tout dépend des connaissances à priori que nous avons sur les défauts que nous cherchons à détecter. Si nous sommes capables de mettre au points un algorithme de reconnaissance de formes suffisamment robuste pour détecter tous les défauts, tout en limitant le nombre de fausses détections à un taux raisonnable, alors la première méthode est plus intéressante. Sinon, et en particulier pour tous les cas où nous cherchons à effacer

tous les défauts indépendamment de leur géométrie, alors la deuxième méthode est de loin la meilleure.

Nous avons fait noter à plusieurs reprises que tous ces algorithmes de détection sont très robustes tant que le mouvement dans la séquence est faible. Cependant, dans certains cas, lorsque les déplacements sont importants, un algorithme de compensation de mouvement peut être nécessaire.

Nous avons aussi proposé plusieurs algorithmes pour interpoler l'information qui a été perdue à cause des défauts. Nous avons commencé des méthodes travaillant trame par trame, pour passer ensuite aux méthodes exploitant l'information se trouvant sur les trames voisines de la trame en cours de traitement. Remarquons que ces dernières méthodes ne sont pas plus complexes que les méthodes spatiales, et qu'elles donnent de très bons résultats.

Chapitre 7

Application du krigeage à la compensation de mouvement

7.1 Introduction

Dans le chapitre 6 nous avons fait un choix méthodologique: nous avons décidé de ne pas utiliser de compensation de mouvement dans nos algorithmes de restauration. Les deux raisons essentielles qui nous ont poussé dans cette direction sont les suivantes:

1. les algorithmes de compensation existants ne donnent pas encore des résultats tout à fait satisfaisants, d'autant plus que dans le cadre de notre application les images sont de mauvaise qualité,

2. nous voulions tester à fond les possibilités offertes par notre approche; le choix d'un algorithme particulier de compensation de mouvement aurait pu introduire un biais dans cette étude.

Depuis, nous avons montré que les algorithmes de restauration que nous avons développés marchent très bien. Cependant, lorsque le mouvement dans la scène est important, les critères de connexité que nous avons utilisés perdent de leur sens.

Nous allons donc maintenant nous occuper de cette phase de compensation de mouvement.

Nous commençons par décrire le krigeage, une méthode d'interpolation linéaire utilisée en géostatistique. Elle nous sert de base pour développer le krigeage inverse, qui nous permet de modéliser les textures douces efficacement.

Or justement les champs de vecteurs de déplacement varient par nature peu sur chaque objet d'une image. Nous appliquons donc le krigeage inverse aux champs de vecteurs de déplacement. Nous l'utilisons pour développer une méthode novatrice de compensation de mouvement.

7.2 Qu'est-ce que le krigeage?

7.2.1 Présentation

Le krigeage est un des outils principaux de la géostatistique. Il a été inventé par G. Matheron sous les trois formes de krigeage simple [50, 51], de krigeage universel [52] et des fonctions aléatoires intrinsèques d'ordre k (FAI-k) [54], et développé en grande partie par les chercheurs du Centre de Géostatistique de l'École des Mines de Paris. Il est utilisé pour interpoler des champs denses de données à partir de données isolées. Par exemple, supposons qu'on veuille dresser la carte des fonds sous-marins d'une baie. Il est pratiquement impossible d'effectuer toutes les mesures de profondeur nécessaires. Au lieu de cela, à partir d'un nombre limité de sondages, on interpolera la carte complète en utilisant le krigeage. La qualité du résultat dépendra des données initiales, ainsi que des hypothèse formulées sur la structure du fond.

La force de la théorie du krigeage est d'avoir évité le recours aux moments d'ordre 1 (moyenne) dans le cas du krigeage simple, puis d'ordre plus élevé dans les krigeages universel [52] et par FAI-k [54], dans l'estimation d'une fonction aléatoire. Par souci pédagogique nous ne présentons ci-dessous que le krigeage simple, bien que dans l'application qui suivra nous utiliserons, plus généralement, des FAI-k.

Une description plus générale pourra être trouvée dans [53, 8], et le lecteur voulant approfondir le sujet pourra se référer aux œuvres de Matheron [50, 51, 52, 54].

7.2.2 Krigeage simple

Le krigeage prend son point de départ dans la théorie des probabilités. La fonction inconnue $z : I\!R^2 \rightarrow I\!R$, que nous voulons estimer, n'est connue qu'aux **points d'échantillonnage** $\{P_i\}_{i=1\ldots n}$. Nous supposons que z est la réalisation d'une fonction aléatoire Z qui est stationnaire de second ordre. Ceci veut dire que:

1. Pour chaque point P, l'espérance de $Z(P)$ existe et ne dépend pas de P: $E(Z(P)) = m$.

2. Pour chaque point P et chaque vecteur $\vec{h}$ de $I\!R^2$ la covariance de Z existe et ne dépend pas de P:

$$cov(P, P + \vec{h}) = E(Z(P)Z(P + \vec{h})) - m^2 = cov(\vec{h}) \tag{7.1}$$

L'**estimateur** Z^* est une fonction aléatoire, obtenue par combinaison linéaire des valeurs de Z aux points d'échantillonnage $\{P_i\}_{i=1\ldots n}$:

$$\forall P : Z^*(P) = \sum_i \lambda_i(P) Z(P_i)$$

Nous devons maintenant calculer $\{\lambda_i(P)\}_{i=1\ldots n}$ pour chaque P où nous voulons estimer Z^*.

Nous voulons que l'espérance de l'erreur soit nulle, *i.e.* nous voulons un **estimateur sans biais**:

$$\forall P : E(Z^*(P) - Z(P)) = 0$$

et ceci indépendamment de la valeur de l'espérance m de Z, donc:

$$\sum_i \lambda_i(P) = 1$$

Dans un souci de clarté, nous écrirons désormais λ_i au lieu de $\lambda_i(P)$.

Le but de Z^* est d'être aussi proche que possible de Z; ceci veut dire, en termes probabilistes, que nous voulons minimiser la variance de l'erreur:

$$\forall P : Var(Z^*(P) - Z(P)) = \sum_i \sum_j \lambda_i \lambda_j cov(P_i, P_j) - 2\sum_i \lambda_i cov(P_i, P) + cov(P, P)$$

Ceci est un problème de minimisation sous contrainte des variables $\{\lambda_i\}_{i=1\dots n}$. La solution est donnée par le système:

$$\begin{cases} \forall j : \sum_i \lambda_i cov(P_i, P_j) + \mu = cov(P_i, P) \\ \sum_i \lambda_i = 1 \end{cases} \tag{7.2}$$

où μ est le Lagrangien multiplicatif. Ce système est linéaire, à $n+1$ inconnues et $n+1$ équations. Dans la majorité des cas il a une solution unique $\{\lambda_i\}_{i=1\dots n}$. L'**estimation** $z^*(P)$ de z au point P est donnée par:

$$z^*(P) = \sum_i \lambda_i z(P_i)$$

Nous pouvons donc calculer z^* en tout point, une fois la covariance $cov()$ choisie.

La variance de krigeage $\sigma^2()$, qui donne une mesure de la précision de l'estimation au point considéré, est donnée par:

$$\forall P : \sigma^2(P) = \sum_i \lambda_i cov(P, P_i) + \mu \tag{7.3}$$

Remarquons que cette variance dépend uniquement de la position des points d'échantillonage et du point P, et non pas des valeurs correspondantes.

Un des problèmes que les géostatisticiens doivent résoudre est le choix de la covariance de Z, qui doit modéliser au mieux la structure spatiale de la fonction à estimer z.

7.2.3 Complexité du krigeage

Le système linéaire précédent (7.2) peut être écrit sous la forme matricielle suivante:

$$\underbrace{\begin{pmatrix} cov(P_1,P_1) & cov(P_1,P_2) & \dots & cov(P_1,P_n) & 1 \\ \dots & & & & \dots \\ cov(P_n,P_1) & cov(P_n,P_2) & \dots & cov(P_n,P_n) & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix}}_{K} \underbrace{\begin{pmatrix} \lambda_1 \\ \dots \\ \lambda_n \\ \mu \end{pmatrix}}_{\Lambda} = \underbrace{\begin{pmatrix} cov(P_1,P) \\ \dots \\ cov(P_n,P) \\ 1 \end{pmatrix}}_{B}$$

L'estimation $z^*(P)$ de P est donnée par:

$$z^*(P) = \begin{pmatrix} \lambda_1 & \dots & \lambda_n & \mu \end{pmatrix} \underbrace{\begin{pmatrix} z(P_1) \\ \dots \\ z(P_n) \\ 0 \end{pmatrix}}_{z}$$

Apparemment, pour chaque point P nous devons calculer la matrice K du système linéaire, l'inverser, calculer Λ, et finalement calculer $z^*(P)$. Ceci peut prendre longtemps, surtout si nous considérons que nous pouvons avoir des centaines de points d'échantillonnage.

Voyons comment réduire le nombre d'opérations nécessaires. Les deux équations précédentes sont:

$$\begin{cases} K\Lambda = B \\ z^* = \Lambda^t z \end{cases}$$

donc:

$$z^* = B^t \underbrace{(K^{-1})^t z}_{C}$$

La matrice K et le vecteur z ne dépendent que des points d'échantillonnage, et en aucune façon du point à interpoler P. Par conséquent il suffit de les calculer une seule fois. Ensuite, pour chaque point P il suffit de calculer le vecteur B, *i.e.* les covariances $cov(P_i, P)$), et opérer le produit scalaire avec le vecteur C.

7.2.4 Propriétés

Parmi les multiples propriétés du krigeage nous n'en retiendrons que deux.

Premièrement, le krigeage est un estimateur **linéaire**: la valeur interpolée en chaque point est une combinaison linéaire des valeurs aux points d'échantillonnage.

Deuxièmement, le krigeage est un estimateur **exact**. Ceci veut dire que les valeurs interpolées aux points d'échantillonnage sont égales aux valeurs originales. En d'autre termes pour tout point d'échantillonnage P_i: $z^*(P_i) = z(P_i)$.

7.2.5 Modèle choisi

7.2.5.1 Modèle non-stationnaire

Dans notre présentation théorique du krigeage, nous avons supposé que la fonction aléatoire Z était stationnaire d'ordre 2. Nous avons trouvé ce modèle en pratique trop restrictif et nous avons opté pour un cadre plus général. Nous considérons donc que Z est le résultat de l'addition d'un polynôme avec une fonction aléatoire stationnaire d'ordre 2. Nous pourrions alors essayer de calculer ce polynôme et nous ramener ainsi au cas précédent. Au lieu de cela nous utilisons la théorie des fonctions aléatoires intrinsèques d'ordre k (FAI-k) [54],

qui nous permet de construire une estimation de la fonction sans séparation préalable de la composante polynômiale. En fait, en travaillant ainsi, nous filtrons les coefficients polynômiaux d'ordre inférieur ou égal à k. Tous les résultats de la section précédente restent vrais.

Nous avons choisi le modèle FAI-1. Le système linéaire correspondant est le suivant:

$$\begin{cases} \forall j : \sum_i \lambda_i cov(P_i, P_j) + \mu_1 x_j + \mu_2 y_j + \mu = cov(P_i, P) \\ \sum_i \lambda_i x_i = x \\ \sum_i \lambda_i y_i = y \\ \sum_i \lambda_i = 1 \end{cases}$$

où (x, y) sont les coordonnées de P, et (x_i, y_i) les coordonnées du point d'échantillonnage P_i.

Le modèle FAI-1 nous permet de choisir des types de covariance plus riches.

7.2.5.2 Covariance

Dans ce travail nous avons choisi comme covariance:

$$cov(P, Q) = |Q - P|^2 log(|Q - P|)$$

où $|Q - P|$ est la distance entre les points P and Q. Les interpolations obtenues sont des splines [56]. Ce modèle de covariance est connu sous le nom de **modèle de plaque mince** parce que le résultat correspond à la forme que prendrait une plaque parfaitement élastique qui passerait à travers les points d'échantillonnage. Nous l'avons choisi parce que justement il produit des surfaces lisses, qui ont un aspect naturel. Cette caractéristique nous sera très utile lorsque nous travaillerons avec des champs de vecteurs.

7.2.6 Exemple: application à la compression de texture

Dans le cadre du projet MORPHECO 2, nous avons appliqué le krigeage a la compression de texture [5, 7, 15]. Cet exemple nous permettra d'illustrer le fonctionnement du krigeage.

En compression (ou codage) d'images, le but est de réduire au minimum la quantité de données nécessaires pour reconstruire une image. L'image reconstruite ne doit par forcément être identique à l'image initiale (on parle alors de compression avec pertes, par opposition à la compression sans pertes), mais il faut préserver une certaine qualité minimale.

Qu'est-ce qui se passerait si nous extrayions un certain nombre de pixels de l'image originales (les points d'échantillonnage), pour ensuite leur appliquer le krigeage afin d'interpoler les valeurs de gris des autres pixels? Nous obtiendrions une estimation de l'image originale. Cette idée est illustrée par la figure 7.1.

Nous donnons un exemple pratique sur la figure 7.2. Nous avons codé la surface de la région lisse (a), qui a été obtenue grâce à un algorithme de segmentation. La taille de

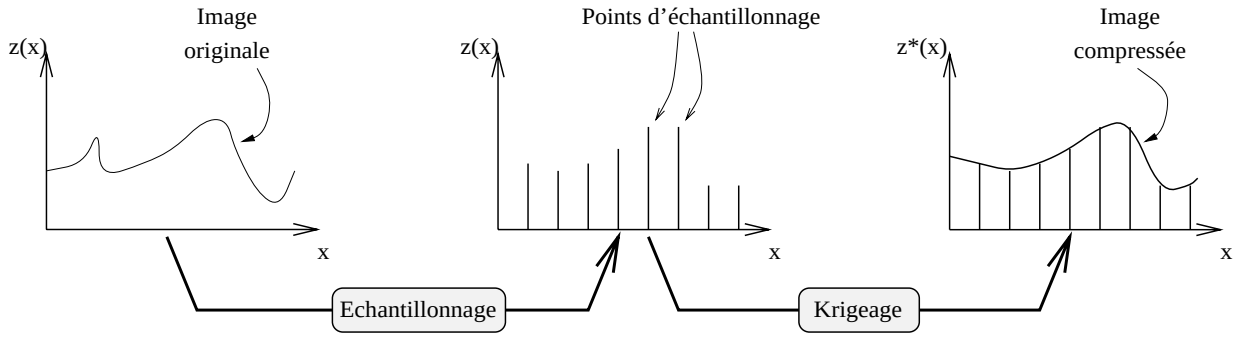


FIG. 7.1 - *Illustration de l'application du krigeage à la compression d'images.*

l'image est de 200 × 200, et la surface de la région est de 15361 pixels. Premièrement, nous avons choisi comme points d'échantillonnage les nœuds d'une grille carrée régulière de pas 8 qui appartiennent à la région d'intérêt. Ensuite, pour chaque pixel de cette région nous avons calculé grâce au krigeage une estimation de son niveau de gris. Comme nous pouvons le voir sur les images 7.2(b) et (b'), nous avons obtenu une bonne approximation de l'image initiale. L'erreur quadratique moyenne (EQM) entre la région interpolée et la région originale est de 17,1. Nous avons choisi une grille régulière de pas 8, donc le taux de compression est de 1/64 approximativement. Nous avons 242 points d'échantillonnage, nous n'avons pas besoin de coder leurs positions, et puisqu'il s'agit d'une image à 256 niveaux de gris, nous avons besoin de 242x8=1936 bits pour coder la texture de la région (en utilisant le codage de Huffman, nous pouvons descendre jusqu'à 170 octets, c'est à dire 1360 bits).

Comparons ces résultats avec ceux obtenus avec une des méthodes les plus utilisées pour la compression de texture: la transformée cosinus discrète adaptative [28, 78] (*Shape Adaptive Cosine Transform*, ou SADCT en anglais). Intéressons-nous aux images (c) et (c') de la figure 7.2, obtenues justement en compressant la texture avec la SADCT. L'EQM et le nombre de bits nécessaires sont respectivement 17,06 et 2658. Remarquons que, même si l'EQM est similaire, en fait l'effet de blocs introduit par la SADCT est très gênant. Pour des nombres de bits supérieurs la SADCT donne des meilleurs résultats en termes d' EQM, mais pour des nombres de bits inférieurs le krigeage est plus performant.

Il faut bien distinguer dans cette méthode de compression deux étapes:

1. Une étape d'échantillonnage qui, pour l'instant, est tout ce qu'il y a de plus classique. C'est à proprement parler la phase de compression.

2. Une étape d'interpolation, effectuée grâce au krigeage. C'est la phase de décompression.

Pour l'instant la phase d'échantillonnage est complètement indépendante de la phase d'interpolation. Nous pouvons envisager au moins deux façons de rendre plus efficace cette

(a) Image originale avec les 242 points d'échantillonnage

(b) Image obtenue après krigeage (EQM=17,1)

(c) Région codée avec SADCT (EQM=17,06)

(a') Détail de (a)

(b') Détail de (b)

(c') Détail de (c)

FIG. 7.2 - *Krigeage d'une région lisse; comparaison avec la SADCT*

première étape. La première consiste à choisir l'emplacement des points d'échantillonnage de façon plus intelligente, par exemple en les plaçant aux endroits caractéristiques de l'image: maxima, minima, cols, etc... Ces méthodes sont décrites dans [5, 7, 15]. La deuxième façon de faire consiste à prendre des positions fixes pour les points d'échantillonnage et à calculer leurs niveaux de gris de façon à ce que l'image interpolée à partir de ces valeurs soit aussi proche que possible de l'image initiale. On s'affranchit ainsi de la propriété d'exactitude du krigeage. C'est ce que nous avons appelé krigeage inverse, et que nous présentons dans la section suivante.

7.3 Le krigeage inverse

7.3.1 Présentation

Le krigeage nous permet, à partir d'un ensemble de points $\mathcal{E}$ où la valeur d'une fonction est connue, de calculer une estimation de cette fonction sur une ensemble $\mathcal{X}$. Nous avons vu dans la section précédente comment appliquer cet outil à la compression d'images.

Maintenant nous adoptons l'approche inverse. Pour l'ensemble de points donnés $\mathcal{E}$, nous calculons leurs niveaux de gris tels que, après krigeage de $\mathcal{X}$, l'interpolation résultante soit aussi proche que possible d'une image donnée sur $\mathcal{X}$. Du point de vue de l'interpolation de type cartographique, où l'on ne connaît les valeurs de la fonction à estimer qu'aux points d'échantillonnage $\mathcal{E}$, cette approche n'est ni pratiquement réalisable, ni intéressante. Mais notre point de vue est différent, puisque nous connaissons la fonction à estimer sur la totalité de l'ensemble $\mathcal{X}$ et voulons la remplacer par une autre, qui sera, elle, résumable à la donnée de (nouvelles) valeurs numériques aux points d'échantillonnage $\mathcal{E}$. Nous baptiserons ce procédé **krigeage inverse**.

L'ensemble de points $\mathcal{E}$ n'est donc pas réellement un ensemble de points d'échantillonnage, leurs niveaux de gris pouvant être différents des niveaux de gris de la fonction initiale aux mêmes points, mais nous avons conservé le même vocabulaire. Nous les appelons aussi **points d'échantillonnage optimal**.

7.3.2 Théorie

7.3.2.1 Définitions et vocabulaire

Soient $\mathcal{E}$ et $\mathcal{X}$ deux sous-ensembles finis de $I\!R^2$. Nous appellerons $\mathcal{E}$ **ensemble d'échantillonnage optimal** et $\mathcal{X}$ **ensemble de points expérimentaux**. Nous indexerons les points de $\mathcal{E}$ avec $I : \mathcal{E} = \{P_i\}_{i \in I}$.

Soit G_{exp} une fonction réelle définie sur $\mathcal{X}$:

$$\begin{array}{rccc} G_{exp} : & \mathcal{X} & \longrightarrow & I\!R \\ & M & \longmapsto & G_{exp}(M) \end{array}$$

G_{exp} correspond aux **valeurs de gris expérimentales**.

Soit G une fonction réelle définie sur $\mathcal{E}$:

$$\begin{array}{rccc} G: & \mathcal{E} & \longrightarrow & I\!R \\ & P_i & \longmapsto & G(P_i) \end{array}$$

Soit $\mathcal{K}$ l'opérateur de krigeage qui à G associe une fonction réelle $\mathcal{K}(G)$ définie sur $\mathcal{X}$ telle que, $\forall M \in \mathcal{X}$, la valeur $(\mathcal{K}(G))(M)$ est le résultat de l'interpolation par krigeage d'une valeur réelle pour M à partir de l'ensemble des points d'échantillonnage optimal $\mathcal{E}$ associés aux valeurs $G(N)$.

$$\begin{array}{rccc} \mathcal{K}: & (\mathcal{E} \rightarrow I\!R) & \longrightarrow & (\mathcal{X} \rightarrow I\!R) \\ & G & \longmapsto & G^* = \mathcal{K}(G) \end{array}$$

7.3.2.2 Problème

Munis des définitions de la section précédente, le problème peut être formulé de la façon suivante:

Quel G minimise la différence quadratique entre $G^* = \mathcal{K}(G)$ et G_{exp}?

7.3.2.3 Solution

Supposons qu'il existe une solution exacte G_0 à ce problème, *i.e.* que nous avons: $G_0^* = G_{exp}$. Ceci veut dire que $\forall M \in \mathcal{X}$ nous avons:

$$G_0^*(M) \quad = \quad G_{exp}(M) \tag{7.4}$$

c'est à dire que:

$$\sum_{i \in I} \lambda_i(M) G_0(P_i) \quad = \quad G_{exp}(M) \tag{7.5}$$

où $\lambda_i(M)$ est le poids de krigeage associé au point P_i lorsqu'on calcule $G_0^*(M)$.

Nous avons autant de ces équations que de points M dans $\mathcal{X}$, et autant d'inconnues $G_0(P_i)$ que de points N dans $\mathcal{E}$. Nous supposons que le nombre de points expérimentaux est plus grand que le nombre de points d'échantillonnage optimal et que les équations sont linéairement indépendantes. Alors le système linéaire est surdéterminé et il n'y a pas de solution exacte. En fait notre problème est une minimisation de l'erreur quadratique: la solution que nous cherchons est celle qui minimise le carré de la différence entre G_0^* et G_{exp}.

La décomposition SVD (*Singular Value Decomposition*) est la méthode de choix pour résoudre ce type de problèmes [68]. En l'appliquant à notre cas nous obtenons la solution qui minimise la différence quadratique mentionnée plus haut.

7.3.3 Exemple: application du krigeage inverse à la compression de texture

Dans le cas de la compression d'images, les différents paramètres du problème sont liés aux caractéristiques de l'image de la façon suivante:

- $\mathcal{E}$ est l'ensemble des points d'échantillonnage optimal, qu'on se donne. Dans notre cas ce seront les points d'une grille régulière.

- $\mathcal{X}$ correspond aux pixels de l'image dont nous voulons modéliser le niveau de gris. Les valeurs que nous avons appelées expérimentales sont les valeurs de gris initiales de l'image. Théoriquement il s'agit de tous les points de l'image, mais en pratique nous sommes limités par les capacités informatiques. Lorsqu'ils sont trop nombreux, nous en faisons un échantillonnage en utilisant là aussi une grille régulière. Toutefois, il faut prendre garde à ce que leur nombre reste supérieur au nombre de points d'échantillonnage optimal.

- G_{exp} correspond aux niveaux de gris de l'image.

- G_0 correspond aux niveaux de gris de l'ensemble des points d'échantillonnage optimal. Ce sont les valeurs que nous cherchons à calculer.

En pratique nous commençons par choisir un ensemble de points d'échantillonnage $\mathcal{E}$. Nous prenons simplement une grille carrée régulière car elle est très simple à coder: la seule information dont nous avons besoin est son pas. Le taux de compression de l'image ou de la région ne dépendra que de cette variable.

Ensuite nous choisissons l'ensemble des points expérimentaux $\mathcal{X}$. Si l'image ou la région concernées sont petites, nous pouvons en prendre tous les points; dans le cas contraire, nous opérons un échantillonnage régulier.

Finalement nous appliquons la procédure de krigeage inverse et nous obtenons les niveaux de gris des points d'échantillonnage optimal.

Il est important de noter que les niveaux de gris obtenus de cette façon sont les meilleurs, pour des ensembles $\mathcal{E}$ et $\mathcal{X}$, et un modèle de krigeage donnés.

La figure 7.3 illustre ce principe. Dans cet exemple nous avons pris tous les points du support comme points expérimentaux.

7.3.4 Résultats

Dans la section 7.2.6 nous avons vu que le krigeage à partir d'un ensemble de points d'échantillonnage donné par une grille régulière produisait de bons résultats. Par exemple, si la grille d'échantillonnage avait un pas de 8, l'EQM résultante était de 17,1 (voir figure 7.4(b)). Afin de comparer l'échantillonnage normal avec l'échantillonnage optimal, nous avons pris la même grille de pas 8 pour bâtir l'ensemble des points d'échantillonnage optimal, une grille carrée de pas 3 pour les points expérimentaux, et nous avons appliqué

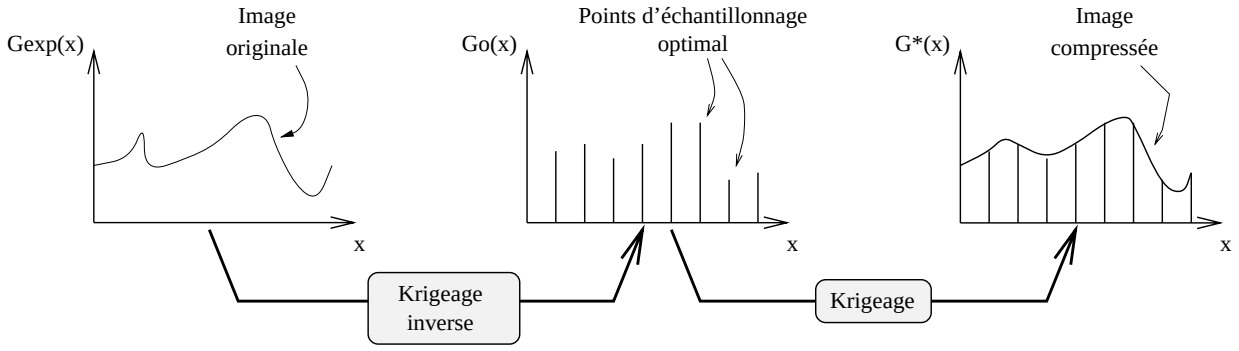


FIG. 7.3 - *Illustration de l'application du krigeage inverse à la compression d'images.*

le krigeage inverse pour calculer les niveaux de gris pour les points d'échantillonnage. L'image obtenue en interpolant à partir de l'ensemble de points d'échantillonnage optimal apparaît sur la figure 7.4(c). L'EQM est 11,5. La réduction de l'EQM montre que nous avons obtenu une amélioration de la qualité de l'image avec le même taux de compression.

(a) Image originale avec points d'échantillonnage $\mathcal{E}$

(b) Région compressée en utilisant l'échantillonnage simple (EQM=17,1)

(c) Région compressée en utilisant l'échantillonnage optimal (EQM=11,5)

FIG. 7.4 - *Comparaison d'images compressées avec krigeage et krigeage inverse*

Nous résumons dans le tableau 7.1 les résultats des tests effectués. Dans la ligne (a) nous indiquons l'EQM des images compressées lorsque nous utilisons un ensemble de points d'échantillonnage normal (pas d'optimisation). Dans les lignes (b), (c), (d) et (e) nous montrons les EQM obtenues avec le krigeage inverse en utilisant différents pas pour la grille des points expérimentaux.

Remarquons que lorsque la grille de points d'échantillonnage optimal et la grille de points expérimentaux sont identique, nous obtenons le même résultat qu'avec le krigeage

	Pas de la grille d'échantillonnage $\mathcal{E}$					
	4	8	12	16	20	24
a) Échantillonnage simple	11,2	17,1	24,7	37,9	33,2	61,3
b) Pas de la grille exp. = 8	-	17,1	18,9	26,3	26,0	39,3
c) Pas de la grille exp. = 5	-	13,3	15,5	24,2	24,9	38,3
d) Pas de la grille exp. = 4	11,2	11,9	16,8	23,9	24,5	38,0
e) Pas de la grille exp. = 3	11,3	11,5	16,5	23,5	23,5	37,7

TAB. 7.1 - *Comparaison de l'échantillonnage simple et de l'échantillonnage optimal pour le codage d'images. L'ensemble d'échantillonnage $\mathcal{E}$ et l'ensemble de points expérimentaux $\mathcal{X}$ sont donnés par des grilles carrées régulières. Les nombres indiquent l'EQM entre les images résultantes et l'image originale. Ligne (a): Résultats obtenus à partir d'un échantillonnage simple. Lignes (b,c,d,e): Résultats obtenus à partir d'un ensemble de points d'échantillonnage optimal.*

simple à partir de la même grille de points d'échantillonnage. Ceci est normal: quand nous calculons les niveaux de gris optimisés des points d'échantillonnage avec comme points expérimentaux les points d'échantillonnage, les niveaux de gris optimisés sont les niveaux de gris des points d'échantillonnage.

Remarquons aussi que la grille de points expérimentaux n'a pas besoin d'être particulièrement dense. Par exemple, pour un pas de 16 pour la grille de points d'échantillonnage (regardez la ligne correspondante dans le tableau 7.1) nous pouvons voir qu'avec une grille de points expérimentaux de pas 8 nous obtenons déjà une amélioration importante par rapport à l'échantillonnage simple, et que les grilles plus denses n'améliorent pas de façon considérable la qualité finale.

Finalement, rappelons que le coût de la compression, *i.e.* la quantité de données que nous devons transmettre, ne dépend pas de la taille de la grille de points expérimentaux, mais uniquement des points d'échantillonnage et de leurs niveaux de gris. Ceci dit, le temps de calcul nécessaire à la compression croît en fonction de la densité de cette grille.

7.3.5 Commentaires

Avec le krigeage inverse nous avons amélioré la qualité de l'image codée sans augmenter le coût du codage.

Notre expérience nous a montré que cette méthode est satisfaisante pour la compression de textures douces. Dès que nous l'appliquons à des textures très irrégulières, ou à des images entières comportant des discontinuités, le modèle s'avère incapable de les représenter. Tout se passe alors comme si on filtrait les hautes fréquences de l'image.

Le problème des discontinuités peut être résolu en découpant l'image en zones de texture homogène avant la compression, comme nous le montrons dans [15]. C'est le principe des systèmes de codage orientés objet.

Nous n'avons pas de solution pour le cas des textures irrégulières, si ce n'est de disposer

de plusieurs outils de compression de textures, qu'on appliquerait en fonction justement du type de texture. Par exemple si la texture est irrégulière on utilisera la SADCT, et si elle est lisse le krigeage. Ce type d'approche est adopté dans plusieurs systèmes de compression [10, 71].

Dans l'analyse des séquences d'images on s'intéresse souvent aux champs de vecteurs de mouvement, champs qui sont très réguliers. Cette constatation nous a poussés à appliquer les techniques que nous venons de décrire aux champs de vecteurs, comme nous verrons dans les sections suivantes.

7.4 Estimation et compensation de mouvement

Avant de nous attaquer à l'application du krigeage aux champs de vecteurs, rappelons rapidement quelques notions sur l'estimation et la compensation de mouvement.

Considérons une séquence d'images I. Les différences entre la trame I_t et la trame I_{t+1} peuvent être dues à trois causes:

1. Mouvement: les objets, ou toute la scène, sont susceptibles de se déplacer, ce qui peut entraîner la disparition ou l'apparition de certains détails.

2. Changement d'éclairage.

3. Bruit et défauts.

Estimer le mouvement entre la trame t et la trame $t+1$ consiste à donner un champ de vecteurs qui indique, pour chaque pixel de I_t, où est-ce qu'il se retrouvera au temps $t+1$. Ce champ de vecteurs est une image $V_{t\rightarrow t+1}$ (que nous noterons simplement V dans la suite) de même support $\mathcal{D}_t$ que I_t, et à valeurs dans $\mathbb{R}\times\mathbb{R}$. Un pixel P, de coordonnées (x,y), appartenant à I_t, se déplacera donc de $V(x,y)$ entre le temps t et le temps $t+1$, et se retrouvera par conséquent à la position $P+V(x,y)$. Théoriquement ce champ V existe: c'est la projection du mouvement réel de la scène tridimensionnelle. Mais la projection sur le plan 2D fait perdre beaucoup d'information, par conséquent la solution peut ne pas être unique. De plus, les changements d'éclairage, les ombres portées, le bruit, les recouvrements d'objets et les problèmes de limite de champ rendent le problème très complexe. Des centaines de chercheurs travaillent actuellement dans le domaine de l'estimation de mouvement pour les séquences d'images, comme en témoigne une littérature très riche; le lecteur peut se référer par exemple à: [4, 82, 19, 18, 33, 22, 27, 48, 89].

Supposons que nous avons estimé V et que pour tout point P de coordonnées (x,y) appartenant à $\mathcal{D}_t$, $P+V(x,y)$ appartient aussi à $\mathcal{D}_t$. La compensation de mouvement de I_{t+1} par rapport à I_t est l'opération qui consiste à déplacer les pixels de I_{t+1} de façon à ce que l'image résultante I^c_{t+1} soit aussi similaire que possible à I_t. L'image I^c_{t+1} est construite de la façon suivante:

$$\forall P \in \mathcal{D}_t : \; I^c_{t+1}(P) = I_{t+1}(P+V(P)) \tag{7.6}$$

Cette opération est appelée **compensation en arrière** de l'image I_{t+1} par rapport à l'image I_t. L'image résultante, I^c_{t+1}, est appelée image **compensée**. Elle correspond en quelque sorte à faire revenir dans le temps l'image I_{t+1}. Cette opération est très utilisée dans le domaine de la compression de séquences d'images [27, 48, 63, 36].

Dans la suite, toutes les compensations de mouvement que nous effectuerons seron faites "en arrière".

7.5 De l'interpolation de texture à l'interpolation de vecteurs

Soit V un champ de vecteurs, c'est à dire une image à valeurs dans $\mathbb{R} \times \mathbb{R}$. Supposons que V n'est connu qu'en un certain nombre de pixels $\{P_i\}_{i \in I}$. Soit P un autre pixel. Comment faire pour interpoler une valeur de vecteur pour P?

Nous avons choisi la solution la plus simple, qui en pratique s'est révélée satisfaisante. Pour chaque pixel P_i, le vecteur $V(P_i)$ du champ V a deux coordonnées: $V_x(P_i)$ et $V_y(P_i)$. Nous pouvons donc séparer notre champ de vecteurs en deux composantes, produisant ainsi deux images à valeurs dans $\mathbb{R}$: V_x et V_y. Nous appliquons le krigeage à ces images. Pour estimer $V(P)$, nous commençons par calculer $V^*_x(P)$ et $V^*_y(P)$ et ensuite nous prenons tout naturellement $V^*(P) = (V^*_x(P), V^*_y(P))$. Remarquons que, grâce au caractère linéaire du krigeage, le résultat de l'interpolation est indépendant du repère choisi.

Nous avons conçu une méthode de compensation de mouvement qui utilise le krigeage.

7.6 Une méthode de compensation de mouvement orientée objet

7.6.1 Structure générale de la méthode

Nous proposons une structure générale pour une méthode de compensation de mouvement basée sur une segmentation de l'image de référence.

Soient I_{t_1} et I_{t_2} deux trames d'une séquence. Nous voulons compenser I_{t_2} par rapport à I_{t_1}. La structure est la suivante:

1. Segmentation de l'image I_{t_1}. Nous obtenons une partition de l'image en régions S_i.

2. Choix des points expérimentaux et estimation de mouvement: dans chaque région S_i nous choisissons un certain nombre de points expérimentaux et nous calculons leurs vecteurs déplacement entre I_{t_1} et I_{t_2}.

3. Interpolation du champ de vecteurs: à l'intérieur de chaque région de la segmentation nous calculons pour chaque pixel une valeur de vecteur à partir des points expérimentaux appartenant à la même région.

4. Compensation de mouvement en utilisant le champ interpolé.

Les différentes étapes de cette méthode de compensation méritent quelques commentaires.

Segmentation de l'image: Les objets de l'image peuvent subir des déplacements différents. Idéalement, il faudrait analyser leur mouvement indépendamment les uns des autres, mais nous n'avons pas pour le moment les moyens d'obtenir une telle segmentation. Nous pouvons simplement espérer que celle que nous utilisons est plus fine que la segmentation idéale de l'image. Nous supposons donc que chaque région S_i est incluse dans un objet de l'image, et que par conséquent elle présente un mouvement cohérent.

Choix des points expérimentaux et estimation de mouvement: Le choix des points expérimentaux peut se faire indépendamment de la méthode d'estimation de mouvement utilisée, mais dans certains cas ce n'est pas possible. Ainsi, certaines méthodes d'estimation ne donnent des résultats fiables que pour certains pixels. Par ailleurs, il faut que chaque région de la segmentation contienne au moins un point expérimental.

Interpolation du champ de vecteurs: Tout l'intérêt de cette méthode réside dans cette phase. En effet, l'interpolation est faite à l'intérieur de chaque région de la segmentation où, d'après notre hypothèse, le mouvement est homogène. Nous disons qu'elle est **adaptative**. Nous limitons ainsi les problèmes de lissage aux bords. Évidemment, nous utilisons le krigeage pour effectuer l'interpolation. Nous obtenons en sortie un champ de vecteurs de déplacement dense.

Remarquons que la compensation par blocs, qui est la méthode la plus classique de compensation de mouvement utilisée dans le domaine de la compression de séquences, peut être décomposée dans le même nombre d'étapes: la segmentation de l'image est un découpage en blocs réguliers. L'estimation du mouvement se fait une fois pour chaque bloc, ce qui revient à calculer le déplacement du pixel central de chaque bloc, et à extrapoler ce vecteur à tous les autres pixels du même bloc. La grande différence par rapport à la structure que nous proposons est que la segmentation n'est pas liée au contenu de la scène. Par conséquent un bloc risque de tomber à cheval sur deux objets qui bougent différemment et produire une compensation inadéquate.

7.6.2 Mise en œuvre

Revenons à notre application et regardons point par point comment nous avons implémenté notre algorithme d'estimation de mouvement. Nous illustrons le fonctionnement de l'algorithme à l'aide de deux trames de la séquence "taxi", qu'on peut apprécier sur la

figure 7.5 (images (a) et (b)). Ces images ont une taille de 256×176 pixels. Remarquons que le déplacement des véhicules entre les deux temps est important.

1. Segmentation de l'image I_{t_1}: elle est effectuée à l'aide d'un algorithme de fusion de régions [58, 49]. Le résultat, pour la trame I_{t_1} apparaît sur la figure 7.5(d).

2. Choix des points expérimentaux et estimation de mouvement: les algorithmes de restauration que nous avons proposés dans les chapitres précédents fonctionnent correctement tant que le mouvement est faible. Donc nous cherchons à compenser surtout les déplacements importants; il faut que la méthode d'estimation de mouvement choisie soit capable de les mesurer. En contrepartie, grâce à la robustesse de nos algorithmes, la précision de l'estimation n'est pas primordiale: nous utilisons des vecteurs avec des coordonnées entières. Ces raisons nous ont amenés à choisir un algorithme d'estimation par blocs pour estimer le déplacement des points expérimentaux. Cet algorithme est peu robuste est risque de produire des valeurs aberrantes. Pour réduire l'influence de ces faux vecteurs, nous prenons un nombre suffisant de points expérimentaux à l'intérieur de chaque région S_i. En pratique nous avons choisi l'intersection d'une grille régulière de pas p et avec la région S_i, où les pas p dépend la surface de S_i. Les points expérimentaux, ainsi que les vecteurs de déplacement, apparaissent sur la figure 7.5 superposés d'abord à l'image initiale (c) et ensuite à la segmentation (d).

3. Interpolation du champ de vecteurs: tout naturellement nous utilisons le krigeage pour interpoler les vecteurs qui sont connus aux points expérimentaux.

4. Compensation de mouvement de I_{t_2} par rapport à I_{t_1} en utilisant le champ interpolé: la compensation résultante apparaît sur la figure 7.5(e).

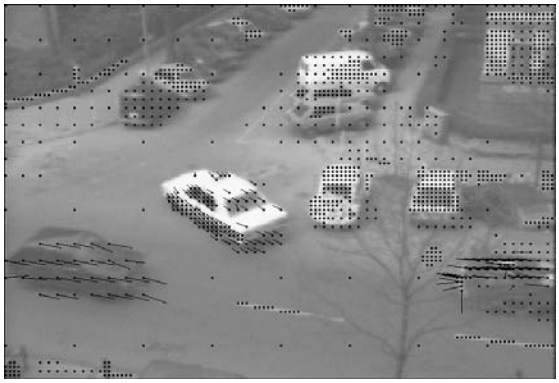
7.6.3 Commentaires

L'image compensée résultante est de bonne qualité. Remarquons cependant qu'à l'avant des véhicules en mouvement apparaît une zone où de toute évidence la compensation n'a pas produit le résultat escompté. Ceci est normal. En effet, ces zones correspondent aux parties de l'image I_{t_1} qui se retrouvent cachées au temps t_2. Donc il est normal que l'algorithme soit dérouté.

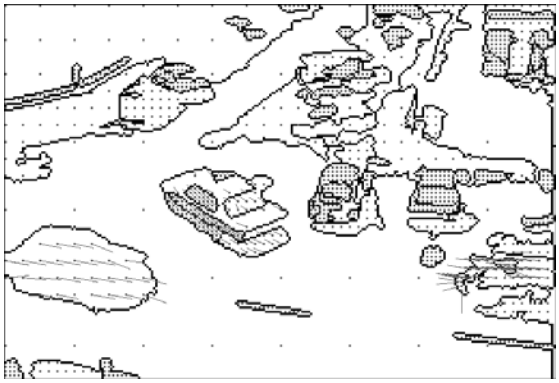
Par ailleurs certains vecteurs produits par la compensation sont incorrects. En particulier ceux qui correspondent à la camionnette mais aussi certains vecteurs sur la voiture de gauche. Nous appliquerons le krigeage inverse pour filtrer le champ de vecteurs et réduire ainsi l'effet des valeurs aberrantes.

(a) Trame I_{t_1}

(b) Trame I_{t_2}



(c) Champ de vecteurs expérimentaux superposés à I_{t_1}



(d) Champ de vecteurs expérimentaux superposés à la segmentation de I_{t_1}

(e) Compensation de I_{t_2} par rapport à I_{t_1}

FIG. 7.5 - *Compensation de mouvement à partir d'une segmentation morphologique, d'un champ de vecteurs produit par un algorithme de block-matching et d'une interpolation par krigeage.*

7.7 Krigeage inverse pour les champs de vecteurs

Cette section ressemble beaucoup à la section 7.3.2, sauf que nous traitons des fonctions à valeurs vectorielles plutôt que des fonctions à valeurs réelles. Le pas, comme nous verrons, est facile à franchir. Nous traitons séparemment chaque composante des vecteurs.

Rappelons que nous cherchons à calculer les vecteurs à associer à des points, dits d'échantillonnage optimal, tels que le champ obtenu en interpolant à partir de ces valeurs soit aussi proche que possible d'un champ mesuré expérimentalement.

7.7.1 Définitions et vocabulaire

Soient $\mathcal{E}$ et $\mathcal{X}$ deux sous-ensembles finis de $I\!R^2$. Nous appellerons $\mathcal{E}$ *ensemble d'échantillonnage optimal* et $\mathcal{X}$ *ensemble de points expérimentaux*. Nous indexerons les points de $\mathcal{E}$ avec I : $\mathcal{E} = \{P_i\}_{i \in I}$.

Soit V_{exp} une fonction à valeurs dans $I\!R^2$ définie sur $\mathcal{X}$:

$$\begin{array}{rccc} V_{exp}: & \mathcal{X} & \longrightarrow & I\!R^2 \\ & M & \longmapsto & V_{exp}(M) \end{array}$$

V_{exp} correspond aux **vecteurs expérimentaux**.

Soit V une fonction à valeurs dans $I\!R^2$ définie sur $\mathcal{E}$:

$$\begin{array}{rccc} V: & \mathcal{E} & \longrightarrow & I\!R^2 \\ & P_i & \longmapsto & V(P_i) \end{array}$$

Soit $\mathcal{K}$ l'opérateur de krigeage qui à V associe la fonction à valeurs dans $I\!R^2$ $\mathcal{K}(V)$ définie sur $\mathcal{X}$ telle que, $\forall M \in \mathcal{X}$, la valeur $(\mathcal{K}(V))(M)$ est le résultat de l'interpolation par krigeage d'une valeur de vecteur pour M à partir de l'ensemble des points d'échantillonnage optimal $\mathcal{E}$ associés aux valeurs $V(N)$.

$$\begin{array}{rccc} \mathcal{K}: & (\mathcal{E} \rightarrow I\!R^2) & \longrightarrow & (\mathcal{X} \rightarrow I\!R^2) \\ & V & \longmapsto & V^* = \mathcal{K}(V) \end{array}$$

7.7.2 Problème

Munis des définitions de la section précédente, le problème peut être formulé de la façon suivante:

Quel V minimise la différence quadratique entre $V^* = \mathcal{K}(V)$ et V_{exp}?

7.7.3 Solution

Supposons qu'il existe une solution exacte V_0 à ce problème, *i.e.* que nous avons: $V_0^* = V_{exp}$. Ceci veut dire que pour tout point M appartenant à $\mathcal{X}$ nous avons:

$$V_0^*(M) = V_{exp}(M) \tag{7.7}$$

$$\sum_{i \in I} \lambda_i(M) V_0(P_i) \quad = \quad V_{exp}(M) \tag{7.8}$$

où $\lambda_i(M)$ est le poids de krigeage associé au point P_i lorsqu'on calcule $V_0^*(M)$.

Nous avons autant de ces équations que de points M dans $\mathcal{X}$, et autant d'inconnues $V_0(P_i)$ que de points P dans $\mathcal{E}$. Nous supposons que le nombre de points expérimentaux est plus grand que le nombre de points d'échantillonnage optimal et que les équations sont linéairement indépendantes. Alors le système linéaire est surdéterminé et il n'y a pas de solution exacte. En fait notre problème est une minimisation de l'erreur quadratique: la solution que nous cherchons est celle qui minimise le carré de la différence entre V_0^* et V_{exp}.

Dans la section 7.3.2 nous avions utilisé la décomposition SVD pour résoudre ce problème de minimisation. Mais ici nous avons un système composé de vecteurs, qui cache en fait deux systèmes linéaires scalaires: celui des abscisses des vecteurs et celui des ordonnées. Mais étant donné que les deux coordonnées des vecteurs sont indépendantes, nous pouvons résoudre à part chacun des deux problèmes de minimisation, obtenant ainsi d'une part les abscisses des vecteurs $V_0(P_i)$ et d'autres part les ordonnées, pour les regrouper ensuite.

7.8 Filtrage de champs de vecteurs

Comment pouvons-nous faire pour appliquer l'optimisation décrite ci-dessus à notre algorithme de compensation de mouvement?

Premièrement, l'application sera adaptative, c'est à dire que nous effectuerons l'optimisation indépendamment pour chaque région de la segmentation.

Deuxièmement, les points expérimentaux sont déjà connus: ce sont évidemment les points où nous avons estimé le déplacement.

La seule question qui n'a pas encore été résolue est le choix des points d'échantillonnage optimal. Leur position n'a pas grande importance, mais leur nombre donnera la complexité du modèle de déplacement possible. Ainsi si nous prenons trois points non alignés nous obtiendrons un modèle de déplacement affine; avec d'avantage de points le mouvement peut être plus complexe, mais nous ne voulons pas non plus modéliser les défauts du champ de vecteurs. Nous avons choisi comme compromis quatre points d'échantillonnage, placés aux quatre coins de l'image.

En résumé, pour chaque région de la segmentation, nous modélisons son mouvement, donné par les vecteurs calculés aux points expérimentaux, grâce au krigeage inverse en utilisant comme points d'échantillonnage optimal les quatre coins de l'image. Nous obtenons ainsi quatre vecteurs, que nous utilisons pour interpoler un champ dense sur toute la région courante grâce au krigeage.

Cette optimisation, appliquée au cas présenté dans la section 7.6.2, donne le champ présenté sur la figure 7.6(b). Rappelons qu'il s'agit d'un champ dense; nous n'avons dessiné

que les vecteurs correspondant aux points expérimentaux, pour pouvoir le comparer avec le champ expérimental donné par la figure 7.6(a). Ces champs ont été superposés à l'image I_{t_1}. Nous les avons aussi superposés à la segmentation (cf. figure 7.7), pour mettre en évidence le caractère adaptatif du filtrage.

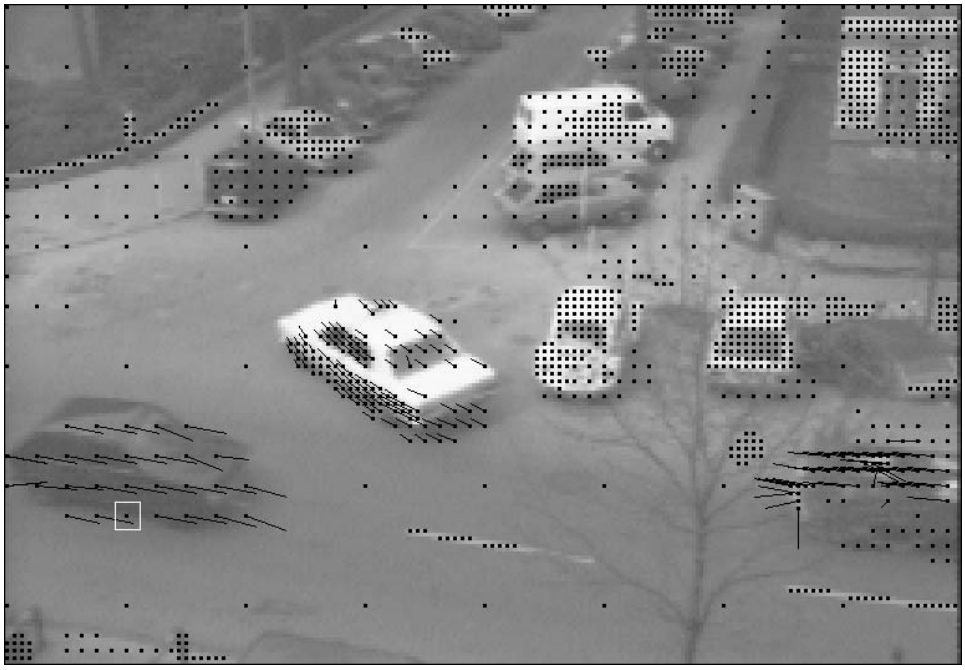
La première impression globale est que le champ optimisé est plus régulier que le champ expérimental. Cependant, nous n'observons pas de lissage de ce champ lorsque nous passons d'un objet en mouvement à un autre objet.

Par exemple sur le champ expérimental (figure 7.6(a)) un vecteur de la voiture de gauche était complètement erroné (nous l'avons entouré d'un rectangle blanc): son estimation avait donné un vecteur nul. Après filtrage l'erreur a disparu (figure 7.6(b)).

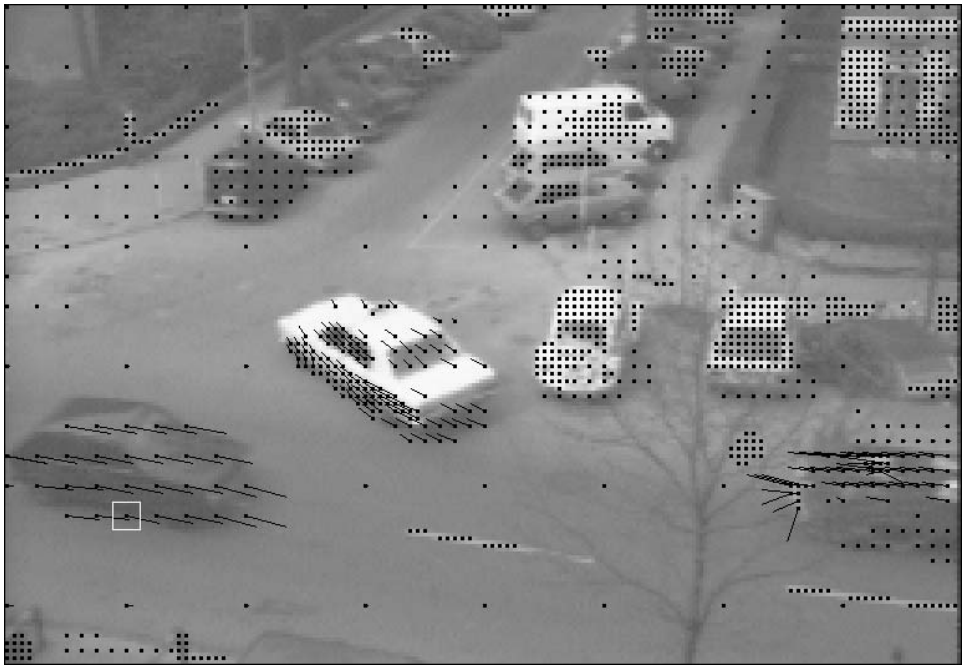
Ce traitement est beaucoup moins efficace pour le véhicule de droite parce que la segmentation était incorrecte. Notre hypothèse initiale, à savoir que chaque région de la segmentation était incluse dans un objet en mouvement, n'est pas vérifiée dans ce cas.

La figure 7.8 permet de comparer le résultat de la compensation sans optimisation avec celui de la compensation avec optimisation. L'amélioration apparaît dans deux régions: le bord gauche de l'image et le véhicule de droite.

Voyons maintenant comment appliquer cette méthode de compensation à notre problème de restauration.

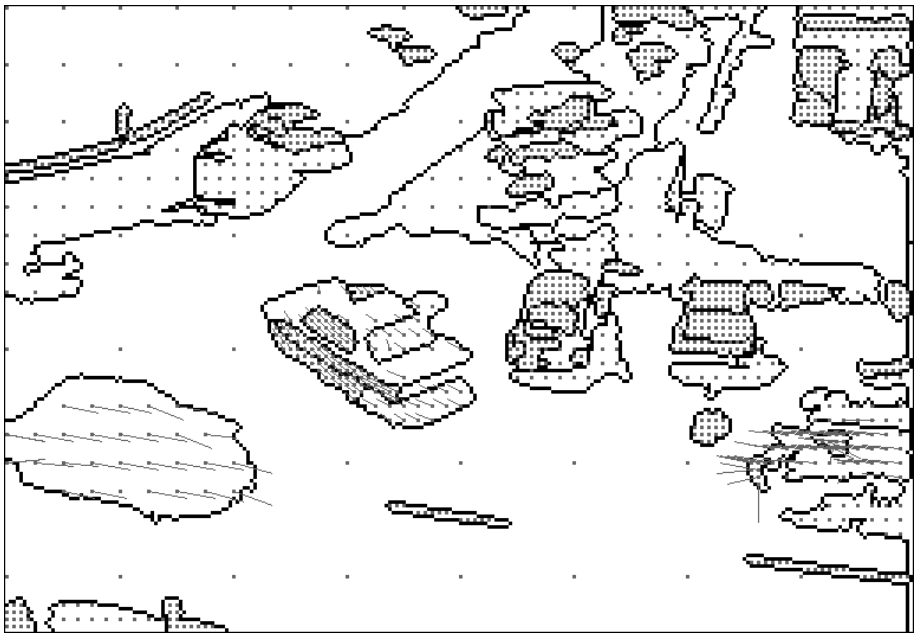


(a) Champ de vecteurs expérimentaux superposé à I_{t_1}

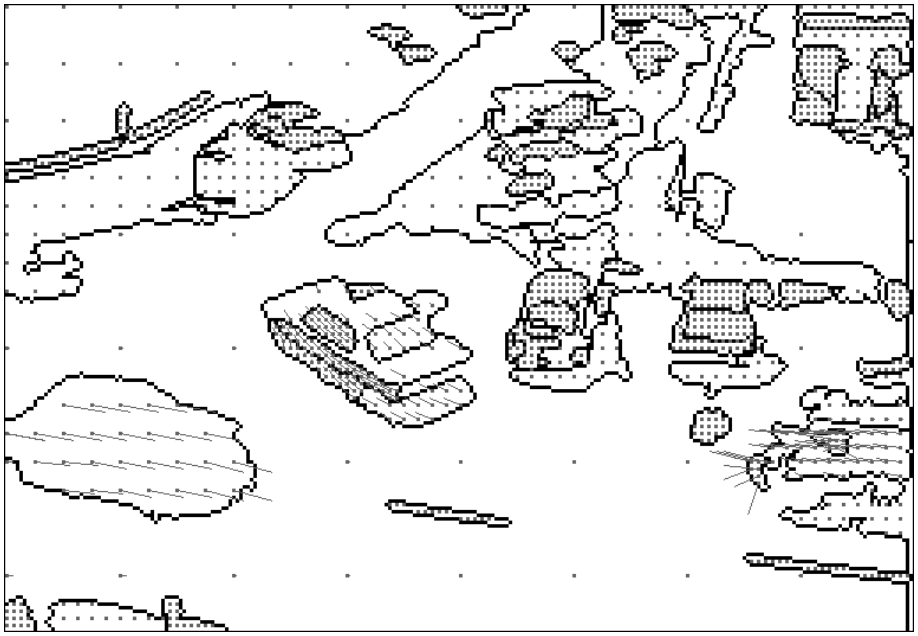


(b) Champ de vecteurs optimisés superposé à I_{t_1}

FIG. 7.6 - *Comparaison du champ de vecteurs expérimental avec le champ de vecteurs optimisé par krigeage inverse*



(a) Champ de vecteurs expérimentaux superposé à la segmentation de I_{t_1}



(b) Champ de vecteurs optimisés superposé à la segmentation de I_{t_1}

FIG. 7.7 - *Comparaison du champ de vecteurs expérimental avec le champ de vecteurs optimisé par krigeage inverse*

(a) Compensation sans optimisation

(b) Compensation avec optimisation

FIG. 7.8 - *Comparaison de la compensation de mouvement obtenue à partir du champ expérimental avec celle obtenue avec le champ optimisé*

7.9 Application à la restauration

Parmi les algorithmes de restauration que nous avons présentés, les seuls qui peuvent éventuellement avoir besoin d'une phase de compensation de mouvement sont ceux destinés à la détection des défauts locaux aléatoires. En effet, nous avions remarqué que le critère de connexité que nous utilisons donne de bons résultats tant que les objets de la scène ne se déconnectent pas au cours du temps. L'algorithme de compensation nous permet d'améliorer la corrélation des trames consécutives, et en particulier d'annuler les déplacements qui peuvent arriver entre deux temps de la séquence.

Soit I_t la trame que nous voulons traiter. Rappelons que pour détecter les défauts locaux aléatoires (clairs ou obscurs) d'épaisseur n, nous devons construire une image tridimensionnelle de travail W avec les $2n+1$ trames: $I_{t-n}, \ldots I_t, \ldots I_{t+n}$. Normalement nous appliquons à cette image tridimensionnelle nos algorithmes pour détecter les défauts de la trame centrale I_t. Maintenant, avant d'opérer la détection, nous appliquons la compensation de mouvement. Pour chaque trame I_{t+p} (avec p compris entre $-n$ et n, différent de 0) nous effectuon une compensation de mouvement par rapport à la trame I_t, obtenant une trame compensée I^c_{t+p}. L'image tridimensionnelle W est donc maintenant constituée des trames $I^c_{t-n}, \ldots I^c_{t-1}, I_t, I^c_{t+1}, \ldots I^c_{t+n}$.

Nous donnons un exemple d'application dans les figures suivantes. Nous cherchons à détecter les défaut aléatoires obscurs d'épaisseur 1 d'une séquence endommagée en présence de mouvement important. Nous montrons les trames I_{t-1}, I_t et I_{t+1} dans la figure 7.9(a). Il s'agit d'un extrait de dimensions 192×288 d'un film ancien. La détection des défauts obscurs d'épaisseur 1 sans compensation de mouvement apparaît en dessous (b). Remarquons que nous avons détecté non seulement les défauts sombres (en blanc, encerclés), mais aussi des petits objets de la scène qui ne sont pas des défauts (en blanc, encadrés).

Le résultat de la compensation de mouvement appliqué aux trames I_{t-1} et I_{t+1} apparaît sur la figure 7.10(a). La qualité des images compensées n'est pas très bonne, mais nous avons amélioré la corrélation temporelle des trames. Ainsi le prouve le résultat de l'application du même algorithme de détection des défauts locaux aléatoires obscurs (figure 7.10(b)). Nous avons diminué le nombre de fausses détections.

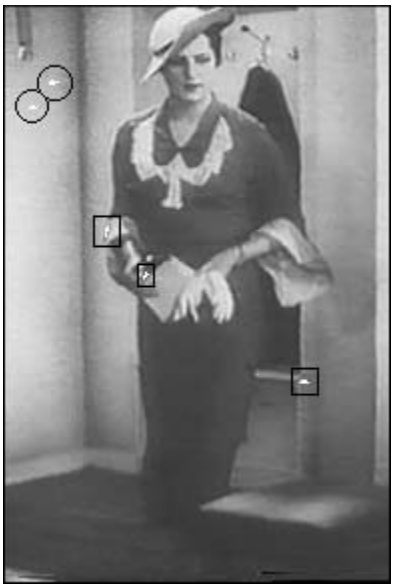
Nous avons donc réduit les problèmes provoqués par le déplacement des objets de petites dimensions.

(a)

(b)

FIG. 7.9 - *Détection de défauts sombres sans compensation de mouvement. (a) Trames originales I_{t-1}, I_t et I_{t+1}. (b) Résultat de la détection des défauts sombres (en blanc) superposé à la trame originale I_t. Les vrais défauts sont encerclés; les faux défauts sont encadrés.*

(a)



(b)

FIG. 7.10 - *Détection de défauts sombres avec compensation de mouvement. (a) Trames compensées I^c_{t-1} et I^c_{t+1} autour de la trame originale I_t. (b) Résultat de la détection des défauts sombres après compensation de mouvement (en blanc) superposé à la trame originale I_t. Les vrais défauts sont encerclés; les faux défauts sont encadrés.*

7.10 Conclusion

Dans ce chapitre nous avons commencé par présenter le krigeage. Nous avons montré qu'il permet d'interpoler des textures lisses de manière très satisfaisante. Ensuite nous avons développé une méthode d'échantillonnage optimal que nous avons baptisée krigeage inverse. Elle permet de modéliser les surfaces lisses.

Nous avons utilisé ces deux techniques pour bâtir un système de compensation de mouvement orienté objet. Nous l'avons mis en œuvre dans le cadre de notre application, ce qui nous a permis d'améliorer les résultats des algorithmes de détection de défauts aléatoires en présence de mouvement.

La structure générale de la méthode de compensation de mouvement est suffisamment souple pour pouvoir être utilisée dans le cadre d'autres applications, telles que la compression de séquences d'images.

Chapitre 8

Présentation du système global

8.1 Introduction

Dans les chapitres précédents nous avons présenté des algorithmes de restauration pour différents types de défauts rencontrés dans les films anciens:

- Pompage
- Vibrations
- Défauts locaux immobiles ou stationnaires (rayures)
- Défauts locaux aléatoires

L'essentiel de notre travail a porté sur le développement de ces algorithmes, ainsi que sur la mise au point d'une méthode de compensation de mouvement. Nous avons effectué des tests indépendants sur des séquences relativement courtes, au plus de quelques dizaines d'images. Les résultats étaient très prometteurs, mais afin de valider notre travail il faudra les appliquer à des films entiers, qui comportent des dizaines de milliers d'images: chacun des algorithmes développés est une bloc qui servira à construire un système complet de restauration automatique de films anciens.

Comment allons-nous agencer ces blocs afin que le système soit aussi efficace que possible, aussi bien du point de vue de la qualité de la restauration que des ressources informatiques?

Ce problème n'est pas trivial.

D'une part le fonctionnement des algorithmes de restauration sera influencé par le résultat des étapes précédentes. Par exemple on a intérêt à effectuer la correction d'histogramme avant la détection des défauts locaux aléatoires puisque, comme nous avons vu, la détection est basée sur une hypothèse de conservation des niveaux de gris. Mais d'autres choix sont moins évidents. Par exemple faut-il corriger les vibrations avant de traiter le problème du pompage?

D'autre part cette application requiert des ressources informatiques considérables. Par exemple si nous traitons des images vidéo d'une résolution de 720×576, avec une fréquence de 25Hz, il nous faut alors $720 \times 576 \times 25 \times 3600 \simeq 37$ Gigaoctets d'espace disque par heure de film. Et si nous prenons des images de travail constituées de 5 trames alors elles occuperont chacune $720 \times 576 \times 5 = 2,07$ Megaoctets de mémoire vive. Et ceci pour une résolution assez basse...

Dans ce chapitre nous proposons un système englobant les différents algorithmes développés de façon à optimiser le résultat de la restauration tout en essayant de limiter les besoins informatiques nécessaires. Nous avons baptisé ce système de restauration **SARSA** (Système Automatique de Restauration de Séquences Animées).

8.2 Traitement de la séquence

Rappelons que nous partons de l'hypothèse que la séquence à traiter ne comporte pas de coupes. Cette hypothèse est vraisemblable; des recherches sur la détection de coupes dans les séquences sont en cours.

Étant donné le nombre d'images à restaurer il n'est pas envisageable de traiter toute la séquence en même temps. Nous avons donc choisi une méthode itérative.

8.2.1 Boucle

A chaque itération nous restaurons une trame $I(t)$. Puisque la plupart des algorithmes de restauration que nous avons présentés effectuent une analyse temporelle dans le voisinage de la trame courante, nous adjoindrons à celle-ci T trames avant et T trames après, construisant ainsi une image de travail tridimensionnelle W constituée de $2T + 1$ trames, indexées par $-T, \ldots, 0, \ldots, T$ respectivement. Ceci est illustré par la figure 8.1. Nous verrons dans la suite comment choisir T en prenant en compte les besoins des différents algorithmes.

Nous aurions pu envisager de restaurer plusieurs images simultanément pour accélérer le processus, mais d'une part l'utilisation d'un éventuel algorithme de compensation de mouvement lors de la phase de récupération d'information serait alors très délicate, et d'autre part en opérant une trame à la fois nous sommes sûrs que la détection des défauts de la trame t bénéficiera de la restauration effectuée sur la trame $t - 1$.

Une fois $I(t)$ restaurée et sauvegardée, nous incrémentons le compteur t et nous recommençons.

8.2.2 Initialisation et finalisation

Étant donné le grand nombre de trames dans un film, l'initialisation (c'est à dire le traitement des T premières trames) et la finalisation (le traitement des T dernières trames) de la boucle sont des problèmes secondaires, mais que nous désirons traiter proprement. Nous verrons rapidement pour chaque algorithme comment faire.

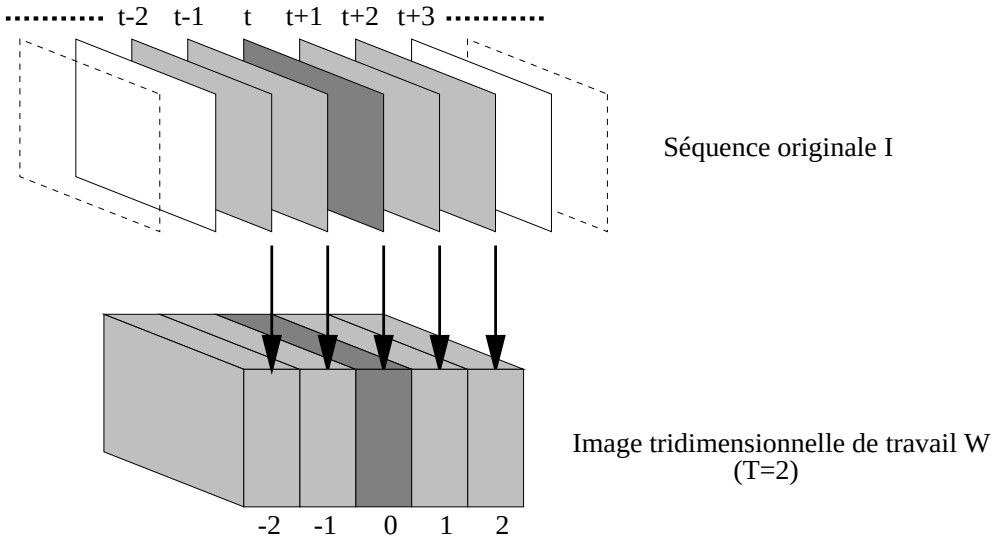


FIG. 8.1 - *Définition de l'image de travail W*

Correction du pompage La première image n'est pas traitée par cet algorithme: elle sert de référence. A partir de là on peut traiter les trames suivantes. Le traitement de la dernière trame de la séquence ne pose pas de problème.

Correction des vibrations La méthode de traitement des vibrations que nous avons présentée dans le chapitre 4 permet de traiter les trames initiales et finales sans problème pratique.

Correction des rayures L'algorithme de correction des rayures que nous avons développé travaille trame par trame, donc les problèmes d'initialisation et de finalisation ne se posent pas.

Correction des défauts locaux aléatoires Supposons que la première trame de la séquence originale est $I(0)$ et que nous traitons la trame $I(t)$ $(0 \leq t < T)$. Alors on enlève à l'image de travail W les $T - t$ trames: $W(-T), W(-T+1) \ldots W(-t-2)$, et on met dans $W(-t-1)$ une trame qui vaut partout le minimum si on cherche des défauts clairs, le maximum si on cherche des défauts sombres.

De façon similaire, lorsqu'on atteint les T dernières trames de la séquence il faudra enlever certaines trames de l'image de travail. Le processus est analogue à celui décrit plus haut.

8.2.3 Choix des paramètres de restauration

Dans l'état actuel des choses, il est nécessaire que l'opérateur du système choisisse les paramètres de restauration pour chaque séquence à traiter (éventuellement il peut opter pour un ensemble de paramètres par défaut). Dans ce sens, le système n'est pas entièrement automatique. Nous avons, malgré cette remarque, conservé l'adjectif *automatique* pour insister sur le fait que, une fois les paramètres choisis, les séquences sont entièrement traitées sans intervention extérieure.

Les paramètres les plus importants indiquent quels algorithmes de restauration il faut appliquer à une séquence d'images. Cependant, l'ordre de ces algorithmes est prédéfini, comme nous verrons dans la suite.

8.3 Comment ordonner les algorithmes de restauration?

8.3.1 Traitement séquentiel et traitement parallèle

Le titre de cette section : "Comment ordonner les algorithmes de restauration" sous-entend que nous avons préféré une approche séquentielle à une approche parallèle. Ceci est en grande partie vrai pour la simple raison que le résultat d'un algorithme produit des images qui sont théoriquement de meilleure qualité que les images d'entrée, ce qui devrait faciliter la tâche aux algorithmes suivants. Cependant, comme nous verrons dans la suite, cet avantage est très réduit dans certains cas (en particulier pour les algorithmes qui traitent les défauts locaux); on pourra alors envisager une parallélisation.

Par ailleurs, un **pipe-line** nous semble faisable. Il suffit d'assigner un processeur à chaque bloc de notre système.

Voyons donc dans quel ordre appliquer les algorithmes.

8.3.2 Classement des algorithmes

Soit I une séquence d'images et soient R_1 et R_2 deux algorithmes de restauration parmi ceux cités plus haut. La qualité de l'image $R_1(I)$ est meilleure que la qualité de l'image I, par conséquent, il est probable que la qualité de $R_2(R_1(I))$ sera meilleure que la qualité de l'image $R_2(I)$. Nous appellerons la différence entre ces deux qualités l'**apport** de l'algorithme R_1 à l'algorithme R_2.

Nous avons résumé ces apports dans le tableau 8.1. Comment lire ce tableau? L'intersection de la ligne correspondant au défaut A et de la colonne correspondant à B donne une estimation subjective de l'apport de la restauration de A à la restauration de B. Cette estimation peut valoir 0 (apport nul ou négligeable) ou 1 (apport important).

Justifions les valeurs que nous avons données à ces apports. L'analyse des défauts globaux (pompage et vibrations) est très peu influencée par la présence de défauts locaux (rayures et défauts aléatoires), vue la surface relativement petite que ces derniers occupent,

$R_1 \nearrow R_2$	Pompage	Vibrations	Rayures	Locaux aléatoires
Pompage	X	1	1	1
Vibrations	0	X	1	1
Rayures	0	0	X	1
Locaux aléatoires	0	0	1	X

TAB. 8.1 - *Apport de l'algorithme de restauration R_1 à l'algorithme de restauration R_2. 0: Apport nul ou négligeable. 1: Apport important.*

donc l'apport de la restauration des défauts locaux à la restauration des défauts globaux est très faible (nous évaluons l'apport à 0). Par contre, la restauration des défauts globaux améliore la cohérence temporelle de la séquence, donc simplifie le travail des algorithmes de détection de défauts locaux, d'où un apport important (1). L'histogramme de chaque image est insensible aux translations de celle-ci, par conséquent l'apport de l'algorithme de correction des vibrations à celui de restauration de l'effet de pompage est nul (0). L'apport dans l'autre sens n'est pas négligeable vu qu'il est plus facile de recaler des images avec des niveaux de gris cohérents (1). Finalement l'apport entre les algorithmes de restauration des défauts locaux est important (1) parce qu'ils fonctionnent mieux quand il y a moins d'artefacts dans l'image.

Notons que l'apport de R_1 à R_2 n'est pas nécessairement égal à l'apport de R_2 à R_1. C'est justement cette dissymétrie qui nous permettra d'établir un classement des algorithmes. En effet, si l'apport de R_1 à R_2 et supérieur à l'apport de R_2 à R_1 alors nous avons tout intérêt à utiliser R_1 avant R_2. Ainsi nous obtenons le classement suivant:

1. Correction de l'effet de pompage
2. Correction des vibrations
3. Restauration des défauts locaux

Ceci ne nous a pas permis de classer les algorithmes de traitement de défauts locaux entre eux. On pourrait envisager d'effectuer cette étape en parallèle, comme nous l'avions évoqué dans la section précédente. Nous avons cependant choisi d'effectuer en premier la restauration des rayures car il s'agit d'un algorithme plus simple et robuste, qui à notre avis sera moins influencé par la présence de défauts locaux aléatoires que l'algorithme de détection de ces défauts par la présence de rayures.

Nous obtenons donc finalement l'ordre suivant:

1. Correction de l'effet de pompage.
2. Recalage des trames.
3. Restauration des rayures.

4. Restauration des défauts locaux aléatoires.

Maintenant que nous avons ordonné nos blocs, voyons comment nous ferons passer l'information entre eux et comment nous structurerons la mémoire dont ils ont besoin.

8.4 Gestion des données

Chaque algorithme de restauration présenté a des besoins différents. Par exemple la correction de l'effet de pompage prend en entrée, en plus de la trame courante, les niveaux de gris minimal, maximal et moyen de la trame précédente, alors que les algorithmes de détection des défauts locaux ont besoin de plus d'information.

Par ailleurs le volume d'information à traiter est si gigantesque qu'il est peu probable qu'il tiendra en entier sur un disque dur. Par conséquent il faudra envisager le problème du stockage des données.

8.4.1 Mémoire vive

Parmi les algorithmes présentés, le plus gourmand en mémoire vive est celui de détection des défauts locaux aléatoires. Si nous cherchons les défauts aléatoires d'épaisseur n, alors l'algorithme a besoin de $T = n$ trames avant et après la trame courante. L'image de travail W sera construite à partir de la trame courante I_t flanquée de T trames de chaque côté:

$$\forall j \in \{-T, -T+1, \ldots 0, \ldots T\} \quad W(j) = I(t+j)$$

Tous nos algorithmes prendront comme entrée cette structure d'image, quitte à n'utiliser qu'une partie de celle-ci.

8.4.1.1 Description de l'image de travail

Le temps courant est t. Supposons donc que nous allons traiter la trame $I(t)$. Notre image de travail W est constituée des trames :

- $W(-T), W(-T+1), \ldots W(-1)$: ce sont les trames qui précèdent la trame à restaurer $W(t)$, et en tant que telles elles ont déjà été restaurées.
- $W(0)$: Trame courante à restaurer. Remarquons que son histogramme a déjà été corrigé.
- $W(1), \ldots W(T-1)$: trames "futures". Elles sont déjà passées à travers l'algorithme de restauration de l'histogramme.
- $W(T)$: Trame qui vient d'être insérée dans la structure. Elle est donc encore égale à $I(t+T)$. Son histogramme sera corrigé pendant le déroulement de cette boucle.

Voyons maintenant comment se déroulera la boucle.

8.4.1.2 Déroulement de la boucle

Correction de l'effet de pompage Cette étape n'affectera que la trame $t + T$ de l'image de travail. On n'a besoin que de la moyenne, du maximum et du minimum des valeurs de gris de la trame $I(T + t - 1)$ après correction de l'effet de pompage. Ces valeurs sont stockées respectivement dans les mémoires MOYENNE, MAXIMUM et MINIMUM. On applique donc l'algorithme décrit dans le chapitre 3 à $W(T)$, on met l'image résultante dans $W(T)$, et on met à jour les variables MOYENNE, MAXIMUM et MINIMUM avec les valeurs du nouveau $W(T)$. Ce processus est résumé dans la figure 8.2.

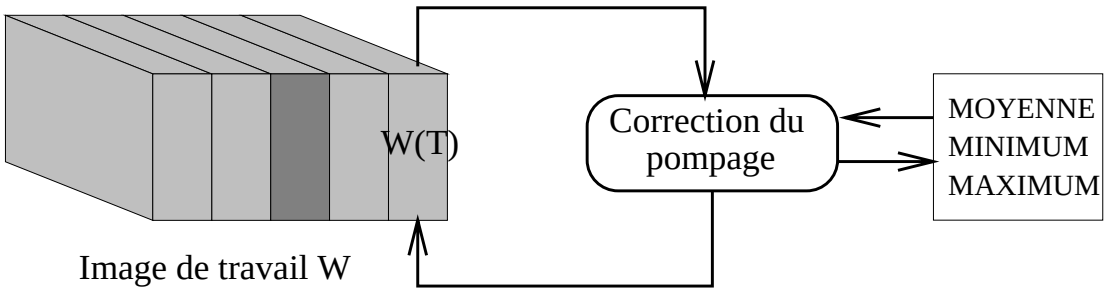


FIG. 8.2 - *Variables d'entrée et de sortie de l'algorithme de correction du pompage*

Correction des vibrations Les valeurs des vecteurs de translation $\vec{t}$ nécessaires au déroulement de l'algorithme décrit dans le chapitre 4 se trouvent dans la mémoire VECTEURS_DE_TRANSLATION. Donc en calculant le nouveau vecteur entre $W(T-1)$ et $W(T)$, et en utilisant les vecteurs en mémoire, on peut recaler la trame $W(0)$. On met le résultat dans $W(0)$. Ce processus est résumé dans la figure 8.3.

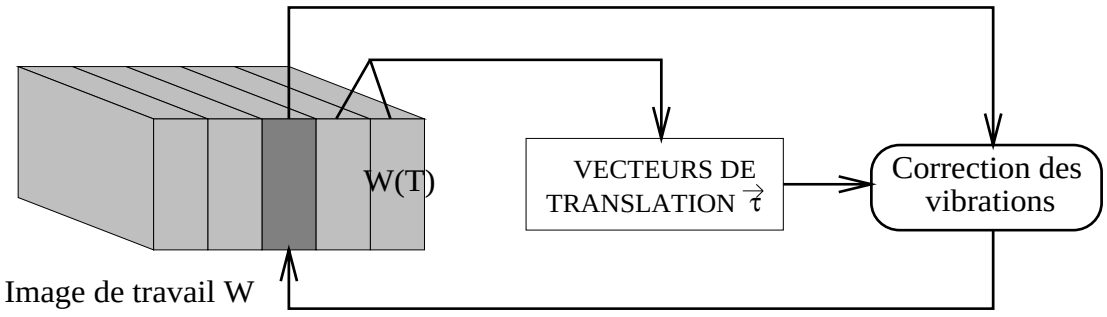


FIG. 8.3 - *Variables d'entrée et de sortie de l'algorithme de correction des vibrations*

Restauration des rayures Pour restaurer les rayures de la trame $W(0)$, il suffit de connaître cette même trame (voir figure 8.4).

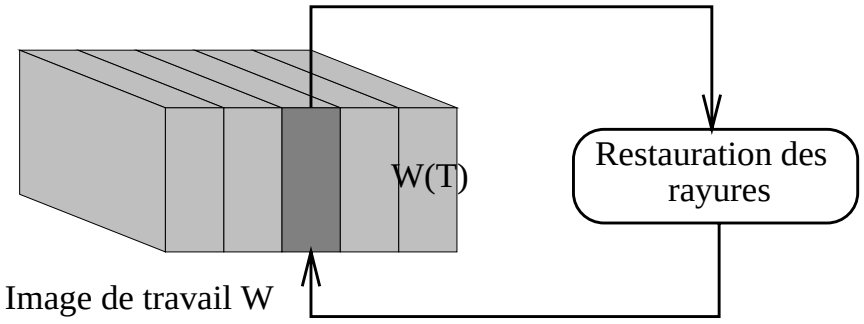


FIG. 8.4 - *Variables d'entrée et de sortie de l'algorithme de restauration de rayures*

Restauration des défauts locaux aléatoires On applique finalement l'algorithme de restauration des défauts aléatoires. La méthode de compensation de mouvement que nous avons développée intervient comme phase de pré-traitement dans ce bloc. La version restaurée ainsi obtenue de $W(0)$ est stockée dans la même trame $W(0)$. Ce processus est résumé dans la figure 8.5.

Sauvegarde $W(0)$ est le résultat de la restauration de $I(t)$. Avant donc de passer à la boucle suivante il faut sauvegarder cette trame. Nous verrons plus de détails sur cette étape dans la suite.

Fin de la boucle et décalage de W Une fois $W(0)$ sauvegardée, on incrémente le compteur t et on "décale" l'image de travail W de la façon suivante:

$$\begin{array}{ll} \forall -T < j < T-1 & W(j) = W(j+1) \\ & W(T) = I(t+T) \end{array}$$

La nouvelle trame $I(t+T)$ incorporée dans notre image de travail est lue sur le disque dur. Nous verrons des détails sur cette étape dans la section suivante.

Les procédures de lecture, décalage et sauvegarde sont illustrées par la figure 8.6.

8.4.2 Mémoire de masse

Nous voulons rendre notre système aussi souple que possible et ne pas nous occuper de l'origine ou de l'acquisition des images, ni de leur stockage définitif. Nous supposerons donc soit que la séquence est suffisamment courte pour tenir en entier sur le disque dur (ce qui n'est pas très réaliste), soit qu'un autre programme s'occupera de transférer les données

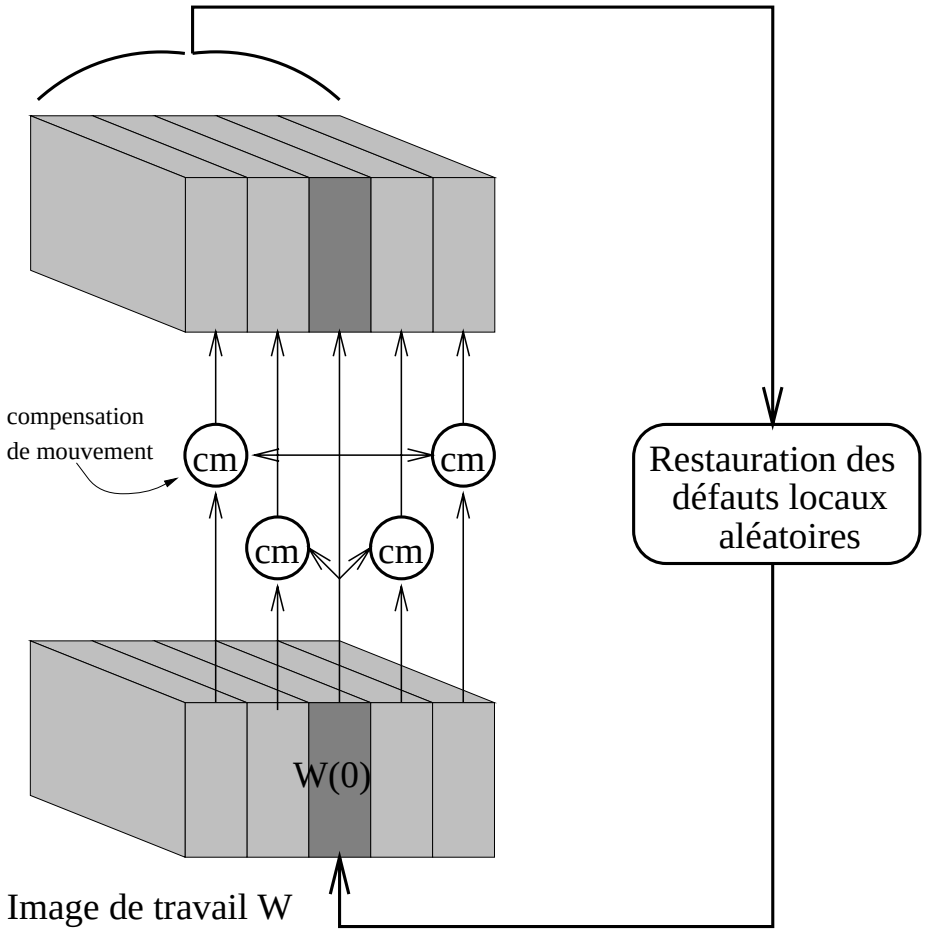


FIG. 8.5 - *Variables d'entrée et de sortie de l'algorithme de restauration des défauts locaux aléatoires*

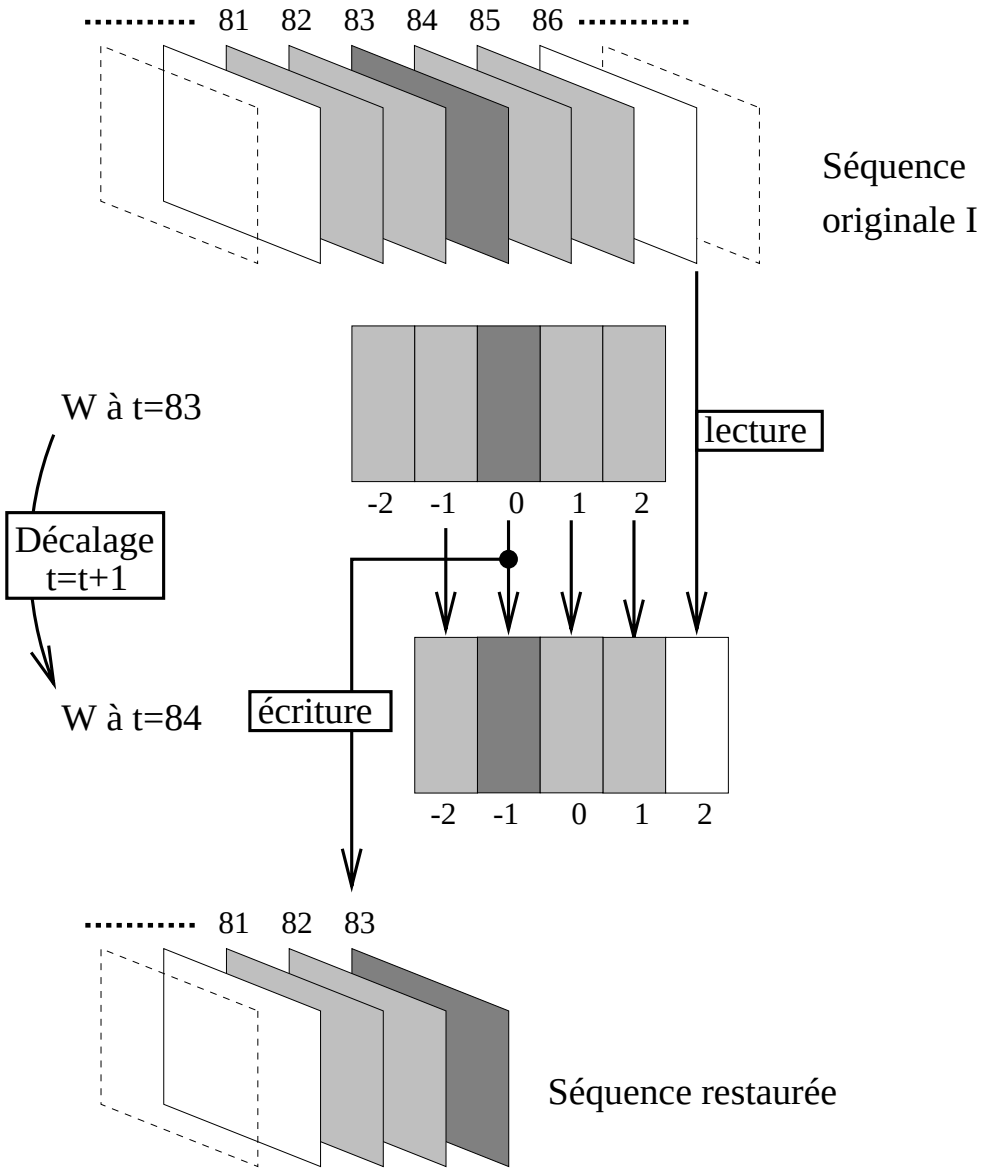


FIG. 8.6 - *Lecture d'une trame, décalage de W et sauvegarde de la trame restaurée.*

sur le disque dur, de les effacer quand elles ne seront plus nécessaires, et de stocker sur un support idoine les images restaurées pour ensuite effacer celles-ci du disque dur.

C'est pour cela que notre système à chaque étape de lecture vérifiera avant tout que l'image qu'il cherche à lire est bien présente sur le disque, et seulement après la lira. Si l'image ne s'y trouve pas, on considérera qu'elle n'a pas encore été transférée sur le disque, et par conséquent, après un certain délai, on fera un nouvel essai de lecture, et ainsi de suite.

De même, à chaque étape d'ecriture, si l'opération de sauvegarde échoue pour cause de manque de place sur le disque, le système attendra un certain temps, puis recommencera. En théorie, le programme d'écriture/lecture devrait tôt ou tard libérer de la place mémoire et le système repartir.

8.4.3 Schéma global

L'ensemble des étapes de restauration, ainsi que les phases de lecture et d'écriture, sont résumées par le schéma de la figure 8.7.

8.5 Performances

Tout au long de ce travail nous nous sommes focalisés sur la qualité des résultats, et non pas tellement sur la rapidité d'exécution des algorithmes. Par ailleurs, la vitesse de traitement dans notre cadre n'est pas aussi critique que dans d'autres applications. Il est parfaitement envisageable de laisser tourner un ordinateur pendant plusieurs jours pour restaurer un film.

Cependant, vu le grand nombre d'images à traiter (à 24 images par seconde nous avons 86400 images par heure de film) et les résolutions très élevées que nous pouvons être amenés à considérer (jusqu'à 4000 points par ligne) nous avons intérêt à ce que les temps de traitement restent raisonnables. D'autant plus que, plus la restauration sera rapide, plus le coût de traitement sera faible!

Nous donnons donc dans la suite quelques mesures de temps de traitement. Ces mesures ont été effectuées pour la séquence "Train", qui constitue le cas de dégradation le plus sévère que nous avons eu à traiter. Sa résolution est de 512×512. Le traitement a été effectué sur un ordinateur SUN SPARC-5. La programmation a été réalisée en partie en C, et en partie en LISP, en utilisant le logiciel XLIM3D développé par le Centre de Morphologie Mathématique de l'Ecole des Mines de Paris. Les algorithmes peuvent être notablement améliorés. Par exemple une technique de descente de gradient permettrait sûrement d'accélérer le calcul des translations globales entre trames successives utilisé pour la correction des vibrations. Par ailleurs le code peut être optimisé.

Par conséquent, il faut considérer les temps de traitement donnés ci-dessous (cf. tableau 8.2) comme une majoration très pessimiste des performances possibles. Nous pensons qu'il est envisageable de diminer d'un facteur 100 ces durées.

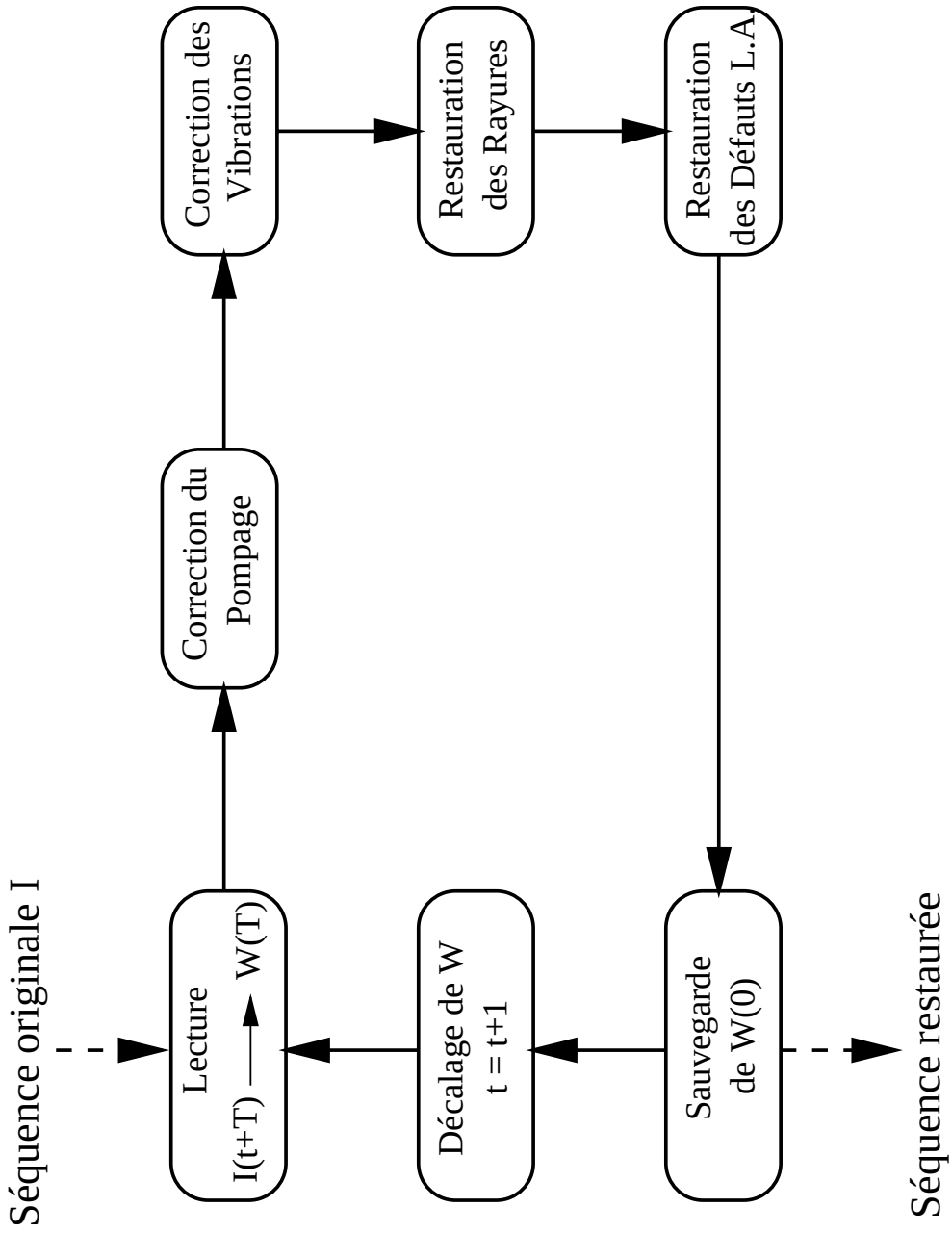


FIG. 8.7 - *Schéma global simplifié du système de restauration SARSA*

Algorithme	Temps de calcul (par image)
Pompage	4s
Vibrations (121 vecteurs testés)	4min
Rayures	30s
Craquelures noires	2min
Défauts locaux aléatoires	2min
Compensation de mouvement	10mn

TAB. 8.2 - *Mesures de temps de traitement pour la séquence "Train" sur une SUN SPARC-5*

8.6 Conclusion

Dans ce chapitre nous avons présenté le système global de restauration de films anciens, baptisé SARSA, où viennent s'enchasser les algorithmes de restauration décrits dans les chapitres précédents.

Le système résultant est souple. Il permet à un opérateur de choisir les traitement qu'il veut pour chaque séquence d'images. A partir de là tout se fait automatiquement. D'ailleurs nous avons prévu un jeu de paramètres par défaut destiné à traiter les défauts les plus courants. La gestion des ressources informatique est assez efficace, quoique pour l'instant les temps de calcul soient assez longs. Les optimisations de code restent à faire.

Le prototype résultant permet de restaurer déjà des films anciens avec de bons résultats.

Chapitre 9

Conclusion

Le patrimoine cinématographique mondial est en danger. Pour le sauver il faut transférer les films sur des supports à plus longue durée de vie et réparer les dégâts qui ont déjà été faits. La seule solution économiquement viable pour mener à bien ce travail de restauration à grande échelle est de développer un système automatique de restauration, ce qui devient possible grâce aux progrès de l'informatique.

Par ailleurs, le développement du multimedia entraîne une augmentation importante de la demande en matériel vidéo de bonne qualité. Un système de restauration permettrait donc d'exploiter les archives cinématographiques et video pour satisfaire ce besoin croissant.

9.1 Objectifs atteints

Nous avons mis au point plusieurs algorithmes qui permettent de corriger les défauts les plus courants des films anciens:

- Pompage: l'analyse des histogrammes des images de la séquence nous a permis de corriger ce défaut.

- Vibrations: en étudiant les translations globales entre trames consécutives nous avons réussi à séparer les translations normales, dues au mouvement de la scène, des vibrations parasites.

- Rayures verticales: nous avons mis au point un algorithme simple et robuste permettant de les détecter trame par trame.

- Défauts locaux aléatoires: en nous basant sur l'analyse de la connexité de la séquence, nous avons développé une méthode générale et robuste de détection de ces défauts. Nous avons aussi proposé plusieurs algorithmes d'interpolation permettant de récupérer l'information perdue.

- Compensation de mouvement: Nous avons aussi proposé une structure générale pour un algorithme de compensation orienté objet utilisant le krigeage. Il permet d'amé-

liorer la corrélation entre les trames consécutives de la séquence, ce qui facilite l'étude basée sur la connexité.

– Modélisation de champs des vecteurs: Nous avons développé le krigeage inverse, qui nous permet avec une grande souplesse de modéliser et filtrer les champs de vecteurs. Il nous a permis d'améliorer la robustesse de l'algorithme de compensation de mouvement.

Ces algorithmes nous ont permis de bâtir un système de restauration automatique de séquences d'images. Les résultats obtenus sont encourageants.

9.2 Autres applications

Les outils que nous avons développés dans le cadre de notre travail peuvent être appliqués avec succès à d'autres domaines. Voyons quelques exemples.

La méthode de correction des vibrations peut être utilisée pour stabiliser les séquences d'images enregistrées par des camescopes amateur.

Les opérateurs par reconstruction que nous avons utilisés pour détecter les défauts aléatoires sont très utiles pour l'analyse de séquences en général. Il permettent par exemple de détecter les objets qui se déplacent rapidement, ainsi que les objets qui apparaissent ou qui disparaissent dans une séquence [16].

La méthode de compensation de mouvement que nous avons développée, basée sur le krigeage et le krigeage inverse, sera utile dans de nombreuses applications d'analyse de séquences d'images.

9.3 Futurs développements

Ce sujet est loin d'être bouclé. Nous pouvons améliorer notre système de restauration de nombreuses façons.

Les algorithmes présentés peuvent être améliorés. Pour l'instant leur implémentation n'est pas efficace. L'optimisation de leur code permettrait d'accélérer le traitement. Des recherches sur le choix automatique des paramètres à utiliser nous semblent aussi intéressantes.

Par ailleurs, la souplesse du système permet le rajout de nouveaux algorithmes pour traiter d'autres défauts.

Finalement, des progrès considérables restent à faire dans le domaine de l'estimation et de la compensation de mouvement.

Bibliographie

[1] G.R. Arce. Multistage order statistics filters for image sequence processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 39(5):1146–1163, 1991.

[2] S. Beucher. Road segmentation by watershed algorithms. Dans *PROMETHEUS Workshop, Sophia Antipolis, France*, Avril 1990.

[3] M. Bierling. Displacement estimation by hierarchical block matching. Dans *Proc. SPIE: Visual communications and image processing*, volume 1001, pages 942–951, 1988.

[4] M.J. Black et A.D. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, 1996.

[5] L. Bouchard, E. Decencière Ferrandière, et P. Salembier. From texture to feature. Race R2053 Deliverable R2053/UPC/GPS/DS/R/014/b1, Morpheco Consortium, Novembre 1995.

[6] C. Cafforio et F. Rocca. Methods for measuring small displacements of television images. *IEEE Trans. Inform. Theory*, 22(5):573–579, 1976.

[7] J.R. Casas, E. Decencière Ferrandière, et P. Salembier. Feature-based coding algorithm. Race R2053 Deliverable R2053/UPC/GPS/DS/R/016/b1, Morpheco Consortium, Janvier 1996.

[8] P. Chauvet. *Processing data with a spatial support: Geostatistics and its methods.* Ecole des Mines de Paris, 1993.

[9] M.N. Chong, P. Liu, W.B. Goh, et D. Krishnan. A new spatio-temporal MRF model for the detection of missing data in image sequences. Dans *Proceedings of ICASSP'97*, Munich, Germany, Avril 1997.

[10] I. Corset, L. Bouchard, S. Jeannin, P. Salembier, F. Marques, M. Pardàs, R. Morros, F. Meyer, et B. Marcotegui. Technical description of SESAME (segmentation-based coding system allowing manipulation of objects). Rapport technique, ISO/IEC JTC1/SC29/WG11. MPEG 95 / 408, Novembre 1995.

[11] Commission Supérieure Technique de l'Image et du Son (CST). *Guide de la conservation des films*. CST, Paris, 1995.

[12] Commission Supérieure Technique de l'Image et du Son (CST). *La restauration numérique des films cinématographiques*. CST, Paris, 1997.

[13] E. Decencière Ferrandière. Motion picture restoration using morphological tools. Dans P. Maragos, R.W. Schafer, et M.A. Butt, éditeurs, *Mathematical Morphology and its applications to signal processing (Proceedings ISMM'96)*, pages 361–368, Atlanta (GA), USA, Mai 1996. Kluwer Academic Publishers.

[14] E. Decencière Ferrandière. Restoration of old motion pictures. *Microscopy, Microanalysis, Microstructures*, 7(5/6):311–316, Octobre/Décembre 1996.

[15] E. Decencière Ferrandière, C. de Fouquet, et F. Meyer. Applications of kriging to image sequence coding. *A paraître dans Signal Processing: Image Communication*, 1997.

[16] E. Decencière Ferrandière, S. Marshall, et J. Serra. Application of the morphological geodesic reconstruction to image sequence analysis. *IEE Proceedings: vision, image and signal processing*, December 1997.

[17] E. Decencière Ferrandière et J. Serra. Detection of local defects in old motion pictures. Dans *VII National Symposium on Pattern Recognition and Image Analysis*, pages 145–150, Barcelona, Spain, Avril 1997.

[18] R. Depommier et E. Dubois. Motion estimation with detection of occlusion areas. Dans *Proceedings of ICASSP'92*, pages 269–273, San Francisco, USA, Avril 1992.

[19] N. Diehl. Object-oriented motion estimation and segmentation in image sequences. *Signal Processing: Image Communication*, 3:23–56, 1991.

[20] M. Van Droogenbroeck. *Traitement d'images numériques au moyen d'algorithmes utilisant la morphologie mathématique et la notion d'objet : application au codage*. Thèse de doctorat, Ecole des Mines de Paris, Paris, Mai 1994.

[21] E. Dubois et S. Sabri. Noise reduction in image sequences using motion compensated temporal filtering. *IEEE Transactions on Communications*, 32(7):826–831, 1984.

[22] M. Dudon, O. Avaro, et C. Roux. Triangle-based motion estimation and temporal interpolation. Dans *IEEE workshop on nonlinear signal and image processing (NSIP'95)*, pages 242–245, Haldiki, Greece, Juin 1995.

[23] F. Friedlander. *Le traitement morphologique d'images en cardiologie nucléaire*. Thèse de doctorat, Ecole des Mines de Paris, Paris, Décembre 1989.

[24] M. Friend. Film/digital/film. Dans *AMIA Conference*. AMIA, Octobre 1994.

[25] S. Geman et D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, Novembre 1984.

[26] S. Geman, D.E. McClure, et D. Geman. A non linear filter for film restoration and other problems in image processing. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, 54(4):281–289, Juillet 1992.

[27] M. Ghanbari, S. de Faria, I.N. Goh, et K.T. Tan. Motion compensation for very low bit-rate video. *Signal Processing : Image Communication*, 7(4-6):567–580, Novembre 1995.

[28] M. Gilge, T. Engelhardt, et R. Mehlan. Coding of arbitrarily shaped image segments based on a generalized orthogonal transform. *Signal Processing : Image Communication*, 1(2):153–180, Octobre 1989.

[29] S.J. Godsill et A.C. Kokaram. Joint interpolation, motion and parameter estimation for image sequences with missing data. Dans *Signal Processing VIII: Theories and Application (proceedings of EUSIPCO-96)*, Trieste, Italy, Septembre 1996. European association for signal processing (EURASIP).

[30] S.J. Godsill et A.C. Kokaram. Restoration of image sequences using a casual spatio-temporal model. Dans *17th Leeds annual statistics research workshop*, Juillet 1997.

[31] W.B. Goh, M.N. Chong, S. Kalra, et D. Krishnan. Bi-directional 3D auto-regressive model approach to motion picture restoration. Dans *IEEE International conference on acoustics, speech and signal processing*, pages 2277–2280. IEEE, Mai 1996.

[32] C. Gratin. *De la Représentation des Images au Traitement Morphologique d 'Images Tridimensionelles*. Thèse de doctorat, Ecole des Mines de Paris, Paris, 1993.

[33] F. Heitz et P. Bouthemy. Multimodal estimation of discontinuous optical flow using markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1217–1232, 1993.

[34] B.K.P. Horn et B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[35] T.S. Huang et Y.P. Hsu. Image sequence enhancement. Dans T.S. Huang, éditeur, *Image sequence analysis*, chapitre 4, pages 289–309. Springer-Verlag, 1981.

[36] J.R. Jain et Jain A.K. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communications*, 29(12):1799–1808, 1981.

[37] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, pages 35–45, Mars 1960.

[38] S. Kalra, M.N. Chong, et D. Krishan. A new auto-regressive (AR) model-based algorithm for motion picture restoration. Dans *Proceedings of ICASSP'97*, Munich, Germany, Avril 1997.

[39] R.P. Kleihorst, R.L. Lagendijk, et J. Biemond. Noise reduction of image sequences using motion compensation and signal decomposition. *IEEE Transactions on Image Processing*, 4(3):274–284, 1995.

[40] J.C. Klein. *Conception et réalisation d'une unité logique pour l'analyse quantitative d'images*. Thèse de doctorat, Université de Nancy, France, 1977.

[41] A.C. Kokaram. *Motion picture restoration*. Thèse de doctorat, Cambridge University, Mai 1993.

[42] A.C. Kokaram. Detection and removal of line scratches in degraded motion picture sequences. Dans *Signal Processing VIII: Theories and Application (proceedings of EUSIPCO-96)*, Trieste, Italy, Septembre 1996. European association for signal processing (EURASIP).

[43] A.C. Kokaram et J.G. Godsill. A system for reconstruction of missing data in image sequences using sampled 3D AR models and MRF motion priors. Dans F. Buxton et R. Cipolla, éditeurs, *Computer Vision - ECCV'96*, Cambridge, UK, Avril 1996. European Vision Society.

[44] A.C. Kokaram, R.D. Morris, W.J. Fitzgerald, et P.J.W. Rayner. Detection of missing data in image sequences. *IEEE Transactions on Image Processing*, 4(11):1496–1508, 1995.

[45] A.C. Kokaram, R.D. Morris, W.J. Fitzgerald, et P.J.W. Rayner. Interpolation of missing data in image sequences. *IEEE Transactions on Image Processing*, 4(11):1508–1519, 1995.

[46] A.C. Kokaram et P. Rayner. Removal of replacement noise in motion picture sequences using 3D auto-regressive modelling. Dans *Signal Processing VII: Theories and Application (proceedings of EUSIPCO-94)*, volume 3, Edinburgh, United Kingdom, Septembre 1994. European association for signal processing (EURASIP).

[47] J. Konrad et E. Dubois. Bayesian estimation of motion vector fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):910–927, 1992.

[48] Ouseb Lee et Tao Wang. Motion-compensated prediction using modal-based deformable block matching. *Journal of Visual Communication and Image Representation*, 6(1):26–34, Mars 1995.

[49] Beatriz Marcotegui. *Segmentation de séquences d'images en vue du codage*. Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1996.

[50] G. Matheron. *Traité de géostatistique appliquée, tome II.* Editions Techniques, Paris, 1963.

[51] G. Matheron. *Les variables régionalisées et leur estimation.* Masson, Paris, 1965.

[52] G. Matheron. *Le krigeage universel (cahiers du CMM, fascicule 1).* Ecole des Mines, Paris, 1969.

[53] G. Matheron. *La théorie des variables régionalisées et ses applications.* Ecole des Mines, Paris, 1970.

[54] G. Matheron. The intrinsic random functions, and their applications. *Advances in Applied Probability*, 5:439–468, 1973.

[55] G. Matheron. *Random sets and integral geometry.* Wiley, New York, 1975.

[56] G. Matheron. Splines and kriging: their formal equivalence. Dans D.F. Merrian, éditeur, *Down-to-earht statistics : solutions looking for geological problems*, pages 77–95. Syracuse university. Geology contributions, 1981.

[57] F. Meyer. *Cytologie quantitative et morphologie mathématique.* Thèse de doctorat, Ecole des Mines de Paris, Paris, 1979.

[58] O. Monga. *Segmentation d'Images par Croissance Hiérarchique de Régions.* Thèse de doctorat, Université Paris Sud. Centre d'Orsay, 1988.

[59] R.D. Morris et W.J. Fitzgerald. Replacement noise in image sequences - detection and interpolation by motion field segmentation. Dans *Proceedings of ICASSP'94*, Adelaide, Australia, Avril 1994.

[60] R.D. Morris et W.J. Fitzgerald. Stochastic and deterministic methods in motion picture restoration. Dans *Proceedings of international workshop in image processing*, Budapest, Hungary, Juin 1994.

[61] R.D. Morris et W.J. Fitzgerald. Stochastic and deterministic methods in motion picture restoration. *Journal of Communications*, 45:17–21, Juillet-Août 1994.

[62] Robin D. Morris. *Image Sequence Restoration via Gibbs distributions.* Thèse de doctorat, Trinity College, 1994.

[63] J. Nieweglowski et P. Haavisto. Temporal image sequence prediction using motion field interpolation. *Signal Processing: Image Communication*, 7(4-6):333–353, Novembre 1995.

[64] M.K. Ozkan, M.I. Sezan, et A.M. Tekalp. Adaptive motion-compensated filtering of noisy image sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 3(4):277–290, 1993.

[65] M. Pardàs et P. Salembier. 3D morphological segmentation and motion estimation for image sequences. *Signal Processing*, 38:31–43, 1994.

[66] M. Pardàs, J. Serra, et L. Torrés. Connectivity filters for image sequences. *Society of Photo-Instrumentation Engineers*, 1769:318–329, 1992.

[67] V. Pinel. La restauration des films. Dans J. Aumont, A. Gaudreault, et M. Marie, éditeurs, *Histoire du cinéma: nouvelles approches (Colloque de Cerisy)*. Publications de la Sorbonne, Août 1989.

[68] W.H. Press, B.P. Flannery, S.A. Teukolsky, et W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[69] P. Richardson et D. Suter. Restoration of historic film for digital compression : A case study. Dans *Proceedings of ICIP-95*, volume 2, pages 49–52. IEEE Computer Society Press, Mai 1995.

[70] P. Salembier. Practical extensions of connected operators. Dans P. Maragos, R.W. Schafer, et M.A. Butt, éditeurs, *Mathematical Morphology and its applications to signal processing (Proceedings ISMM'96)*, pages 97–110, Atlanta (GA), USA, Mai 1996. Kluwer Academic Publishers.

[71] P. Salembier, F. Marqués, M. Pardàs, R. Morros, I. Corset, S. Jeannin, L. Bouchard, F. Meyer, et B. Marcotegui. Segmentation-based video coding system allowing the manipulation of objects. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):60–74, Février 1997.

[72] P. Salembier et J. Serra. Multiscale image segmentation. Dans *Proc. SPIE: Visual communications and image processing*, volume 1818, pages 620–631, Boston, USA, Novembre 1992.

[73] J. Serra. *Image analysis and mathematical morphology*. Academic Press, 1982.

[74] J. Serra. *Image Analysis and Mathematical Morphology - Volume II: Theoretical Advances*. Academic Press, London, 1988.

[75] J. Serra. Connectivity on complete lattices. *Accepted for publication in: Journal of Mathematical Imaging and Vision*, 1997.

[76] J. Serra et P. Salembier. Connected operators and pyramids. Dans *Proc. SPIE: Non linear algebra and morphological image processing*, volume 2030, pages 65–76, San Diego, USA, Juillet 1993.

[77] A.G. Setos. The fox movietone news preservation project: an introduction. *SMPTE Journal*, 105(9):532–536, 1996.

[78] T. Sikora. Low complexity shape-adaptive DCT for coding of arbitrarily shaped image segments. *Signal Processing : Image Communication*, 7:381–395, 1995.

[79] P. Soille. Geodesic transformations in mathematical morphology. Rapport technique N-24/94/MM, Centre de Morphologie Mathématique, Ecole des Mines de Paris, Janvier 1994.

[80] S.R. Sternberg. Morphology for grey tone functions. *Computer Vision, Graphics, and Image Processing*, 35, 1986.

[81] R. Storey. Electronic detection and concealment of film dirt. *SMPTE Journal*, pages 642–647, Juin 1985.

[82] T.Y. Tian et M. Shah. Motion estimation and segmentation. *Machine Vision and Applications*, 9:32–42, 1996.

[83] P. van Roosmalen, S.J.P. Westen, R.L. Lagendijk, et J. Biemond. Noise reduction for image sequences using an oriented pyramid thresholding technique. Dans *Proceedings of ICIP-96*, volume 1, pages 375–378, Lausanne, Switzerland, Septembre 1996. IEEE Computer Society Press.

[84] L. Vincent. *Algorithmes morphologiques à base de files d'attente et de lacets. Extension aux graphes.* Thèse de doctorat, Ecole des Mines de Paris, Paris, Mai 1990.

[85] L. Vincent. Morphological area opening and closing for gray scale images. Dans *Proceedings of NATO: Shape in picture workshop*, Driebergen, The Netherlands, Septembre 1992.

[86] L. Vincent. Grayscale area openings and closings, their efficient implementation and applications. Dans J. Serra et P. Salembier, éditeurs, *Mathematical Morphology and its applications to signal processing (Proceedings ISMM'93)*, pages 22–27, Barcelona, Spain, Mai 1993. UPC Publications Office.

[87] T. Vlachos. Improving the efficiency of MPEG-2 coding by means of film unsteadiness correction. Dans *Proc. SPIE: Digital Compression Technologies and Systems for Video Communications*, volume 2952, pages 534–542, Octobre 1996.

[88] T. Vlachos et G. Thomas. Motion estimation for the correction of twins-lens telecine flicker. Dans *Proceedings of ICIP-96*, volume 1, pages 109–112, Lausanne, Switzerland, Septembre 1996. IEEE Computer Society Press.

[89] J. Zhang et G.G. Hanauer. The application of mean field theory to image motion estimation. *IEEE Transactions on Image Processing*, 4(1):19–33, Janvier 1995.