Segmentation de séquences d'images en vue du codage

Beatriz Marcotegui

ECOLE NATIONALE SUPERIEURE DES MINES DE PARIS

SEGMENTATION DE SEQUENCES D'IMAGES EN VUE DU CODAGE

THESE

présentée devant
l'Ecole Nationale Supérieure des Mines de Paris
par

Beatriz MARCOTEGUI

pour obtenir le titre de Docteur en Morphologie Mathématique

Soutenue le 5 Avril 1996 à Fontainebleau devant le jury composé de:

MM.	Jean	SERRA	*Président*
	Touradj	EBRAHIMI	*Rapporteur*
	Gérard	EUDE	*Examinateur*
	François	HOTTIER	*Examinateur*
	Ferrán	MARQUES	*Rapporteur*
	Fernand	MEYER	*Examinateur*

A ma famille (Luis, Evelia, Mª Jesús y Antonio)
A mes parents adoptifs Liliane et Gérard

Whether you think you can or
you can't - you are right
Zig Ziglar

MERCI

à Jean Serra qui, grâce à son année sabbatique à Barcelone, m'a donné l'occasion et le plaisir de rentrer dans l'intéressant domaine de la Morphologie Mathématique. Merci pour toutes les connaissances apprises et pour l'accueil exceptionnel dans un centre actif à niveau scientifique, mais aussi humain.

à Fernand Meyer d'avoir guidé et suivi de près ce travail. Merci pour d'avoir su m'encourager même quand les conditions n'étaient pas les plus favorables. Merci pour sa pleine disponibilité.

à Ferrán Marqués, mon responsable à Barcelone, à côté de qui il est impossible de ne pas être joyeux.

à Philippe Salembier et Montse Pardàs, qui avec leurs bons résultats m'ont toujours poussé à améliorer les miens.

à José Crespo avec qui j'ai eu des fructueuses discussions.

à Etienne Decencière pour tout.

à Christophe Gratin, père de mes connaissances en informatique, pour sa précieuse aide professionnelle ainsi que l'amitié qu'il m'a offerte.

à Michel Bilodeau l'encyclopédie vivante, toujours ouverte à la bonne page.

à Roland Brémond pour ses nombreuses relectures de mes rédactions et ses conseils avisés.

à Pascal Laurenge, la tranquillité personnifiée, qui m'a appris par l'exemple qu'il faut vivre la vie sans stress.

à Fabrice Lemonnier pour son cœur ouvert.

à Laura Andriamasinoro pour ses bons conseils.

à Lionel Bouchard, Patrick Brigger, Josep Ramón Casas, I. Corset, Toni Gasull, Chuang Gu, Sylvie Jeannin, Ferrán Marqués, Josep Ramón Morros, Montse Pardàs, Óscar Ribes, Philippe Salembier et Lluis Torres pour la bonne ambiance qu'ils ont su créer à l'intérieur du projet Morpheco.

à Michel Gauthier et à Marc Waroquier pour leur bonjour du matin.

à Liliane et Gérard Pipault pour être graine de bonheur et de bonne humeur. Merci d'avoir rempli le rôle de parents adoptifs.

à ma famille (père, mère, tante et frère) au sein de laquelle j'ai eu la bonne chance de naître et le bonheur de vivre.

à Jaime, pour avoir gardé la clé du bonheur pendant deux ans en théorie, trois ans et demi en pratique, sans avoir désespéré une seule minute.

à Lothar Bergen et Kiki Wiginton, Serge Beucher, Jacek Cichoz, Luc Decker, Claire Helène Demarty, Alice Gauthier, Cezer Goren, Dimitry et Irena Gorokhovik, Nicole Guegan, Philippe Guiblin, Marcin Iwanowski, Dominique Jeulin, Vladimir Kharitonov, Jean-Claude Klein, Laure Messiaen, Mariusz Mlynarczuk, Réné Peyrard et Evelyne Gagnaire, Raphael Sasportas, Laurent Savary, Pierre Soille, Edouard Squillaci, Hugues Talbot, Gilberto Tillán, Corinne Vachier, Marc Vandrogenbroeck qui ont contribué généreusement à l'ambiance sympathique du centre et qui ont rendu mon séjour à Fontainebleau inoubliable.

Contents

Résumé

Le développement d'applications telles que la vidéotéléphonie, la vidéoconférence ou le multimedia ont créé la nécessité de coder des séquences d'images à très bas débit. Les systèmes de compression d'image orientés objet, appelés aussi méthodes de deuxième génération, offrent une alternative prometteuse aux méthodes classiques. Ces méthodes se rapprochent du comportement de l'œil humain, destinataire final des systèmes de codage, et décrivent une image en termes d'objets présents dans la scène. Ainsi, un système de codage orienté objet opère en deux étapes: une première étape d'analyse, appelée segmentation, qui décompose l'image en objets (ensembles de pixels vérifiant une condition d'homogénéité) et une seconde étape qui modélise la forme et le contenu des objets segmentés.

Le travail de cette thèse est consacré à la segmentation de séquences d'images, sachant que l'objectif final est de l'intégrer dans un système de codage orienté objet.

Nous avons commencé notre travail en nous fondant sur le paradigme de segmentation de la morphologie mathématique: la ligne de partage des eaux du gradient à partir des marqueurs des régions que l'on veut extraire. Ce schéma a donné lieu à de nombreuses applications mais, dans le domaine du codage, cela conduit à des difficultés réelles. Ces difficultés sont dues à la présence de détails fins, ou, ce qui revient au même, à la présence de mouvements rapides qui génèrent des objets fins dans l'axe temporel. En effet, dans le domaine du codage, les conditions d'échantillonnage morphologique ne sont pas vérifiées. L'analyse des difficultés rencontrées nous a conduit à un autre paradigme de segmentation, basé cette fois-ci sur la fusion de régions.

Nous avons ainsi développé un algorithme de fusion de régions. Implémenté sur une structure de graphe dont les nœuds et les arêtes sont évalués, l'algorithme est à la fois efficace et suffisamment flexible pour prendre en compte une large variété de critères. En particulier, dans le but de rendre le système de codage globalement cohérent, nous avons pris en compte dans les critères de fusion les capacités des algorithmes de codage appliqués après la segmentation.

Enfin, nous avons illustré le potentiel qu'offre la segmentation à un système de codage. En effet, la segmentation introduit la notion d'objet dans le codeur. Ainsi le développement de fonctionnalités, telles que le suivi d'objets, devient

une extension naturelle des algorithmes de codage. Ces fonctionnalités rentrent dans le cadre des nouveaux objectifs fixés par la norme de codage vidéo MPEG 4.

Chapter 1

Introduction

1.1 La compression d'image; définition du taux de compression

Récemment la compression d'image, et en particulier la compression de séquences d'images, est devenue un sujet de grand intérêt grâce au développement d'applications telles que le vidéotéléphone, la vidéoconférence, et les multimedia pour ne citer que les plus importantes. Les images contiennent une énorme quantité d'information ce qui rend coûteux leur stockage ou leur transmission si elles ne sont pas préalablement compressées. Par exemple, pour transmettre une séquence de 25 images par seconde de 256×256 pixels avec 256 niveaux de gris (8 bits par pixel), sans la compresser, il faudrait un débit de 13 Mbits / seconde. Afin de mieux exploiter les moyens de transmission, qui sont physiquement limités, on est obligé de rechercher des redondances de manière à transmettre l'information avec un nombre de bits plus faible. Le but de la compression d'image est de réduire autant que possible le nombre de bits nécessaire pour reconstruire une copie fidèle de l'image originale. La représentation canonique d'une image digitale requiert $B = N \times M \times b$ bits, où $N \times M$ sont les dimensions de l'image et b le nombre de bits pour représenter un point-image (un échantillon d'image sera appelé pixel qui est une abréviation du terme anglais "picture element"). Si on est capable de reconstruire une copie de l'image à partir de C bits, on dira que le taux de compression obtenu est de

$$T = \frac{B}{C}$$

Pour un système de compression donné, plus le taux de compression est élevé, plus l'image codée s'éloigne de l'image originale. Selon les applications, les taux de compression recherchés sont différents. Une première classification des méthodes de codage peut être envisagée à partir des objectifs que l'on se fixe. Certaines méthodes ont été définies pour être capables de reconstruire exactement l'image originale (méthodes sans pertes). Les taux de compression atteints ne dépassent pas 2 ou 3. Cette contrainte est souvent trop stricte. Le taux de compression dont on a besoin dans la plupart des applications concernant des images, est supérieur d'au moins un ordre de grandeur à celui qu'on utilise dans la compression de données, mais en même temps il permet une petite distorsion

de l'information dans la mesure où l'œil humain ne la perçoit pas. L'objectif est de restituer une copie de l'original avec une certaine distorsion, mais visuellement proche de l'image de départ. On peut obtenir des images en apparence identiques à l'original avec un taux de compression d'environ 10. Néanmoins, certaines applications requièrent un taux de compression plus élevé. Pour ceci, il est nécessaire d'extraire l'information qui nous permettra de récupérer une image simplifiée mais représentative de l'image originale.

1.2 Des méthodes classiques aux méthodes orientées objet

Les premiers pas en compression d'image utilisent la théorie de l'information. Deux groupes de techniques de codage ont été développés depuis les années soixante dans ce cadre: les techniques spatiales, qui travaillent directement avec les valeurs des échantillons de l'image et les méthodes transformées, qui utilisent la représentation spectrale de l'image et la simplifient. Ces méthodes, décomposant l'image en blocs réguliers et exploitant la corrélation à l'intérieur de chaque bloc, dans le domaine spatial ou spectral, font partie de ce qu'on appelle *première génération des systèmes de compression d'image.* Les blocs utilisés pour exploiter la corrélation de l'image ne sont pas liés à son contenu. Les transitions entre blocs deviennent visibles quand on essaye d'atteindre des taux de compression élevés.

Les méthodes orientées objet, appelées aussi méthodes de *deuxième génération*, apparaissent comme une alternative. Ces méthodes se rapprochent du comportement de l'œil humain, destinataire final des systèmes de codage, et décrivent une image en termes d'objets. L'image n'est plus décomposée en blocs arbitraires du point de vue de son contenu, mais en objets correspondant à des entités présentes dans la scène. L'intérêt du codage orienté objet est d'obtenir des images simplifiées mais contenant une information liée au contenu de la scène. On peut espérer qu'une analyse de l'image à traiter optimise les performances du codeur.

Les méthodes de la première génération codent simplement le contenu de chaque bloc (la position et la forme des blocs est connue a priori); par contre, les méthodes orientées objet doivent coder la position, la forme et le contenu de chaque objet. Si une image est représentée en détail, beaucoup d'informations vont être nécessaires pour coder les contours des objets, ce qui désavantage les méthodes orientées objet par rapport aux méthodes classiques. Par contre, les méthodes orientées objet donnent de meilleurs résultats lorsque la compression est très forte: elles permettent une grande simplification de l'image en ne retenant de l'image que les traits géométriques saillants, alors que les hautes fréquences auxquelles l'œil humain est moins sensible sont négligées.

Les méthodes orientées objet présentent aussi un intérêt du point de vue des applications à la robotique. La décomposition de l'image en objets est un point de départ pour sa compréhension, ce qui ouvre les portes à des applications d'automatisme. Elle offre aussi un grand potentiel pour développer les nouvelles fonctionnalités de MPEG-4, telles que la manipulation d'objets, le codage sélectif (coder avec plus de précision le premier plan, par rapport au fond), le suivi d'objets .... La structure d'un système de codage orienté objet est un bon

support pour développer des applications "content based"

1.3 Méthodes orientées objet

Les méthodes orientées objet décrivent une image en termes d'objets présents dans une scène et codent la position, la forme (les contours) et le contenu (la texture) de chacun des objets détectés. L'objectif n'est pas de restituer une image identique à l'original, mais d'extraire l'information pertinente décrivant la scène. Les propriétés géométriques des objets sont préservées, ce qui élimine les effets de bloc d'une approche classique. Un système de codage orienté objet opère en 3 étapes:

1. La première étape est la segmentation, chargée d'identifier les objets présents dans la scène. Un objet est défini comme un ensemble de pixels (une région) vérifiant une condition d'homogénéité On lui assigne une étiquette (ou label) qui sert à l'identifier. La plupart des applications d'analyse d'image se focalisent sur un champ d'application particulier, ce qui permet d'adapter les algorithmes à chaque cas particulier. Par exemple, dans une application médicale on peut avoir à chercher des particules correspondant à des cellules, ou à des tumeurs, avec des critères de taille, de forme, de texture ou de couleur. Les caractéristiques des objets cherchés sont définies dans les spécifications de l'application. Le codage devant s'appliquer à une grande variété d'images, le critère de sélection d'objets ne doit pas utiliser des informations sur l'image a priori. L'introduction de la segmentation dans une application de codage nous oblige à définir des critères d'homogénéité généraux, valables pour une image d'entrée arbitraire.

2. Une fois que les objets ont été détectés dans l'étape de segmentation, leurs position et forme doivent être codées. Le codage de contours code l'information relative à la partition de l'espace.

3. Finalement le codage de texture code le contenu des objets.

1.4 Plan de cette thèse

Le propos de cette thèse est le développement d'un algorithme de segmentation de séquences d'images sachant que l'objectif final est de l'intégrer dans un système de codage orienté objet.

Nous avons commencé notre travail en nous fondant sur le paradigme de segmentation de la morphologie mathématique: la ligne de partage des eaux du gradient à partir des marqueurs des régions que l'on veut extraire. Ce schéma a donné lieu à de nombreuses applications mais, dans le domaine du codage, cela mène à des difficultés réelles. La source de ces difficultés est la présence de détails fins, ou, ce qui revient au même, la présence de mouvement rapide qui génère des objets fins dans l'axe temporel. En effet, dans le domaine du codage, les conditions d'échantillonnage morphologique ne sont pas vérifiées. L'analyse des difficultés rencontrées nous a conduit à un autre paradigme de segmentation par fusion de régions.

Nous avons ainsi développé un algorithme de fusion de régions. Implémenté sur une structure de graphe dont les nœuds et les arêtes sont évalués, l'algorithme est à la fois efficace et suffisamment flexible pour prendre en compte une large variété de critères. Notamment, dans le but de rendre le système de codage globalement cohérent, nous avons pris en compte dans les critères de fusion les capacités des algorithmes de codage qui sont appliqués après la segmentation.

Finalement nous avons illustré le potentiel que la segmentation offre à un système de codage. La segmentation introduit la notion d'objet dans le codeur. Ainsi le développement de fonctionnalités, telles que le suivi d'objets, devient une extension naturelle des algorithmes de codage. Ces fonctionnalités rentrent dans le cadre des nouveaux objectifs fixés par la norme de codage vidéo MPEG 4.

Chapter 2

Description des outils morphologiques

Dans le cadre d'un système de codage orienté objet, la Morphologie Mathématique apparaît très attractive car c'est une approche géométrique du traitement du signal, et en tant que telle, elle permet de manipuler aisément des critères de forme, de taille, de connexité ... Les premiers concepts de la Morphologie Mathématique furent énoncés en 1901 par Minkowski. Sur cette base, Matheron et Serra, en établirent les fondements dans les années soixante. Depuis, cette théorie s'est largement développée et a été diffusée dans la communauté internationale. La prolifération de conférences, d'articles et d'applications industrielles en témoignent. La puissance de ses outils l'ont fait entrer dans de nombreux champs d'applications: imagerie médicale, sciences des matériaux, vision robotique, télédétection, analyse topographique, reconnaissance de caractères ... Les lecteurs intéressés par une connaissance approfondie de la Morphologie Mathématique peuvent se référer aux ouvrages [46, 47]. Dans ce chapitre nous allons seulement décrire les aspects pratiques de cette théorie. Ils nous permettront de mieux comprendre le développement des chapitres qui suivent.

La Morphologie Mathématique est une branche non linéaire du traitement du signal qui applique la théorie des ensembles à l'analyse d'image. Les objets de l'univers sont généralement tridimensionnels, mais ils sont perçus à travers une projection bidimensionnelle sur la rétine. Cette projection n'est pas la somme des luminances de tous les points le long d'une ligne de vision, ce qui ferait de la vision un processus linéaire: la plupart des objets sont opaques et cachent ainsi ceux qui sont derrière. Cette propriété justifie une approche non linéaire de l'analyse d'image. La Morphologie Mathématique, théorie intrinsèquement géométrique, introduit notamment la notion d'élément structurant. L'élément structurant est un objet de forme connue, qui est choisi en fonction des propriétés géométriques qu'on veut extraire de l'image. L'image à analyser est comparée à l'élément structurant. La Morphologie Mathématique établit un formalisme mathématique pour décrire quantitativement les propriétés géométriques d'un objet.

2.1 Transformations Morphologiques élémentaires

Note: Quand nous parlons d'ensembles nous faisons référence à des images binaires. Les images à teintes de gris sont décrites comme étant des fonctions de $\mathbf{R}^2 \rightarrow \mathbf{R}$.

2.1.1 L'Érosion

Soit X un sous-ensemble de $\mathbf{R}^2$ (une image binaire) et B un élément structurant. Un élément structurant est un sous-ensemble de $\mathbf{R}^2$. Le translaté de B par $\vec{p}$, noté B_p, est défini par:

$$B_p = \{p + q \ / \ q \in B\}$$

L'érosion de l'ensemble X par l'élément structurant B est définie comme l'ensemble des points p tels que B_p (translaté de B par $\vec{p}$) est contenu dans X.

$$\epsilon_B(X) = \{p \in \mathbf{R}^2, B_p \subseteq X\} = \bigcap_{b \in B} X_{-b}$$

Les effets géométriques de l'érosion sont:

- coupure des objets aux étranglements.
- rétrécissement des objets de taille supérieure à la taille de B.
- disparition d'objets de taille inférieure à B.

L'extension de cette définition à l'espace des fonctions ($\mathbf{R}^2 \rightarrow \mathbf{R}$) est obtenu en remplaçant l'opérateur $\cap$ par l'opérateur inf.

$$\forall p \in \mathbf{R}^2 \qquad \epsilon_B(f)(p) = \{inf(f(x)), x \in B_p\}$$

2.1.2 La Dilatation

Elle est définie comme l'ensemble des points p tels que l'intersection de B_p (translaté de l'élément structurant B par $\vec{p}$) avec X ne soit pas nulle.

$$\delta_B(X) = \{p \in \mathbf{R}^2, B_p \cap X \neq \emptyset\} = \bigcup_{b \in B} X_{-b} = \bigcup_{x \in X} \check{B}_x$$

La dilatation est la transformation duale de l'érosion. Dilater X est équivalent à éroder son complémentaire X^c:

$$\delta_B(X) = (\epsilon_B(X^c))^c$$

L'extension de cette définition à l'espace des fonctions $\mathbf{R}^2 \rightarrow \mathbf{R}$ est obtenue en remplaçant l'opérateur $\cup$ par l'opérateur sup.

$$\forall p \in \mathbf{R}^2 \qquad \delta_B(f)(p) = \{sup(f(x)), x \in \check{B}_p\}$$

L'érosion et la dilatation sont les briques de base pour la construction de nombreuses transformations morphologiques. Nous allons présenter dans la suite celles dont nous aurons besoin tout au long des prochains chapitres.

2.1.3 Le Gradient Morphologique

La segmentation d'images nécessite la quantification des variations de contraste en chaque point de l'image pour mettre en évidence les contours des objets, ou, ce qui revient au même, repérer les zones homogènes de l'image. Le gradient morphologique, défini par Beucher comme le dilaté moins l'érodé:

$$\Gamma(f) = \delta_B(f) - \epsilon_B(f)$$

nous fournit une mesure du module du gradient de la fonction (supposée continûment différenciable), classiquement défini comme:

$$\left|\vec{grad}(f)\right| = \left[\left(\ \frac{\partial f}{\partial x}\ \right)^2 + \left(\ \frac{\partial f}{\partial y}\ \right)^2\right]^{1/2}$$

Le gradient morphologique assigne à chaque point la plus grande différence entre les luminances de deux points de son voisinage.

2.1.4 L'Ouverture

Considérons l'érodé de l'ensemble X par l'élément structurant B. En général il n'est pas possible de récupérer l'ensemble initial moyennant une dilatation de l'ensemble érodé. L'érosion est une application non injective, c'est-à-dire que plusieurs ensembles $(X, Y, ...)$ peuvent avoir le même érodé $(X \ominus B)$. Le plus petit ensemble tel que son érodé est $X \ominus B$ est appelé l'ouvert de X, et il est obtenu en dilatant son érodé. La combinaison érosion suivie de dilatation est une "ouverture".

$$X \circ B = \gamma_B(X) \ = \ \delta_{\check{B}}(\epsilon_B(X))$$

où $\check{B}$ est l'ensemble transposé de B: $\check{B} = \{-x \ / \ x \in B\}$.

L'ensemble ouvert est une simplification de l'ensemble initial.

L'ouverture morphologique est une ouverture algébrique particulière. Le terme d'ouverture s'applique à d'autres transformations qu'une érosion suivie d'une dilatation. Il englobe toutes les transformation qui vérifient les propriétés algébriques suivantes:

- transformation croissante: elle préserve l'inclusion d'ensembles.

$$X \subseteq Y \ \Rightarrow \ (X \circ B) \subseteq (Y \circ B)$$

- transformation anti-extensive: le résultat est inclus dans l'ensemble de départ.

$$X \circ B \subseteq X$$

- transformation idempotente:

$$(X \circ B) \circ B = X \circ B$$

2.1.5 La Fermeture

En inversant l'ordre des opérations utilisées pour définir l'ouverture, nous obtenons une nouvelle opération: la fermeture.

$$X \bullet B = \phi_B(X) \; = \; \epsilon_{\check{B}}(\delta_B(X))$$

La fermeture est l'opération duale de l'ouverture. L'effet d'une fermeture est donc le même que celui de l'ouverture, mais appliqué au complémentaire.

$$(X^c \bullet B)^c = X \circ B$$

Les propriétés algébriques de la fermeture sont:

- transformation croissante: elle préserve l'inclusion d'ensembles.

$$Si \;\; X \subseteq Y, alors \; (X \bullet B) \subseteq (Y \bullet B)$$

- transformation extensive: l'ensemble de départ est inclus dans le résultat.

$$X \subseteq X \bullet B$$

- transformation idempotente:

$$(X \bullet B) \bullet B = X \bullet B$$

2.1.6 Interprétation de l'ouverture et de la fermeture

L'ouverture d'un ensemble X par un élément structurant B est la surface recouverte par B lors de son déplacement à l'intérieur de l'ensemble. Une propriété similaire s'applique à la fermeture, mais cette fois-ci, l'élément structurant parcourt le complémentaire de l'ensemble.

Par construction, l'ouverture et la fermeture respectent les quatre principes de toute transformation morphologique: invariance par translation, compatibilité avec les homothéties, connaissance locale et continuité.

2.1.7 Les chapeaux haut de forme (top-hat)

Le chapeau haut de forme est une transformation qui permet de détecter ce que l'ouverture (chapeau blanc), ou la fermeture (chapeau noir), ont fait disparaître. Etant donné un élément structurant B, on définit ainsi le chapeau haut de forme d'une fonction f par:

$$chapeau\ blanc: \;\; WTH_B(f) = f - \gamma_B(f)$$
$$chapeau\ noir: \;\; BTH_B(f) = \phi_B(f) - f$$

2.2 Les transformations géodésiques

Contrairement aux transformations euclidiennes qui agissent sur l'ensemble de l'image, les transformations géodésiques ne s'appliquent qu'à l'intérieur de zones prédéterminées, appelées masques géodésiques. Ainsi, dans toute transformation géodésique interviennent deux images: l'image à transformer X et le masque géodésique M.

Supposons que nous travaillons en connexité 4, c'est-a-dire, le voisinage d'un pixel (i,j) est:

$$V_4(i,j) = \{(i-1,j),(i,j-1),(i+1,j),(i,j+1)\}$$

Pour restreindre l'action de la transformation au masque géodésique la connexité qui s'applique à chaque pixel n'est pas $V_4(i,j)$, mais $V_4(i,j) \cap M$, c'est-à-dire qu'un pixel n'appartenant pas au masque ne sera pas considéré comme voisin d'aucun pixel.

Ces transformations sont à la base des opérations les plus performantes, comme par exemple la reconstruction.

2.2.1 La dilatation géodésique

Définition 2.1 *La dilatation géodésique de taille 1 de l'ensemble X dans le masque M, notée $\delta_M^{(1)}(X)$, est définie comme l'intersection du dilaté de taille 1 et de M.*

$$\delta_M^{(1)}(X) = (\delta_B^{(1)}) \cap M$$

où B est le voisinage d'adjacence pour la connexité choisie.

La dilatation géodésique de taille n de l'ensemble X conditionnée au masque M, $\delta_M^{(n)}(X)$, est obtenue en itérant n fois la dilatation géodésique de taille 1:

$$\delta_M^{(n)}(X) = \underbrace{\delta_M^{(1)}(\delta_M^{(1)}(...\delta_M^{(1)}}_{nfois}(X)))$$

Remarquons que la dilatation géodésique de taille n n'est pas l'intersection de la dilatation morphologique de taille n avec M.

$$\delta_M^{(n)}(X) \neq (\delta^n(X)) \ \cap \ M$$

Comme contre-exemple nous pouvons imaginer un ensemble M formé de plusieurs composantes connexes, dont l'une est l'ensemble X. La dilatation géodésique de n'importe quelle taille de X dans M sera toujours la composante initiale (l'ensemble X) tandis que l'intersection de la dilatation de taille n de X avec M peut contenir des parties d'autres composantes connexes.

L'extension de cette transformation pour les images à teintes de gris est directe. La dilatation géodésique de la fonction f dans la fonction masque g est définie de la façon suivante:

$$\delta_g^1(f) = inf(\delta(f), g)$$
$$\delta_g^{n+1} = inf(\delta(\delta_g^n(f)), g)$$

2.2.2 L'Érosion géodésique

L'érosion géodésique est la transformation duale de la dilatation géodésique. Son expression pour les images binaires est:

$$\epsilon_M^n(X) = M - \delta_M^n(M - X)$$

et pour les images à teintes de gris:

$$\epsilon_g^n(X) = -\delta_{-g}^n(-f)$$

2.2.3 La Reconstruction

Par construction, le résultat de la dilatation géodésique est inclus dans le masque. Par ailleurs, la dilatation est une opération extensive. Donc à partir d'une certaine itération la dilatation géodésique ne modifie pas l'image. La suite de dilatations géodésiques de taille croissante $(\delta_M^{(n)})$ converge vers une image appelée image reconstruite. La reconstruction de X (marqueur) conditionnellement à M (masque) est la dilatation géodésique de X dans M jusqu'à idempotence.

Le résultat de la reconstruction dans le cas binaire est l'ensemble des composantes connexes du masque qui coupent l'ensemble de départ; elle reconstruit les particules de M qui sont marquées par X.

Dans le cas numérique, la reconstruction de f dans g a comme seuls maxima les pics de g qui sont *marqués* par f.

Nous verrons plus loin que cette transformation nous permettra de changer l'homotopie de l'image gradient pour lui imposer les extrema souhaités, et éliminer ainsi la sursegmentation engendrée lors d'une application directe de l'algorithme de ligne de partage des eaux. En outre cette transformation est à la base des filtres connexes qui nous permettront de simplifier l'image sans dégrader le contour, condition nécessaire pour aboutir à une bonne segmentation.

2.3 Filtres connexes

Dans la section précédente nous avons vu que les combinaisons d'érosions et de dilatations constituent des filtres morphologiques (ouvertures et fermetures) qui éliminent les composantes connexes de taille inférieure à celle de l'élément structurant. Les ouvertures et fermetures peuvent aussi être combinées pour obtenir des filtres plus évolués. Cependant tous les filtres construits de cette manière, en même temps qu'ils éliminent les petites composantes, dégradent les contours des objets restants. L'introduction de la reconstruction donne une dimension supplémentaire au filtrage morphologique. Après l'élimination des composantes filtrées, les objets encore présents après le filtrage retrouvent leur contour original grâce à la reconstruction. Cette idée fut introduite en 1976 [20]: après une érosion, les composantes qui n'avaient pas complètement disparu étaient utilisées comme marqueurs pour reconstruire les particules originales. Cette opération, appelée ouverture par reconstruction, est le pionnier des filtres connexes. Ces filtres possèdent d'excellentes performances qui sont à l'origine d'un important développement du traitement d'images.

Commençons par décrire le cas binaire. Tout se passe comme si les filtres connexes considèrent chaque composante connexe individuellement. Chaque

composante connexe (soit de l'ensemble X , soit de son complémentaire X^c) est soit préservée intégralement, soit complètement effacée. Les composantes connexes sont donc des particules indivisibles. Cette caractéristique assure la conservation du contour des objets qui ont survécu au filtrage.

Définition 2.2 *Soit E un ensemble et $P(E)$ l'ensemble des sous-ensembles de E. Un opérateur Ψ:$P(E) \rightarrow P(E)$ est connexe quand $\forall A \subseteq E$, la différence symétrique entre A et $\Psi(A)$ est une union de composantes connexes de A ou de A^c.*

Intuitivement un opérateur connexe élimine des particules ou bouche des trous.

Pour étendre cette notion aux images à teintes de gris, commençons par définir quelques concepts.

Définition 2.3 *Les zones plates sont les plus grandes composantes connexes de l'espace E pour lesquelles la valeur de la fonction f est constante.*

Remarquons que lorsque le niveau de gris d'un pixel est différent de ceux de ses voisins, la zone plate associée à ce pixel est réduite à ce seul pixel.

Définition 2.4 *Soit E un ensemble et $P(E)$ l'ensemble des sous-ensembles de E. Une partition est une transformation τ: $E \rightarrow P(E)$ telle que:*
(1) $\forall x \in E \qquad x \in \tau(x)$
(2) $\forall x, y \in E \quad soit\ \tau(x) = \tau(y), \quad soit\ \tau(x) \cap \tau(y) = \emptyset$

La fonction qui à un point associe la zone plate qui le contient constitue une partition de l'image, car elle vérifie les deux conditions de la définition 2.4

Les entités préservées par un opérateur connexe appliqué à une fonction sont les zones plates de l'image. Les opérateurs connexes considèrent les zones plates comme des entités indivisibles. Par conséquent, une zone plate de l'image d'entrée sera contenue à l'intérieur d'une zone plate de l'image de sortie.

Définition 2.5 *Un filtre Ψ est connexe si, pour toute fonction f, la partition de zones plates de $\Psi(f)$ est moins fine que celle de f.*

Le gradient morphologique d'une image est égal à zéro au pixel p, si et seulement si, la boule élémentaire B_p est contenue dans la zone plate associée à p. La définition 2.5 implique que l'ensemble des zéros du gradient morphologique de $\Psi(f)$ contient celui du gradient de f.

2.3.1 Filtres par reconstruction

La reconstruction appliquée après une ouverture ou une fermeture restaure les composantes connexes (zones plates pour le cas numérique) qui n'ont pas complètement disparu. Ce sont par conséquent des filtres connexes. La combinaison d'ouvertures et fermetures par reconstruction élimine des détails clairs et sombres, tout en préservant les contours des objets. Cette opération simplifie la tâche de la segmentation et lui permet de rester fidèle aux données originales.

2.3.2 Filtres aréolaires

Les ouvertures et fermetures par reconstruction qu'on vient de présenter éliminent toute composante qui ne contient pas l'élément structurant. Ainsi la boule élémentaire, l'élément structurant de base qui contient les voisins directs d'un pixel, élimine toute structure dont une des dimensions est inférieure à 3 pixels, même si ce sont des structures longues et contrastées.

Lorsque les structures linéaires acquièrent une importance pour l'interprétation de l'image, des filtrages adaptés doivent être développés. Une première solution qui préserve les structures linéaires utilise le théorème démontré dans [47]:

Théorème 2.1 *Le supremum d'ouvertures est aussi une ouverture. De façon analogue l'infimum de fermetures est aussi une fermeture.*

L'ouverture définie comme le supremum d'ouvertures linéaires dans plusieurs directions (ou l'infimum de fermetures pour les détails sombres) préserve toutes les structures claires qui ne sont pas éliminées au moins dans une des directions. Cette procédure oblige à calculer une ouverture dans un grand nombre de directions (chaque orientation possible de l'élément structurant). Ceci multiplie le temps de calcul d'une ouverture par le nombre de directions utilisées. D'un autre côté la préservation des lignes courbes n'est pas assurée.

Une autre solution qui tient compte de toutes les directions (et formes) possibles sans augmenter le temps de calcul est fournie par les filtres aréolaires. Voyons les effets d'un filtre aréolaire sur une image binaire:

Définition 2.6 *Une ouverture aréolaire binaire de paramètre λ ($\gamma^a_\lambda(X)$) élimine toute composante connexe d'une image binaire dont l'aire (nombre de pixels) est inférieure au paramètre λ.*

Soit $(X_i)_{i\in I}$ l'ensemble des composantes connexes de X.

$$\gamma^a_\lambda(X) = \bigcup \{X_i \mid i \in I, Area(X_i) \geq \lambda\}$$

Remarquons que par définition les composantes connexes de l'image originale sont soit effacées soit préservées intégralement. Par conséquent on est face à un filtre connexe.

La fermeture aréolaire est définie par dualité comme:

$$\phi^a_\lambda(X) = [\gamma^a_\lambda(X^c)]^c$$

Dans le cas numérique le filtre aréolaire affecte les extrema de l'image d'aire inférieure à λ. Il a été démontré dans [55] que l'ouverture aréolaire de paramètre λ d'une image I est le supremum des images, inférieures à I et dont les maxima régionaux ont une surface supérieure ou égale à λ. Cette définition donne un algorithme efficace pour implémenter les filtres aréolaires. Voyons une expression plus formelle de cette définition.

Définition 2.7 *Soit f une fonction de $E \rightarrow \mathbf{R}$. L'ouverture aréolaire de taille λ de f est définie comme:*

$$(\gamma^a_\lambda(f))(x) = sup\{h \leq f(x) \mid x \in \gamma^a_\lambda(S_h(f))\}$$

où $S_h(f)$ est le seuillage de la fonction f au niveau h.

$$S_h(f) = \{x \in E \mid f(x) \geq h\}$$

Les filtres aréolaires préservent toute composante de taille (nombre de pixels) supérieure ou égale au paramètre λ, quelle que soit sa forme. L'élément structurant s'adapte aux formes de l'image.

2.4 La ligne de partage des eaux

L'algorithme de base de la Morphologie Mathématique pour segmenter des images est la ligne de partage des eaux. L'idée a été proposée par Lantuéjoul [23] et étendue au domaine de l'analyse d'images par Beucher [2]. Les lecteurs intéressés par une mise au point efficace de cet algorithme peuvent se reporter aux travaux de Meyer [29, 31], et de Vincent et Soille [56].

Notre but est de détecter les contours des objets présents dans l'image de la façon la plus précise possible. Imaginons les images à niveaux de gris comme des surfaces topographiques: plus le niveau de gris est clair, plus l'altitude est élevée. Si la valeur de luminance représente l'altitude, cela ne signifie pas que l'image puisse être découpée correctement par un simple seuillage. Un seuil élevé détectera bien les pics et un seuil bas extraira bien les vallées, mais cela nous oblige à parler en termes absolus, ce qui n'est pas cohérent puisque l'image peut ne pas être éclairée uniformément. L'information de la luminance n'étant pas pertinente, nous allons recourir à la fonction gradient qui nous donne une information sur les variations de luminance dans l'image. La façon la plus simple d'approximer ce gradient a été décrite par Beucher [2] et rappelée dans la section 2.1 de ce document. À chaque pixel x on assigne la plus grande différence entre les luminances de deux pixels de son voisinage, c'est-à-dire la différence entre le dilaté et l'érodé de la fonction.

Le gradient morphologique est une image numérique, dans laquelle les plus hautes valeurs correspondent aux contours les plus contrastés de l'image originale. Nous retrouvons ici le même problème qu'avec la luminance. Un seuillage faible donnera une forte segmentation, tandis qu'un seuillage haut perdra des régions peu contrastées. Le seuillage du gradient présente également le défaut de fournir des contours ouverts. Il n'est pas facile de trouver les contours à partir du gradient, mais on peut aisément trouver des régions ne contenant pas de contours. Les minima régionaux du gradient (plateaux d'altitude uniforme sans voisins plus bas) marquent les régions de plus faible variation de luminance par rapport aux environs. Par conséquent les contours ne passent pas dans ces régions. Si on prend une interprétation topographique de la fonction gradient et qu'on imagine la pluie qui tombe sur la surface, on verra que si une goutte tombe sur le point x, elle descendra jusqu'à se trouver dans le fond d'un bassin, c'est-à-dire un minimum de la fonction (voir fig. 2.1). Ainsi chaque minimum a sa propre zone d'attraction constituée par des points à partir desquels ce minimum est atteint. Cette zone d'attraction est appelée bassin versant. Deux bassins versants voisins possèdent une frontière commune sur laquelle une goutte peut choisir le minimum vers lequel elle va descendre. Cette ligne est la ligne de crête et il semble logique de la considérer comme un contour idéal.

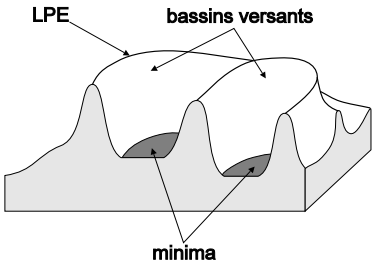


Figure 2.1: Ligne de partage des eaux.

On peut illustrer ce même concept par analogie avec une inondation plutôt que par la pluie qui tombe. On en tire un algorithme plus facile à implémenter. Imaginons donc que cette surface topographique soit trouée aux emplacements des minima. Plongeons lentement cette surface dans un lac. L'eau va passer par les trous en commençant par ceux qui percent les minima les plus profonds et va progressivement inonder le relief. A tout moment les différents lacs seront à la même altitude. Supposons de plus que l'on empêche les eaux provenant de lacs différents (donc de minima différents) de se mélanger, en construisant sur la surface topographique une digue là où ces eaux pourraient se rejoindre. Lorsque la totalité de la surface aura été couverte, seules les digues émergeront, délimitant autant de lacs que la fonction inondée (le gradient) possédait de minima. Ces digues constituent ce qu'on appelle la ligne de partage des eaux [3].

L'application directe de la ligne de partage des eaux sur une image réelle produit une forte sursegmentation car la détection des extrema est très sensible au bruit. L'introduction des marqueurs [35] signalant les régions importantes d'une image résout le problème de la sursegmentation et rend l'algorithme applicable aux cas réels. L'imposition des marqueurs fait disparaître la sursegmentation parce que les bassins versants non marqués sont inondés par l'eau provenant d'un bassin versant marqué sans construire de digue entre les deux (voir fig. 2.1). La ligne de partage des eaux sera placée sur la ligne de crête la plus élevée du gradient entre deux marqueurs, ce qui correspond au contour entre deux régions significatives.

La ligne de partage des eaux requiert une fonction de marquage pour éviter la sursegmentation. La qualité de la segmentation sera directement liée à cette fonction de marquage.

2.5 La dynamique

Le gradient d'une image à niveaux de gris a une énorme quantité de minima, ce qui produit une sursegmentation de l'image lorsqu'on calcule directement une ligne de partage des eaux. La dynamique est une mesure de contraste introduite par M. Grimaud [15, 16]qui nous permet de classer les minima pour pouvoir choisir les plus significatifs comme marqueurs des régions importantes et ainsi arriver à une segmentation propre.

Commençons par donner quelques définitions sur lesquelles nous nous appuierons pour définir la dynamique.

Définition 2.8 *Un chemin C(x,y) sur une image f est une suite de pixels ad-*

jacents (p_0, p_1, p_2, ..., p_n) qui relient les pixels x *et* y*, tels que:*

- p_0 *est le pixel* x*, et* p_n *est le pixel* y*.*
- *Pour tout i ∈ [0,n], les pixels* p_i *et* p_{i+1} *sont voisins dans la grille utilisée.*

Définition 2.9 *La dynamique d'un chemin C(x,y) qui rejoint les pixels x et y sur une image f, est la différence d'altitude entre la plus grande et la plus petite altitude rencontrée sur le chemin (fig. 2.2).*

$$Dyn_f(C(x,y)) = \{sup(abs(f(x_i) - f(x_j))) \quad \forall x_i, x_j \in C(x,y)\}$$

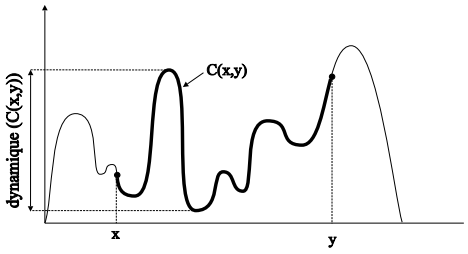


Figure 2.2: Dynamique d'un chemin

Définition 2.10 *La dynamique entre deux points* x *et* y *est la plus petite des dynamiques des chemins qui rejoignent* x *et* y*.*
$Dyn_f(x,y) = inf(Dyn_f(C(x,y)))$ *pour tout chemin C(x,y) qui rejoint x et y*
En une dimension cette définition est la même que celle de la dynamique d'un chemin, mais en deux ou trois dimensions c'est différent puisqu'il y a plusieurs chemins entre deux points.

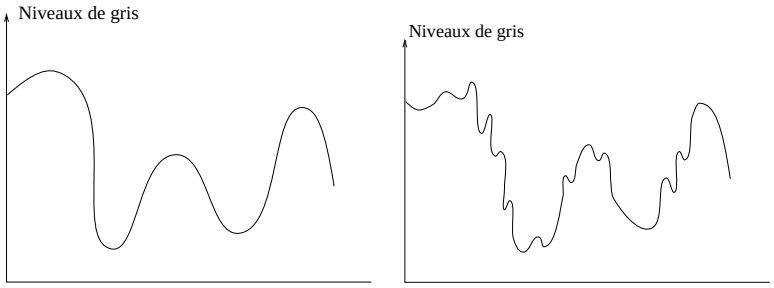
Finalement nous arrivons à la notion qui nous intéresse: la dynamique des extrema.

Nous pouvons penser que le contraste d'un minimum est défini par la profondeur du bassin versant auquel il appartient. Cette profondeur peut être estimée par la différence d'altitude entre le point le plus bas du contour de son bassin versant et le minimum. Cette estimation est bonne avec une image non bruitée comme celle de la figure 2.3a, mais pose des problèmes avec une image comme celle de 2.3b, dans laquelle la vallée est composée de nombreux bassins versants dont les profondeurs ne donnent pas une bonne idée du contraste de la structure. Cette figure nous montre que nous ne pouvons pas travailler à l'échelle de bassin versant pour estimer le contraste des structures.

Définition 2.11 *La dynamique d'un minimum* M *est la plus petite des dynamiques des chemins qui lient* M *à un point* y *appartenant à un bassin versant dont le minimum est strictement plus bas que* M*.*

$$Dyn_f(M) = inf(Dyn_f(x,y), x \in M, y \in BV/altitude(M) > minimum(BV(y)))$$

si on note *BV(y)* le bassin versant du point y.



(a) Image nette (b) Image bruitée

Figure 2.3:

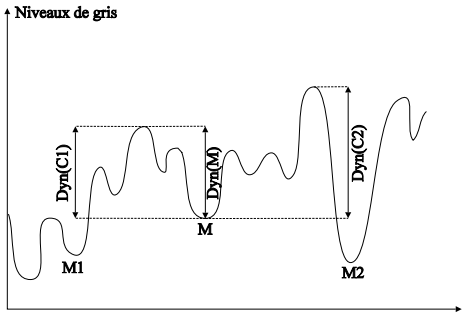


Figure 2.4: Dynamique d'un minimum.

Nous allons illustrer cette définition avec un profil de niveau de gris (fig. 2.4). Nous cherchons la dynamique du minimum M. A une dimension nous n'avons que deux chemins à tester, vers la gauche (chemin C_1) ou vers la droite (chemin C_2), jusqu'au premier bassin versant dont le minimum est strictement plus bas que M. La dynamique du chemin C_1 est plus petite, donc on la retient comme étant la dynamique du minimum M.

Une autre façon plus intuitive d'exprimer cette idée est la suivante:
Nous allons appeler "apogée" le point le plus haut d'un chemin. Parmi tous les chemins qui relient le minimum M avec des pixels d'altitude plus petite que M, nous allons considérer celui d'apogée minimum AP_{min}. La dynamique de M sera la différence entre AP_{min} et le minimum M. Les deux définitions sont équivalentes.

La dynamique d'un minimum est définie à partir des minima plus bas que ce minimum. Par conséquent, la dynamique du minimum le plus bas de l'image n'est pas définie. Par convention nous dirons qu'elle est infinie. Ceci ne pose pas de problèmes, parce que la valeur de la dynamique du minimum n'est pas importante en elle même. Cette valeur n'a d'importance que par comparaison avec celles des autres minima de l'image. Ainsi une valeur infinie signifie tout simplement que ce minimum a une dynamique plus grande que tous les minima à dynamique finie. En pratique nous allons lui assigner la dynamique de l'image, c'est-à-dire la différence entre le maximum et le minimum de l'image.

La mesure de la dynamique des minima est une généralisation de la notion de *h-minima*. Les *h-minima* d'une fonction f, sont les minima de la fonction $Rec(f+h, f)$. Cette transformation assure un contraste d'au moins h pour tout minimum détecté. L'approche par dynamique a l'intérêt de fournir l'information de contraste de tous les minima de l'image, ce qui permet de les hiérarchiser. Le seuillage de la dynamique au niveau h est équivalent à l'extraction des h-minima.

Nous pouvons remarquer que la dimension spatiale n'est pas considérée pour évaluer le contraste. Les longueurs des chemins C_1 et C_2 n' interviennent pas pour calculer la dynamique. La structure marquée par un minimum M peut être grande ou petite, le résultat n'est pas modifié. Ceci est nouveau dans les transformations morphologiques. Avec un opérateur morphologique comme par exemple le top-hat, nous aurions besoin de connaître a priori une approximation de la taille de la structure pour évaluer le contraste.

Chapter 3

Généralités sur les séquences

L'objectif principal de cette thèse est de développer des algorithmes pour traiter des séquences d'images. Voyons dans ce chapitre la description générale du cadre de travail.

Une séquence d'images est une succession d'images bidimensionnelles qui montre l'évolution temporelle d'une scène. La cadence est de 25 images / seconde, ce qui correspond au seuil à partir duquel l'œil humain perçoit la séquence comme un stimulus continu, grâce à la persistance rétinienne. Par la suite, nous appellerons "trame" ou "plan" chaque image bidimensionnelle correspondant à un instant donné de la séquence.

L'évolution du monde réel est relativement lente, comparée à la fréquence d'échantillonnage temporel, ce qui implique une forte redondance temporelle. Ainsi, si nous nous intéressons à coder la séquence de la manière la plus compacte possible, on a intérêt à utiliser l'information qui décrit les changements d'une image par rapport à la précédente, plutôt que l'information qui décrit leur contenu.

3.1 Différentes approches pour traiter la dimension temporelle

Plusieurs méthodes ont été testées pour traiter la dimension temporelle. Ces méthodes sont décrites dans les paragraphes suivants.

3.1.1 2D + 2D

La première approche utilise un algorithme bidimensionnel pour chaque plan de la séquence et produit une segmentation indépendante pour chaque trame. Ensuite les régions des trames successives sont mises en correspondance pour ne coder à chaque instant que les différences par rapport à la trame précédente ou pour ajouter au système de codage d'autres fonctionnalités comme par exemple une application de suivi d'objets. Pour plus de détails sur le suivi d'objets se reporter au chapitre 11.

Malgré la ressemblance des images originales consécutives, les segmentations calculées de manière indépendante sont différentes et les régions se correspondent mal. Voyons sur la figure 3.1 deux images extraites de la même séquence.

Leur aspect visuel est très similaire mais pourtant les segmentations respectives sont différentes: l'une produit plus de régions dans le visage tandis que l'autre garde plus de détails sur la veste. Le classement des régions suivant leur importance visuelle varie légèrement d'un temps à l'autre ce qui fait que quand on choisit les plus importantes, certaines apparaissent et disparaissent selon la trame. Avoir de telles fluctuations de régions augmente le coût de codage et provoque un papillotement de l'image désagréable pour la vue.

Par ailleurs, même si les mêmes régions sont choisies à différents instants, leurs contours respectifs sont calculés de manière indépendante, provoquant également des fluctuations. La visualisation de la séquence produit un effet de fourmillement sur le contour, ce qui produit une impression visuelle gênante ainsi qu'un codage peu économe.

(a) Image Originale 1

(b) Image 1 Segmentée

(c)Image 1 Codée avec la moyenne

(d) Image Originale 2

(e) Image 2 Segmentée

(f)Image 2 Codée avec la moyenne

Figure 3.1: Instabilité des algorithmes bidimensionnels appliqués à une séquence

3.1.2 Approche Tridimensionnelle

Dans la bibliographie nous pouvons trouver des applications qui traitent une séquence temporelle comme une image tridimensionnelle. Tel est le cas de Friedlander [13] qui segmente des images de cardiologie nucléaire. Au lieu de traiter chaque image séparément, il empile les images dans l'ordre chronologique et considère une connexité 3D, c'est-à-dire que le voisinage d'un pixel du temps t comporte aussi bien des pixels de la même trame, que des pixels de l'image précédente $t-1$ et de l'image suivante $t+1$. Toutes les opérations (filtrage, laplacien, ligne de partage des eaux...) sont réalisées en connexité 3D, c'est-à-dire, le traitement d'un pixel tient compte de son passé et de son futur, ce qui force une cohérence temporelle donnant robustesse à l'algorithme.

Une approche tridimensionnelle résout donc les problèmes d'instabilité dus au traitement séparé de chacune des images de la séquence. Elle force les régions

à être cohérentes tout au long de la séquence, tant qu'elles sont en scène. Les régions sont tridimensionnelles (volumiques), ce qui nous donne l'information de la position et de la forme de toutes les régions à chaque instant de la séquence.

Le principal inconvénient de cette approche est lié au fait qu'il faut traiter toute la séquence avant de commencer à la transmettre. Ceci se traduit par un retard important qui n'est pas adapté aux applications en temps réel, comme par exemple la vidéotéléphonie. Par ailleurs, les besoins de mémoire de cette approche sont très importants.

3.1.3 Approche récursive

L'approche récursive suppose connue la segmentation de l'image $t-1$ et l'utilise pour segmenter l'image t. Le résultat t connu, il est utilisé pour traiter $t+1$, qui est utilisé pour traiter $t+2$ et ainsi de suite. La première image doit alors être initialisée, par exemple par un algorithme bidimensionnel.

Cette approche est utilisée dans le cadre du projet PROMETHEUS [58]. L'objectif de ce projet est de segmenter des scènes routières pour une application de conduite automatique. La segmentation de la route se réalise de la manière suivante: supposons que l'image $t-1$ a déjà été segmentée (S_{t-1}). Pour segmenter la route au temps t, on utilise la segmentation précédente S_{t-1}: les régions du temps $t-1$ sont érodées (pour s'assurer qu'elles soient à l'intérieur de leur nouvelle position au temps t) et introduites au temps t comme marqueurs des régions à segmenter. Le système est initialisé grâce à une segmentation bidimensionnelle.

Cette méthode marche très bien dans le cadre des scènes routières. Le problème est simplifié par rapport au notre: on connaît a priori les régions qu'on cherche à segmenter (la route, le ciel) ainsi que leur emplacement (la route en bas, le ciel en haut, si la voiture roule encore). Les régions sont suffisamment grandes pour ne pas craindre que leur érodé soit en dehors du nouvel emplacement. Malheureusement ce n'est pas notre cas. Certaines de nos régions peuvent disparaître ou se déconnecter temporellement à cause de l'érosion ce qui nous empêche de les utiliser comme marqueurs.

Néanmoins utiliser le résultat précédent pour segmenter le temps courant est une façon intéressante d'adresser le problème de la segmentation de séquences d'images. D'une part c'est une approche qui n'introduit pas de retard, ce qui permet de développer des applications interactives, et d'autre part les besoins de mémoire sont réduits (par rapport à une approche tridimensionnelle). Par ailleurs elle impose une stabilité temporelle intéressante pour certaines applications, en particulier le codage.

Dans une approche récursive, à chaque instant nous voyons la séquence à travers une fenêtre d'épaisseur deux: une image déjà traitée et une image à traiter. La fenêtre glisse au long de la séquence au fur et à mesure que les images sont traitées. Contrairement au projet PROMETHEUS [1, 58], nous n'érodons pas les régions précédentes mais nous les plaçons en face du temps courant dans une image tridimensionnelle. Ensuite, en utilisant soit une procédure de ligne de partage des eaux, soit de croissance de régions, soit de fusion de régions, les régions du temps $t-1$ prennent place au temps courant. Le passage des régions du temps $t-1$ au temps t a été appelé projection par M. Pardàs. Elle a obtenu de bons résultats en utilisant la croissance de régions [39]. Nous présenterons dans cette thèse une autre méthode de projection qui permet l'introduction

directe de nouveaux objets de la scène. Mais, quelles que soient les modalités particulières, tous les travaux cités ont montré que c'est la meilleure manière d'imposer une cohérence temporelle à la segmentation de séquences d'images tout en permettant le mouvement et la déformation des objets présents dans la scène.

La figure 3.2 illustre cette procédure. Nous avons la segmentation S_{t-1} qui comporte trois régions. L'image $t-1$ et l'image t sont réunies dans une image tridimensionnelle et les régions de S_{t-1} s'étendent au temps t par une procédure de croissance de marqueurs. Une fois la segmentation S_t obtenue, la fenêtre $(t-1,t)$ glisse vers les temps $(t,t+1)$ et on itère la procédure.

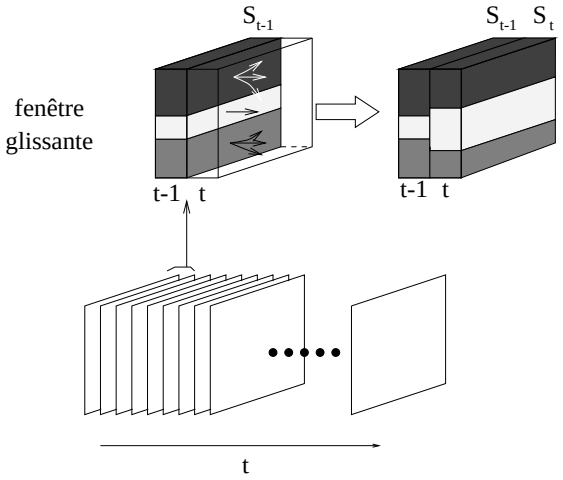


Figure 3.2: Fenêtre glissante

L'approche récursive remplit les conditions requises par notre application: stabilité temporelle sans retard. Par conséquent, elle sera la procédure utilisée dans les chapitres qui suivent.

Nous avons décidé de traiter les séquences d'images avec un algorithme récursif. Cela implique de traiter les images deux par deux. Voyons quelles sont les différences possibles entre deux plans consécutifs. En effet, ces différences déterminent les conditions dans lesquelles l'algorithme doit travailler. Etudions en détail ces conditions ainsi que le comportement de l'algorithme dans chacune d'elles.

3.2 Les différences entre deux plans successifs d'une séquence

Les différences entre plans successifs d'une séquence sont dues à trois causes:

1. déplacement et déformation d'objets présents dans la scène. Cette situation est correctement traitée par l'algorithme que nous venons de décrire. La seule condition nécessaire pour récupérer au temps t la position et la forme d'un objet ancien est que les positions respectives de l'objet au temps $t-1$ et au temps t aient une intersection non nulle. Cette intersection, caractérisée par un contraste temporel bas, favorise la propagation

des marqueurs dans la direction temporelle, ce qui permet aux objets précédents de se placer au temps courant. Par contre, si le déplacement est important, cette intersection peut ne pas exister (cela arrive surtout dans le cas de petites régions). Dans ce cas, l'objet se trouve isolé au temps $t-1$, entouré de fortes barrières contrastées, ce qui l'empêche d'atteindre le temps courant. Tel est le cas de la région 2 sur la figure 3.3.

L'absence de connexion temporelle empêche la transmission de l'information du passé au présent, ce qui mène à une des deux situations suivantes: soit la région disparaît de la segmentation, avec la perte de qualité que cela entraîne, soit la région doit être codée comme si on ne la connaissait pas, avec une augmentation du coût de codage. Les deux conséquences sont négatives pour les algorithmes de codage; nous verrons dans la section 3.3 de possibles améliorations pour traiter ce type de déconnexions.

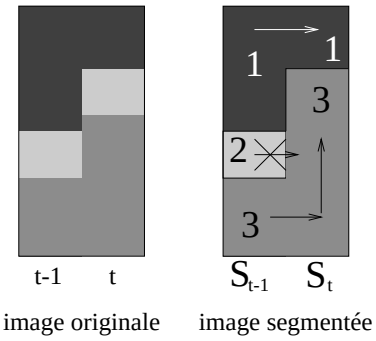


Figure 3.3: Déconnexion temporelle

2. disparition de régions. Si une région disparaît, son espace est remplacé soit par une région voisine qui se déforme, soit par une nouvelle région qui apparaît. S'il s'agit de la déformation d'une région voisine le cas est abordé par l'algorithme de la fenêtre glissante décrit précédemment. Le processus est le suivant: la région qui s'est déformée atteint le temps t par une zone de bas contraste et ensuite par propagation spatiale rempli l'espace libérée par la région qui a disparu. Pour mieux comprendre ce mécanisme regardons la figure 3.4. La segmentation S_{t-1} comprend trois régions dont une disparaît au temps t. L'espace qu'elle occupait est rempli au temps t par propagation spatiale de ses régions voisines.

   Par contre si la région disparue est remplacée par de nouvelles régions nous sommes face à une situation qui requiert un traitement spécial (voir point suivant).

3. apparition de nouvelles régions. C'est une situation qui ne peut pas faire appel à l'information du passé, par conséquent elle doit être codée entièrement. De toute façon, même si les nouvelles régions sont plus chères à coder, nous devons leur laisser la porte ouverte dans la segmentation, pour être capables de suivre correctement l'évolution de la scène.

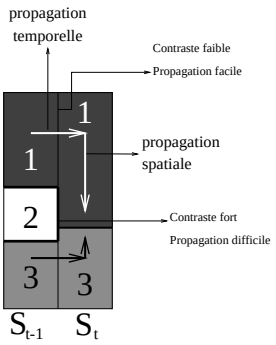


Figure 3.4: Disparition d'une région

3.3 Le mouvement

La plupart des changements entre deux trames consécutives d'une séquence peuvent être décrits en termes du mouvement des objets présents dans la scène. Coder l'information de mouvement nous permet d'éliminer une grande partie de la redondance temporelle et, en conséquence, d'obtenir des taux de compression élevés.

Dans le cadre des projets européens (MORPHECO et MAVT) ont été développés des algorithmes d'estimation de mouvement orientés objet, c'est-à-dire, des estimations de mouvement basées sur les résultats de la segmentation à deux instants de la séquence. Ces algorithmes sont chargés de calculer les paramètres de mouvement (translationnel, affine, quadratique ...) qui décrivent au mieux la transformation de l'objet d'une trame à l'autre. Le fait de travailler avec des objets (supposés être des entités physiques) améliore les performances par rapport aux algorithmes qui travaillent avec des blocs. Un bloc peut se placer à cheval entre deux objets de mouvements indépendants ce qui fausse son estimation de mouvement.

Nous venons de décrire la collaboration segmentation-estimation de mouvement. C'est le choix du projet MORPHECO (dans le cadre duquel nous avons développé nos algorithmes): d'abord segmentation et ensuite estimation de mouvement. Néanmoins la collaboration inverse a aussi été étudiée. Sanson [45] commence par estimer le mouvement entre deux trames et ensuite il segmente le champ de vecteurs de mouvement obtenu.

Ce sont deux choix différents qui mènent à des résultats cohérents tous les deux. La conclusion qui peut en être tirée est que la segmentation et l'estimation de mouvement sont deux procédés qui peuvent collaborer pour améliorer leurs performances respectives.

Dans notre cas nous sommes capables de suivre les objets qui ont une intersection non nulle entre leur positions respectives à deux instants différents, mais nous avons des problèmes quand un mouvement rapide déconnecte temporellement un objet. Les algorithmes d'estimation de mouvement peuvent nous aider à résoudre ce problème.

Nous avons un couple d'images consécutives et nous allons les regrouper dans une image tridimensionnelle. Comme nous avons vu, les différences entre les deux images peuvent être dues à l'apparition et la disparition de régions et

au déplacement et déformation d'autres. L'apparition des régions correspond à des changements dans la scène qui ne peuvent pas être prédits, ce qui oblige à les coder entièrement. Par contre le déplacement et la déformation des régions peuvent être prédits à partir de l'image précédente [38]. Pour améliorer les performances de la segmentation nous pouvons estimer le mouvement entre les deux trames et compenser en mouvement l'image $t-1$, c'est-à-dire, lui appliquer les vecteurs de mouvement calculés. De cette façon on obtient une image avec l'information du passé qui ressemble le plus possible à l'image courante. Si maintenant on regroupe S_{t-1} compensée (au lieu de S_{t-1}) avec l'image t on réduit les possibilités de déconnexion d'objets à cause du mouvement et en conséquence on obtient une segmentation plus stable dans le temps, ce qui nous mène à des taux de compression plus élevés. Nous décrivons en détail cette procédure dans le chapitre 9.

3.4 Structure générale d'un système de codage récursif orienté objet

Dans un système de codage récursif nous pouvons distinguer deux modes de fonctionnement:

- mode INTRA-FRAME

  Ce mode de fonctionnement est utilisé pour coder la première image d'une séquence et les images de rafraîchissement qui ont lieu régulièrement au cours de la séquence pour éviter une trop grande erreur cumulée. Comme son nom l'indique, pour coder un plan, le mode INTRA utilise seulement l'information présente dans ce plan. Le processus est entièrement bidimensionnel, l'information des plans antérieurs et postérieurs n'est pas utilisée.

- mode INTER-FRAME

  Ce mode utilise l'information d'autres plans pour coder la trame courante; plus précisément on utilise l'information de l'image précédente. De cette façon on exploite les connaissances déjà acquises concernant la scène et on code seulement les changements par rapport au temps précédent.

L'exploitation de la corrélation temporelle existante dans une séquence d'images est la clé pour obtenir des taux de compression élevés. En mode *intra* la scène est complètement inconnue et doit être entièrement décrite: une partition de l'image est obtenue et tous les objets sont codés à travers leur contour et leur contenu. A partir de la deuxième image, le codage sera efficace si l'information connue n'est pas retransmise. En mode *inter* on ne codera que les changements par rapport au temps précédent. Le codage par un vecteur de mouvement est une façon efficace de traiter l'évolution d'un objet dans une séquence: une fois l'objet connu, il suffit de savoir où le placer dans l'image. Des estimations de mouvement basées sur la notion d'objet ont été développées dans le cadre du projet *MORPHECO*.

La figure 3.5 montre la structure générale du système de compression dans lequel s'encadre notre travail. Nous pouvons distinguer trois étapes: (1) codage, (2) transmission et (3) décodage. Le codage à son tour comprend 4 étapes:

1. segmentation: décomposition de l'image en régions correspondant autant que possible aux objets présents dans la scène. Celui-ci est l'objectif de cette thèse.

2. estimation de mouvement: une fois les objets segmentés, leur mouvement de la trame précédente à la trame courante est estimé [25] pour exploiter la redondance temporelle. Le codage des paramètres de mouvement génère le train numérique de mouvement.

3. codage de contour: les vecteurs de mouvement sont appliqués à la segmentation précédente. Les différences de contour entre l'image ainsi générée et l'image segmentée courante sont transmises dans le le train numérique de contour.

4. codage de texture: les vecteurs de mouvement sont appliqués à l'image codée précédente et l'erreur de prédiction est codée dans le le train numérique de texture. On obtient ainsi l'image codée courante.

Dans la suite nous utiliserons le terme anglais bitstream pour désigner le train numérique.

Les trois bitstreams (mouvement, contour, texture) sont envoyés à travers le canal au récepteur qui décode l'image de la manière suivante:

1. le bitstream de mouvement est décodé et les vecteurs de mouvement ainsi générés sont appliqués, d'une part à la segmentation précédente générant la "segmentée" précédente compensée et d'autre part à l'image "décodée" précédente, générant la décodée précédente compensée.

2. le bitstream de contour est décodé. Les erreurs de prédiction par rapport à la segmentée précédente compensée sont corrigées, obtenant ainsi la segmentée courante.

3. le bitstream de texture est décodé. Les erreurs de prédiction par rapport à la codée précédente compensée sont corrigées obtenant l'image décodée courante.

Notre travail consiste à obtenir une segmentation de la séquence stable dans le temps pour pouvoir exploiter la redondance temporelle. Nous avons commencé notre travail en nous fondant sur le paradigme classique de la morphologie mathématique: segmentation = ligne de partage des eaux de l'image gradient à partir des marqueurs des régions que l'on veut extraire. Ce schéma a été utilisé avec succès dans de nombreuses situations. Cependant son application au codage mène à des difficultés réelles lorsque les conditions de l'échantillonnage morphologique ne sont pas réunies, c'est-à-dire, en présence de détails fins ou, ce qui revient au même dans l'axe temporel, en présence de mouvement rapide. Nous analyserons ces difficultés et nous verrons comment elles nous ont mené vers un autre paradigme de segmentation, par fusion de régions [36].

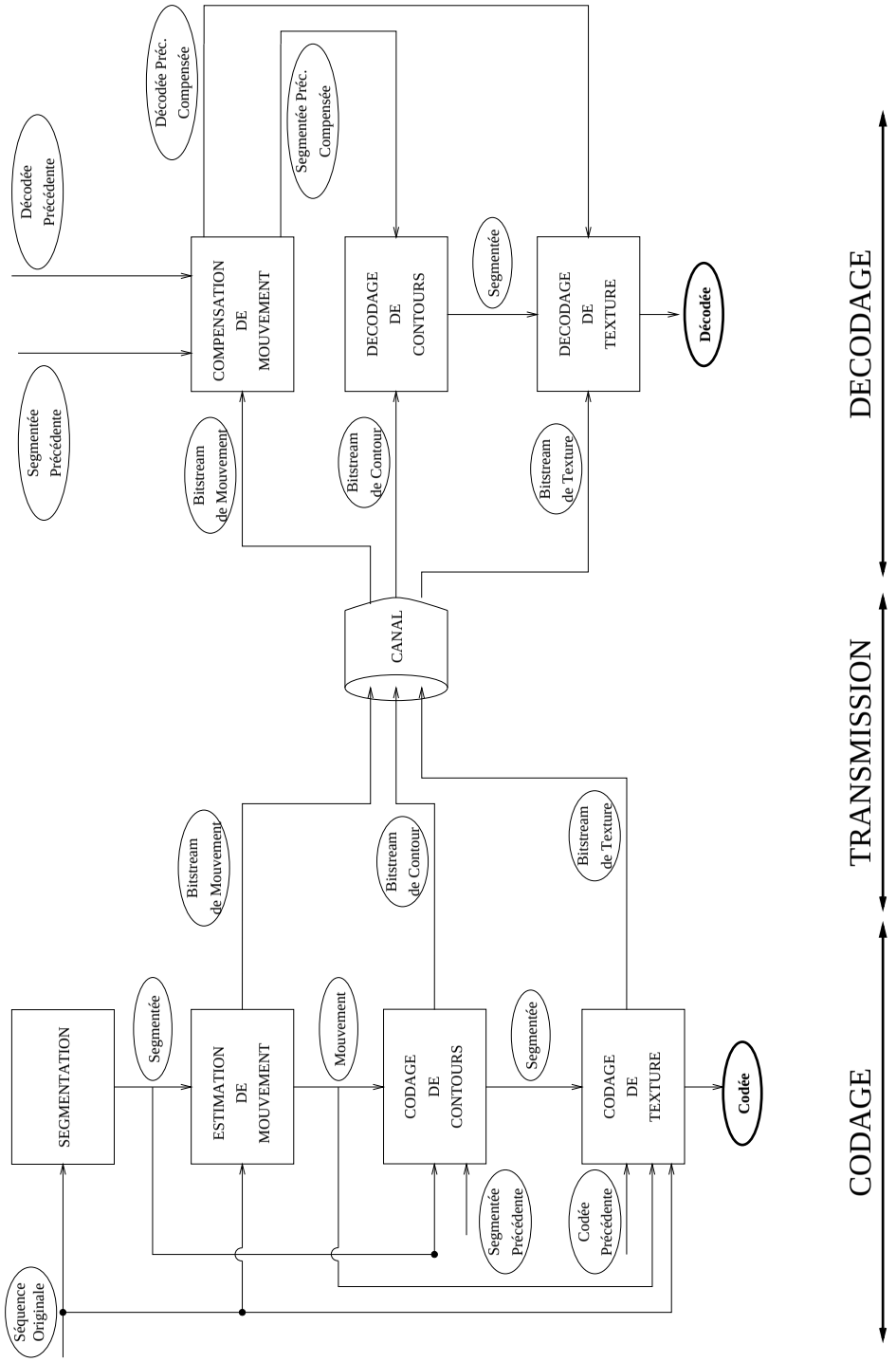


Figure 3.5: Diagramme de blocs d'un système de codage orienté objet

Chapter 4

Segmentation 2D basée sur la ligne de partage des eaux classique

Ce chapitre est consacré à la segmentation d'images réelles par des méthodes morphologiques classiques: gradient + imposition de marqueurs + ligne de partage des eaux. Cette approche a résolu de nombreux problèmes d'analyse d'image, dans des domaines d'application variés, parmi lesquels nous pouvons citer les suivants:

- imagerie médicale: détection des cellules cancéreuses [28], détection de vertèbres dans des images topographiques [41], détection automatique des opacités du sein [52].
- sciences des matériaux: analyse de matériaux composites [22].
- analyse des milieux poreux [17].
- agro-alimentaire: analyse de processus de fermentation [24].
- contrôle: détection automatique [19] pour la réalisation d'un système de sanction automatique dans le cadre du contrôle non destructif de pièces coulées.
- applications tridimensionnelles: mesure des propriétés d'une mousse de polyuréthane [14], extraction du contour de la matière grise du cerveau.
- cardiologie nucléaire, où la segmentation est spatio-temporelle [13].
- conduite automatique (aussi spatio-temporelle): détection de la route à suivre et des possibles obstacles à éviter [1, 58]. Cette application a abouti à une implémentation hardware en temps réel.

Toutes ces applications connaissent a priori les caractéristiques des éléments à extraire de l'image et cette information est utilisée pour définir l'algorithme de sélection des marqueurs. Notre étude est orientée vers une application de codage. Les images à traiter sont aussi diverses que le monde qui nous entoure

et nous ne pouvons pas faire d'hypothèse de taille, de forme, ou de niveau de gris sur les objets d'intérêt existant dans l'image. Par conséquent, les critères de sélection de marqueurs doivent rester généraux, sans utiliser d'information a priori sur l'image traitée. Dans ce chapitre nous présentons un critère de sélection des objets les plus visibles d'une image. Nous verrons par la suite les limitations de cette approche basée sur le gradient, limitations qui vont nous pousser à ouvrir une autre voie de recherche dans les chapitres qui suivent.

Cet algorithme de segmentation morphologique, dans une version n'utilisant pas de connaissance a priori sur l'image, peut être décomposé en trois étapes:

1. Une étape de simplification qui élimine les traits non pertinents de l'image, simplifiant ainsi la détection des régions homogènes.

2. Une étape d'extraction de marqueurs: étape qui a pour but de placer un *marqueur* à l'intérieur de chaque région homogène. Les contours de ces régions seront déterminés dans l'étape suivante.

3. Une étape de croissance de marqueurs: le but est de placer de façon précise les contours des objets sélectionnés par l'étape précédente. Cette croissance est réalisée par l'algorithme de ligne de partage des eaux appliqué à l'image gradient.

Voyons en détail chacune de ces trois étapes.

4.1 Simplification de l'image

Les images captées par une caméra sont bruitées, ce qui rend difficile la détection de zones homogènes. Un filtrage préalable simplifie la tâche de la segmentation. La Morphologie Mathématique possède des outils très performants dans ce domaine [48]. Les filtres morphologiques qui ont été présentés dans le chapitre précédent vont maintenant être appliqués à des images réelles.

Commençons par illustrer l'effet des ouvertures-fermetures par reconstruction. Les ouvertures par reconstruction éliminent tout détail clair qui ne contient pas l'élément structurant, préservant les contours des régions qui le contiennent. L'effet des fermetures est le même mais sur les détails sombres. Ainsi, la combinaison d'ouvertures et fermetures élimine toute structure (claire ou sombre) de taille inférieure à l'élément structurant.

Voyons quelques exemples: le résultat de l'application d'une ouverture suivie d'une fermeture de taille 1, à l'image de la figure 4.1(a) est présenté dans la figure 4.1(b). L'image est au format qcif, c'est-à-dire de taille 176×144 pixels. Après filtrage, l'information pertinente est préservée et les 16717 zones plates (composantes connexes de niveaux de gris constant) de l'image originale ont été réduites à 10460.

L'ouverture-fermeture par la boule élémentaire (élément structurant de base qui contient les premiers voisins d'un pixel) élimine toute structure dont une des dimensions est inférieure à trois pixels. Ces structures correspondent la plupart du temps au bruit et leur disparition facilite la segmentation.

(a) Image Originale (b) Image filtrée

Figure 4.1: Filtrage par reconstruction

D'autre part notre algorithme travaille avec la ligne de partage des eaux du gradient. Par construction, toutes les régions qui seront présentes dans la segmentation correspondent à un bassin du gradient. Ce bassin sera inondé à partir d'un de ses minima. Par conséquent la présence d'une région dans la segmentation exige qu'elle contienne au moins un minimum du gradient. Or, si on regarde l'aspect du gradient morphologique d'un détail fin (voir fig. 4.2) on observe qu'il n'y a pas de minimum à l'emplacement du détail, ce qui l'empêche d'apparaître dans la segmentation. Par ce fait l'ouverture-fermeture par la boule élémentaire est un filtrage qui homogénéise l'image sans affecter le fonctionnement de notre algorithme de segmentation.

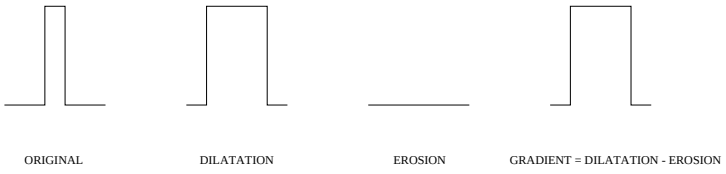


Figure 4.2: Gradient d'un détail fin.

Des filtres de taille supérieure à la boule élémentaire peuvent être appliqués pour simplifier encore plus l'image. Néanmoins, en augmentant la taille du filtre on risque de supprimer des objets fins mais longs et contrastés, par conséquent visibles. Voyons l'effet d'une ouverture-fermeture de taille 2 sur l'image 4.3(a). Nous pouvons observer que les bandes sombres du bâtiment, régions très contrastées et visibles ont été supprimées.

Les filtres aréolaires se révèlent plus adaptés aux filtrages plus forts, car les objets fins sont préservés s'ils sont suffisamment longs. Nous avons retenu les filtres aréolaires pour notre étape de simplification parce qu'ils nous fournissent des images plus simples avec une information plus pertinente. Le problème maintenant est de choisir la taille du filtre à utiliser. Pour ceci nous avons étudié l'évolution de l'image filtrée avec une taille de filtre croissante.

Puisque le filtre aréolaire est un filtre connexe, les simplifications de taille croissante élargissent les zones plates de l'image, et en conséquence le nombre de minima du gradient de l'image simplifiée diminue. Ainsi, la fonction $f(\lambda)$ qui exprime le nombre de minima du gradient de l'image simplifiée, en fonction de la taille du filtre aréolaire, est une fonction décroissante. La figure 4.1(a)

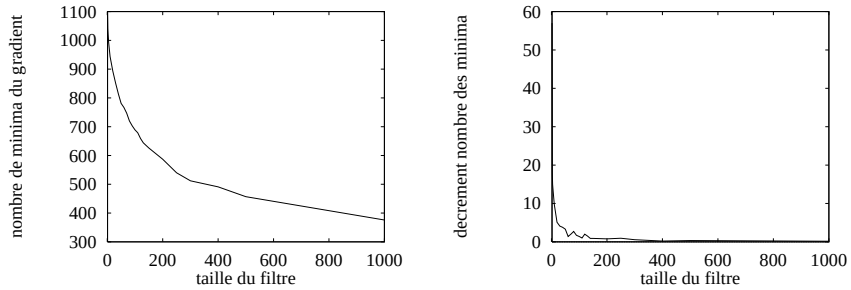
(a) Filtre par reconstruction de taille 2

(b) Filtre aréolaire de taille 60

Figure 4.3: Comparaison de filtres connexes.

illustre la fonction $f(\lambda)$ de l'image 4.1(a). La dérivée $f'(\lambda)$ de cette fonction exprime le nombre de minima qui disparaissent du fait de l'augmentation de 1 de la taille du filtre. La figure 4.4(b) présente la fonction dérivée $f'(\lambda)$. Nous observons une dérivée prononcée pour des tailles de filtre petites ce qui signifie que l'image contient un grand nombre de structures de petite taille. Il est très probable que ces structures correspondent au bruit ou à une texture fine que la segmentation ne peut pas représenter. Nous choisirons une taille de filtre pour laquelle $f'(\lambda)$ est devenue petite.

Pour l'exemple qui nous concerne, une soixantaine de minima disparaissent quand on passe d'un filtrage de taille 1 à un filtrage de taille 2. À partir d'une taille 60 cette pente est inférieure à 3 minima par taille du filtre. Ainsi un filtre aréolaire de taille 60 sera appliqué à cette image à l'étape de simplification (voir figure 4.3(b)), ce qui produit une image avec 9424 régions. On considère que le bruit a été éliminé et qu'on a obtenu les régions significatives, quoiqu'elles soient encore trop nombreuses pour un codage efficace. Le même algorithme est appliqué à d'autres images, dont les résultats sont présentés dans la section 4.4.



(a) Nombre de minima du gradient

(b) Nombre de minima qui disparaissent lors qu'on augmente de 1 la taille du filtre (dérivée du nombre de minima).

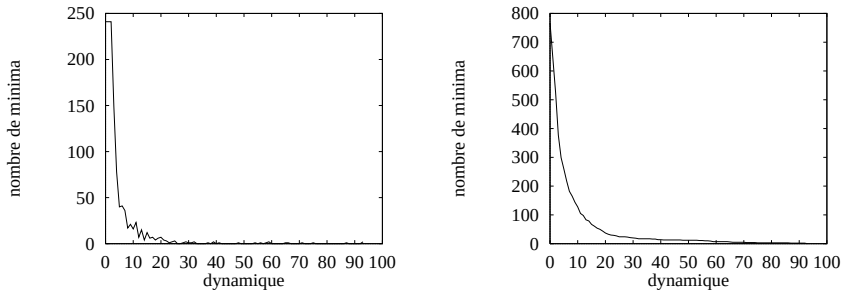
Figure 4.4: Filtrage aréolaire

4.2 Extraction de marqueurs

Nous avons une image à niveaux de gris dont le gradient a un nombre important de minima, ce qui produit une sursegmentation lorsqu'on calcule directement une ligne de partage des eaux. Ce qui nous intéresse c'est de classer ces minima pour choisir parmi eux les marqueurs des régions importantes, et arriver à une segmentation qui corresponde à celle de l'œil humain.

Les filtrages de taille croissante d'une image nous fournissent des partitions de plus en plus grossières. En augmentant la taille du filtre, les objets importants sont sélectionnés avec un critère exclusivement de taille. Néanmoins l'importance visuelle d'un objet fait intervenir d'autres critères, comme par exemple le contraste. C'est pourquoi, après un filtrage qui élimine des régions associées au bruit, nous allons quantifier l'importance visuelle des régions restantes. Ceci va nous permettre de sélectionner les plus visibles pour obtenir une segmentation avec un nombre de régions adapté au taux de compression visé. Cette opération est appelée *extraction de marqueurs*.

Commençons par voir quel est le contraste des régions après filtrage. Pour ce faire nous allons utiliser le concept de dynamique présenté dans la section 2.5. La figure 4.5(a) nous montre le nombre de minima en fonction de la dynamique pour l'image gradient correspondant à la figure 4.3(b). La plupart des minima ont une petite dynamique ce qui correspond à des régions de faible contraste et par conséquent pas très visibles pour l'œil humain. Si nous enlevons progressivement les minima qui ont les dynamiques les plus petites, c'est-à-dire les moins contrastés, nous obtenons à chaque fois une image plus simple, mais toujours en conservant les régions les plus contrastées. Sur la figure 4.5(b) nous voyons le nombre des régions présentes dans la segmentation pour les différentes valeurs de seuillage de la dynamique des minima.



(a) Histogramme de la dynamique

(b) Histogramme cumulé de la dynamique des minima

Figure 4.5: Nombre de régions en fonction de la dynamique.

La dynamique nous fournit une mesure du contraste relatif des minima. Considérons le contraste comme un critère de classement de l'importance des minima et sélectionnons les N régions les plus importantes, N étant un paramètre d'entrée de l'algorithme qui nous permet de générer une segmentation adaptée au taux de compression visé. Sur la figure 4.6(a) nous avons les 75 marqueurs les plus contrastés de l'image 4.3(b). Nous observons que plusieurs marqueurs ont été choisis sur une même bande sombre du bâtiment (voir figure 4.6(b)), ce qui donnera lieu à une représentation de la bande en plusieurs régions non

perceptibles. Par contre des régions bien visibles, comme certaines parties du visage n'ont pas été sélectionnées. Ce défaut est dû au fait que l'importance d'une région ne dépend pas seulement de son contraste avec les régions voisines mais aussi de sa taille. A contraste égal une grande structure sera plus visible qu'une petite structure, ou, ce qui revient au même, une structure moins contrastée peut être plus visible du fait qu'elle est plus grande.

Un classement des minima qui tienne compte de la dynamique et de la taille des régions serait plus adapté à la vision humaine. Mais lorsqu'on veut classer les minima par ordre d'importance on ne connaît pas encore la taille qu'aura la région que représente le minimum parce qu'on ne sait pas jusqu'où va se propager ce minimum lors de l'inondation du relief. La solution qu'on a retenue est de faire d'abord une sursegmentation, puis de donner un poids à chaque minimum en fonction de la surface de son bassin dans l'image sursegmentée. Ce poids est une fonction logarithmique de la surface. Une nouvelle estimation de l'importance d'un minimum est obtenue en multipliant la dynamique par le poids en fonction de la surface. Avec ce nouveau classement des minima un seuillage est appliqué de manière à retenir le nombre de minima souhaité. Les 75 minima sélectionnés avec ce nouveau critère sont montrés sur la figure 4.7. Les segmentations obtenues à partir de ces marqueurs sont présentées dans la section suivante.

(a) 75 minima de plus grande dynamique.

(b) Détail des marqueurs sur le bâtiment.

Figure 4.6: Sélection des marqueurs avec la dynamique.

Figure 4.7: Sélection des marqueurs avec la dynamique et la taille.

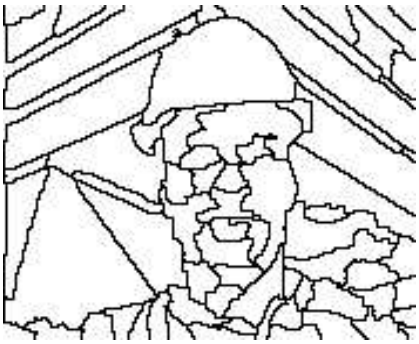
4.3 Croissance de marqueurs

Le but de cette étape est de placer les contours des objets jugés significatifs par l'étape précédente. Pour ceci nous allons utiliser la ligne de partage des eaux qui place le contour entre deux marqueurs sur la plus haute ligne de crête du gradient entre eux, ce qui correspond à la ligne de plus fort contraste entre eux. Cette étape est automatique. Il suffit de faire un changement d'homotopie du gradient pour qu'il n'ait que les minima choisis. Ce changement d'homotopie est réalisé par une reconstruction dont l'image marqueur est une image à zéro à l'emplacement des marqueurs et à la valeur maximale ailleurs. Le masque est l'image gradient.

Voyons sur la figure 4.8(a) la segmentation obtenue à partir des marqueurs de plus forte dynamique de la figure 4.6(a). Nous pouvons observer la coupure de la bande du bâtiment en plusieurs régions comme il avait été annoncé. La même étape de décision appliquée aux marqueurs de l'image 4.7 produisent l'image segmentée de la figure 4.8(c).

(a) Extraction des marqueurs: dynamique.

(b) Image 4.8(a) codée avec 5 coefficients de Fourier.



(c) Extraction des marqueurs: dynamique et taille.

(d) Image 4.8(c) codée avec 5 coefficients de Fourier.

Figure 4.8: Segmentations 2D, 75 régions

4.4 Résultats de segmentations 2D

Rappelons rapidement l'algorithme de segmentation retenu et appliquons-le à d'autres images.

L'étape de simplification est réalisée par un filtre aréolaire. Les filtres aréolaires qui appartiennent à la famille des filtres connexes, fusionnent les zones plates, ce qui fait diminuer le nombre de minima de l'image gradient. Plus la taille du filtre est grande, moins le gradient a de minima. La taille du filtre que nous avons choisie est telle que si nous l'augmentons encore d'un pixel, le résultat du filtrage élimine un nombre petit de minima du gradient. Dans la pratique ce nombre petit de minima est de trois, paramètre heuristique qui donne de bons résultats pour des images test variées.

L'extraction des marqueurs est réalisée en deux étapes: une première sélection à partir de la dynamique nous donne une sursegmentation qui nous permet d'estimer la taille des régions. Une nouvelle estimation de l'importance de chaque minimum est obtenue en multipliant sa dynamique par un poids dépendant de sa surface. Avec ce nouveau classement des minima, un seuillage est appliqué de manière à retenir le nombre de minima souhaité.

L'homotopie du gradient est modifiée pour ne garder que les minima choisis; une ligne de partage des eaux nous fournit finalement les contours cherchés.

Voyons quelques résultats: Pour avoir une idée de la qualité de la segmentation nous allons habiller les régions d'une texture, obtenant ainsi une image codée. De nombreuses techniques peuvent être utilisées à ce propos; du simple niveau de gris moyen, passant par les approximations polynomiales, jusqu'aux décompositions fréquentielles ainsi que des méthodes stochastiques. Notre objectif n'étant pas de développer de nouvelles méthodes de codage de texture, nous utiliserons celles qui existent déjà. Dans ce chapitre nous allons utiliser la décomposition fréquentielle de Fourier. Le contenu des régions est ramené au domaine fréquentiel et dans ce domaine les N fréquences de plus grande amplitude sont choisies. A partir de ces N fréquence nous obtenons, par transformée inverse de Fourier une approximation du signal à l'intérieur de la région.

Le coût de codage d'un coefficient de Fourier inclut le codage de la fréquence, l'amplitude et la phase de la raie fréquentielle choisie. Les coefficients sont quantifiés et un codeur entropique élimine la redondance qui pourrait exister entre eux. Le coût final d'un coefficient est alors d'environ 20 bits.

Le codage de la segmentation 4.8 (a) à partir des 5 coefficients de plus grande amplitude dans le domaine fréquentiel apparaît sur la figure 4.8 (b). La relation signal sur bruit de cette image est de 25,35 dB. Le même codage de texture appliqué à la segmentation 4.8 (c) donne la figure 4.8(d) dont la qualité est de 26,17 dB.

Le taux de compression est le rapport entre le nombre de bits de l'image originale et le coût du codage. Le nombre de bits de l'image originale est de $176 \times 144 \times 8 = 202752$ *bits* (taille de l'image en pixels par nombre de bits par pixel). Le coût du codage comprend le coût des contours plus le coût de la texture. Les contours sont codés avec un algorithme de chain code [12, 11, 27] qui utilise 1.3 bits pour coder un point de contour et la texture avec 5 coefficients de Fourier par région.

La segmentation 4.8(a) comporte 3175 points de contour (P.C.) et 75 régions. Son coût est alors de:

$$cot = 3175\ P.C. \times 1.3 \frac{bits}{P.C.} + 75 régions \times 5 \frac{coefficient}{région} \times 20 \frac{bits}{coefficient} = 11627 bits.$$

Ce qui fait un taux de compression de

$$compression = \frac{202752\ bits}{11627\ bits} = 17.43$$

Le même calcul pour l'image 4.8(d) donne:

$$cot = 3710 P.C. \times 1.3 \frac{bits}{P.C.} + 75 régions \times 5 \frac{coefficient}{région} \times 20 \frac{bits}{coefficient} = 12323 bits$$

$$compression = \frac{202752\ bits}{12323\ bits} = 16.45$$

Sur la figure 4.9 nous avons le résultat de l'application du même algorithme sélectionnant les 100 régions les plus visibles. Le taux de compression obtenu est de 15.5 et la qualité de 26.6 dB.

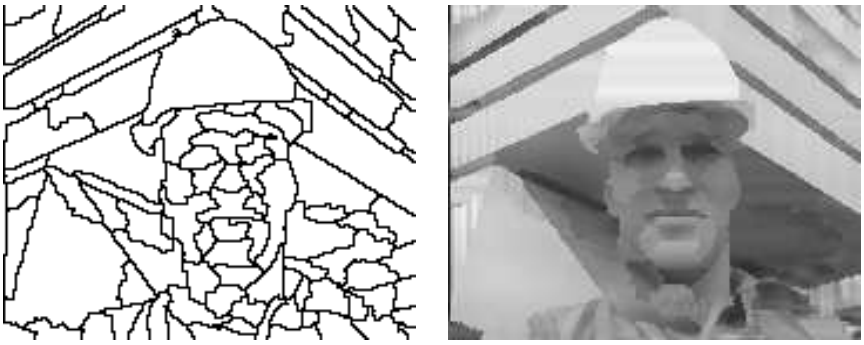
La figure 4.10 montre la segmentation de l'image 4.10(a) avec deux niveaux de résolution (fig.4.10(b) et fig. 4.10(d)). Le codage de la segmentation 4.10(b) est présenté dans la figure 4.10(c). Le taux de compression obtenu est de 16.87 et la qualité de 26.05 dB. Le codage de la segmentation 4.10(d) est sur la figure 4.10(e) dont la qualité est de 27.24 dB et la compression de 13.1.

La même procédure a été appliquée à l'image de la figure 4.11(a), nous permettant d'obtenir les segmentations 4.11(b) et 4.11(d) dont les qualités respectives sont de 32.52 dB et 33.59 dB et les taux de compression de 34.3 et de 24.5.

Finalement nous avons obtenu deux segmentations de la figure 4.12(a) avec 10 (fig. 4.12(b)) et 16 régions (fig. 4.12(d)). Les qualités respectives sont de 21.28 dB et 21.66 dB et les taux de compressions de 180 et de 132.

La qualité de ces images codées peut être améliorée sans augmenter le coût de codage, au prix d'une charge de calcul un peu plus grande. D'abord, le nombre de coefficients utilisés pour coder une région peut être adapté à la taille de la région et à la complexité de son contenu. Par ailleurs, nous travaillons avec des régions de forme quelconque, tandis que la transformée de Fourier le fait sur des fenêtres rectangulaires. Le spectre d'une région de forme quelconque comprend l'information propre au motif intérieur, mais aussi celle due au contour. L'imbrication de ces deux informations introduit une dispersion spectrale qui interdit toute sélection efficace des fréquences. L'extrapolation du contenu de la région dans une fenêtre rectangulaire déconvolue l'information de la forme de celle de son contenu et permet de mieux choisir les composantes fréquentielles du signal. Nous avons utilisé une technique d'extrapolation très simple , qui consiste à répéter spatialement le signal de la région jusqu'à remplir la fenêtre rectangulaire. D'autres techniques d'extrapolation plus sophistiquées [50] peuvent être utilisées, comme celle présentée dans [10]. Ceci nous permet de mieux choisir les raies spectrales d'une région et par conséquent d'obtenir une image codée de meilleure qualité. Illustrons la différence de qualité des deux méthodes de codage de texture. Sur la figure 4.13 nous avons les images codées à partir de la segmentation de la figure 4.10(b). L'image 4.13(a) utilise une extrapolation

simple (par répétition) tandis que l'image 4.13(b) utilise l'extrapolation de [50]. Avec le même coût de codage nous sommes passés d'une qualité de 26.05 dB à 26.76 dB.



(a) 100 régions

(b) Image 4.9(a) codée avec 5 coefficients de Fourier.

Figure 4.9: Segmentations 2D

4.5 Problème de résolution

La ligne de partage des eaux appliquée à la segmentation consiste à inonder le relief du gradient à partir de ses minima. La zone que chaque minimum inonde constitue sa zone d'influence, ce qui représente une région en termes de segmentation.

Par construction toute région qui ne contient pas un minimum du gradient, comme par exemple un détail fin, disparaît de la segmentation. C'est pourquoi l'image segmentée est dépourvue de détails fins ce qui produit une importante perte de qualité visuelle.

Nous avons essayé de résoudre ce problème de deux façons, illustrées dans les paragraphes suivants.

4.5.1 sur-échantillonnage de l'image

La première solution consiste à élargir l'image dans le sens spatial, de telle sorte qu'un pixel de l'image de départ soit représenté par quatre pixels de l'image d'arrivée. De cette façon, on génère l'espace nécessaire pour placer un contour entre deux régions fines. Ainsi, les détails fins sont vus à une échelle accessible pour le gradient, ce qui résout le problème de résolution. En échange, l'image sera quatre fois plus grande, par conséquent le temps de calcul et les besoins de mémoire seront aussi multipliés par quatre. Le lecteur peut trouver une description plus complète dans [43, 21].

4.5.2 deux niveaux de résolution

La deuxième solution consiste à travailler à deux niveaux de résolution. Les détails fins d'une image sont filtrés donnant ainsi lieu à deux images:

- une image dépourvue de détails fins, qui peut être traitée avec le gradient.

(a) Image originale

(b) 75 régions

(c) Image 4.10(b) codée avec 5 coefficients de Fourier.

(d) 100 régions

(e) Image 4.10(c) codée avec 5 coefficients de Fourier.

Figure 4.10: Segmentations 2D

(a) Image originale

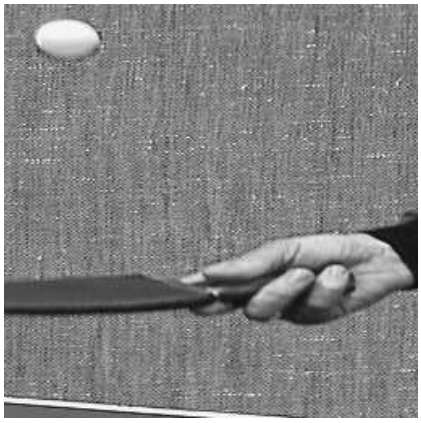
(b) 35 régions

(c) Image 4.11(b) codée avec 5 coefficients de Fourier.

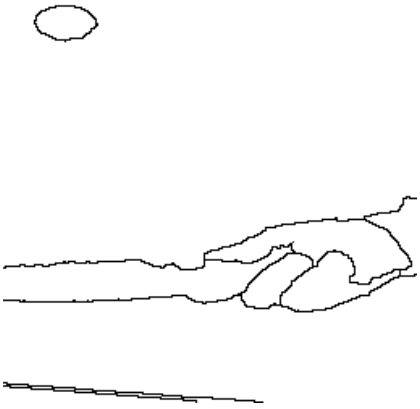
(d) 50 régions

(e) Image 4.11(c) codée avec 5 coefficients de Fourier.

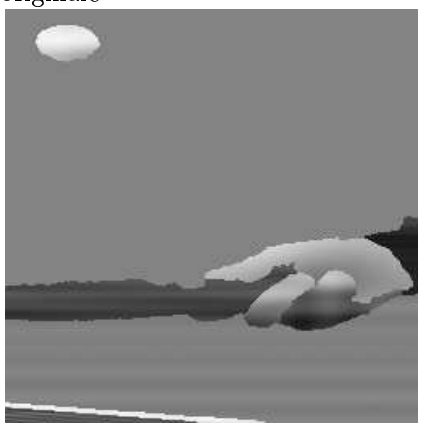
Figure 4.11: Segmentations 2D



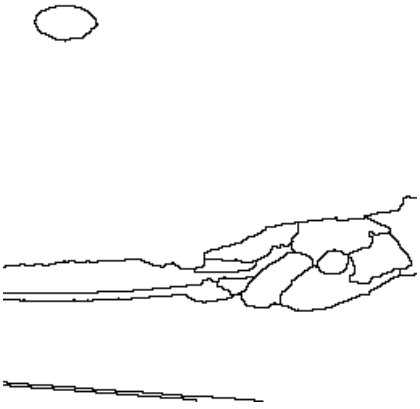
(a) Image originale



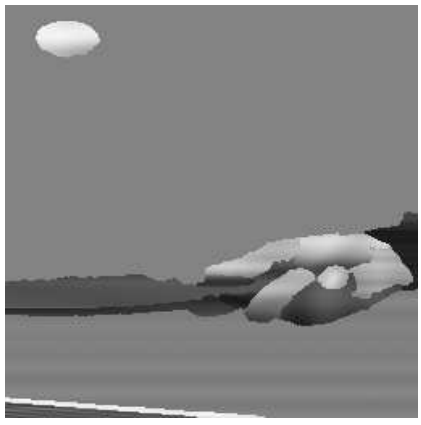
(b) 10 régions



(c) Image 4.12(a) codée avec 5 coefficients de Fourier.



(d) 16 régions



(e) Image 4.12(c) codée avec 5 coefficients de Fourier.

Figure 4.12: Segmentations 2D

(a) Image codée utilisant une extrapolation simple (répétition)

(b) Image codée utilisant une extrapolation plus sophistiquée

Figure 4.13: Segmentations 2D

- une image résidu obtenue par différence entre l'image originale et l'image filtrée. Cette image comporte les détails fins et ne peut pas être segmentée à l'aide d'une image gradient. Elle est traitée en utilisant les filtres perceptifs [33].

Voyons le filtre utilisé pour décomposer l'image. Il s'agit du filtre gommette présenté dans [34]. La plupart des filtres morphologiques sont basés sur les ouvertures et fermetures. Ils enlèvent les structures de taille inférieure à l'élément structurant, mais en même temps ils modifient les contours des grandes structures. Voyons sur la figure 4.14(b) le résultat de l'application d'un filtre alterné séquentiel de taille 2 à l'image de la figure 4.14(a). Nous observons par exemple sous le visage, le changement de forme d'un objet qui résiste au filtrage. Pour éviter la déformation des objets qui résistent au filtrage nous pouvons utiliser les ouvertures et fermetures par reconstruction. Voyons sur la figure 4.14(c) l'effet du filtrage par reconstruction. Les contours des grandes structures n'ont pas été modifiés mais nous observons que certains détails fins (par exemple, les détails sur la caméra) ont été reconstruits comme une extension du visage qui résiste au filtrage.

(a) Image Originale

(b) Filtre alterné séquentiel de taille 2

(c) Filtre alterné séquentiel par reconstruction de taille 2

Figure 4.14: Filtre alterné séquentiel et filtre alterné séquentiel par reconstruction

Voyons sur la figure 4.15 le principe du filtre gommette, dont l'objectif est d'éliminer les détails fins sans modifier les contours des grandes structures. Le filtre gommette utilise le top-hat et le top-hat par reconstruction pour distinguer ce qui est une extension d'un objet (à ne pas filtrer), de ce qui est un détail superposé.

La procédure du filtre gommette est décrite ici:

- Construction du Top-Hat (résidu d'une ouverture ou fermeture): TH_n
- Construction du Top-Hat par reconstruction (résidu d'une ouverture ou fermeture par reconstruction): TH_n^{rec}.
- Un seuillage du Top-Hat par reconstruction marque les détails à supprimer.
- La reconstruction du Top-Hat à partir des marqueurs nous donne l'ensemble des détails complets à supprimer.
- Le résultat des opérations précédentes est soustrait à l'image originale.

La formule du filtre est donc la suivante:

$$gommette = X - Rec\{TH_n \times Seuil(TH_n^{rec}), TH_n\}$$

Voyons un exemple de comportement de ce filtre. Soit la figure 4.15(a) l'image de départ. Elle contient un objet de grande taille avec un objet petit superposé. L'ouverture de cette image (fig. 4.15(f)) élimine l'objet superposé, mais modifie aussi la forme du grand objet. La figure 4.15(g) donne le top-hat. Si nous observons l'ouverture par reconstruction (fig. 4.15(c)), nous réalisons que le détail superposé a été reconstruit avec le niveau de gris du grand objet qui résiste au filtrage. De ce fait, le top-hat par reconstruction 4.15(d) a un niveau de gris qui correspond à la différence de niveau de gris entre le détail et le grand objet. Quand ce niveau de gris est suffisamment fort, on conclut que la structure n'appartient pas à l'objet. Alors, un seuillage du top-hat par reconstruction 4.15(e) marque les détails qui doivent être filtrés. Pour récupérer la globalité du détail qui doit être enlevé, les marqueurs obtenus sont reconstruits à l'intérieur du top-hat (voir fig 4.15(h)). Le résultat du filtre gommette apparaît sur la figure 4.15(b).

Illustrons cette procédure avec un exemple. Prenons l'image de la figure 4.16(a) et décomposons-la en deux images: une image dans laquelle les détails ont été gommés 4.16(b) et une image résidu 4.16(d). Les deux images suivent des traitements différents: l'image filtrée est segmentée avec la procédure présentée précédemment. Le résultat obtenu est sur la figure 4.16(c). Le résidu est filtré par un filtre perceptif [34]. Les détails visibles sont ceux qui ont un fort contraste et ceux qui ayant un contraste plus faible, ont une taille suffisamment grande. La figure 4.16(e) montre les détails qui ont été retenus comme importants. Ces détails sont restitués sur l'image des grandes structures (fig. 4.16(c)), obtenant ainsi la figure 4.16(f). Le problème de codage des détails de manière efficace est abordé dans [4, 5].

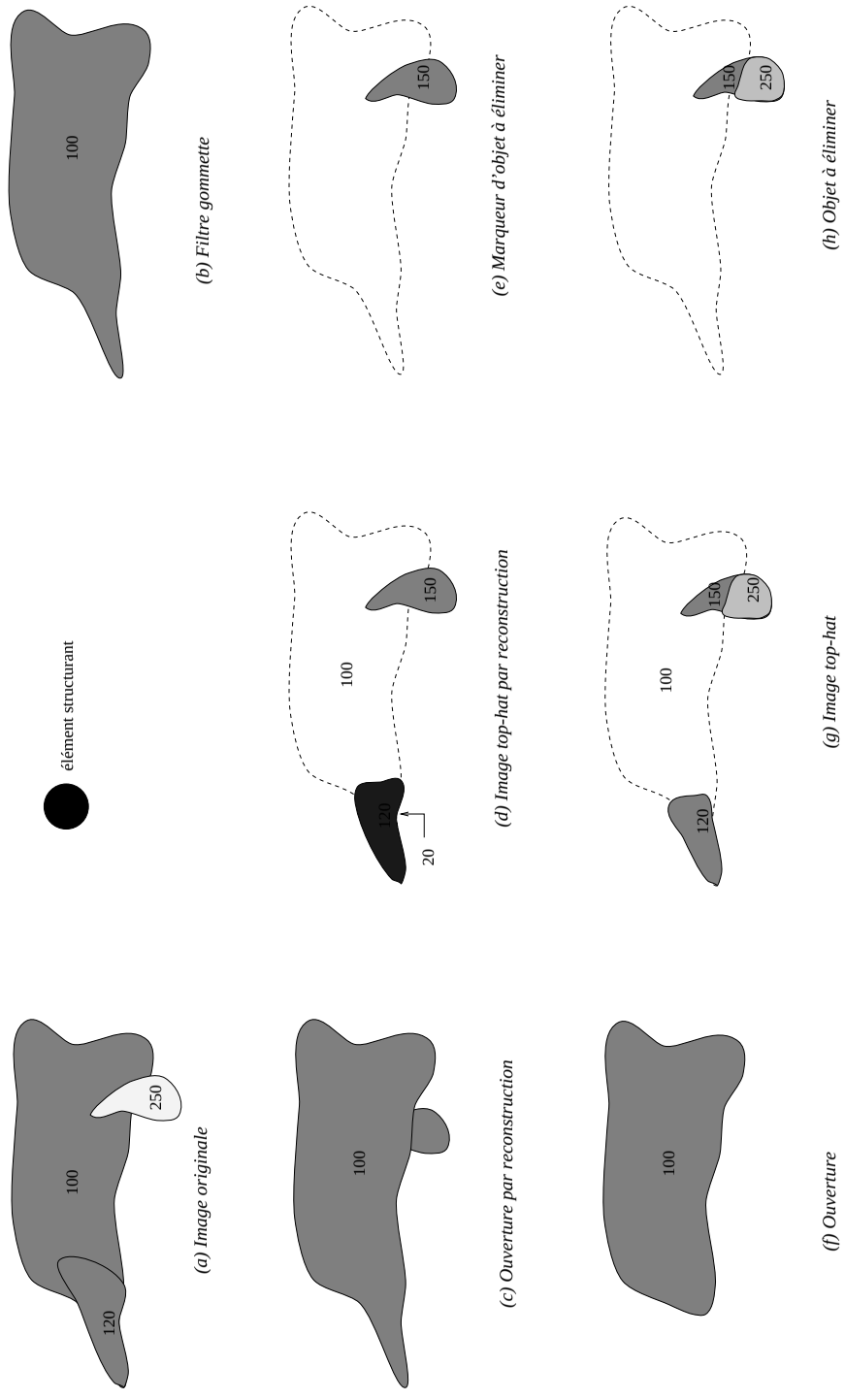


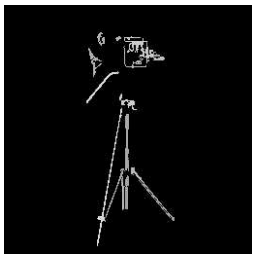
Figure 4.15: Principe du filtre gommette.

(a) Image Originale

(b) Filtre gommette

(c) Segmentation des grandes structures

(d) Composante des détails



(e) Détails filtrés

(f) Restitution des détails

Figure 4.16: Décomposition d'image par filtre gommette

4.6 Problème lié à la fragilité de la dynamique

L'inondation du gradient à partir de tous ses minima produit une forte sur-segmentation. Le problème est ainsi abordé en deux étapes: une première qui sélectionne les régions importantes et une deuxième qui place les contours des régions retenues. Pour sélectionner les régions, la dynamique nous aide à classer les minima du gradient selon leur contraste. Nous avons utilisé cette mesure comme critère de visibilité d'un minimum: plus la dynamique est forte, plus le minimum est important.

Néanmoins nous avons pu constater que la dynamique est une mesure fragile: une fissure dans une barrière de fort contraste fait obtenir une estimation faible du contraste. Pour pallier cette fragilité nous proposons la solution suivante.

Nous calculons la dynamique du gradient et nous choisissons un nombre de minima supérieur au nombre de régions visé. Une sur-segmentation est ainsi obtenue. Ensuite nous étudions le gradient sur les frontières de la sur-segmentation que nous venons de générer et nous associons à chacune d'elles une évaluation plus pertinente que la valeur minimale, par exemple la valeur moyenne. Le gradient moyen le long de la frontière donne une mesure plus fiable du contraste entre deux régions, que la dynamique calculée sur le gradient initial. Ainsi, ce nouveau "gradient", est utilisé pour recalculer la dynamique. Parmi les minima sélectionnés pour obtenir la sur-segmentation, nous choisissons les plus contrastés selon la nouvelle dynamique qu'on a calculée. Une segmentation avec un nombre inférieur de régions est obtenue. Cette procédure peut être répétée avec un nombre décroissant de régions et constitue un algorithme de segmentation bottom-up.

Voyons le résultat que nous obtenons avec cette nouvelle dynamique. Prenons l'image de la figure 4.17(a) comme image originale et segmentons-la avec la dy-

namique initiale, comme nous l'avons fait dans la section 4.4. L'image segmentée obtenue est sur la figure 4.17(b) et l'image codée correspondante (avec 5 coefficients de Fourier par région) est sur la figure 4.17(c). Maintenant considérons une sur-segmentation et réévaluons la dynamique selon le contraste tout au long de ses frontières. La segmentation avec le même nombre de régions obtenue à partir de la dynamique réévaluée est illustrée par la figure 4.17(d). Nous voyons que la dynamique initiale avait fusionné le visage avec la voiture à cause de la ressemblance des niveaux de gris autour de l'oreille. La dynamique réévaluée, ferme les possibles fissures dans les barrières de fort contraste et donne une meilleure estimation de l'importance du visage. Le visage est ainsi segmenté. En échange on a perdu des régions sur la veste, qui correspondent à des ombres, beaucoup moins visibles.

La dynamique réévaluée donne ainsi une meilleure estimation de l'importance visuelle des régions.

4.7 Conclusions

L'algorithme classique de segmentation de la morphologie mathématique sépare le problème en deux étapes: une première qui consiste à sélectionner les régions importantes et une deuxième qui extrait les contours de ces régions. La première étape constitue la principale difficulté de la segmentation. La deuxième consiste à calculer la ligne de partage des eaux, ce qui se fait de façon automatique.

Nous avons utilisé cet algorithme pour développer une application de codage. Le critère de sélection des régions est pour nous l'importance visuelle, que nous avons estimée à partir de la dynamique. Celle-ci est une mesure exclusivement de contraste, mais l'importance visuelle d'une région dépend aussi de sa taille. Or, au moment de sélectionner les marqueurs nous ne connaissons pas la taille des régions qu'ils représentent. Donc, pour estimer la taille des régions, nous réalisons plusieurs étapes de segmentation.

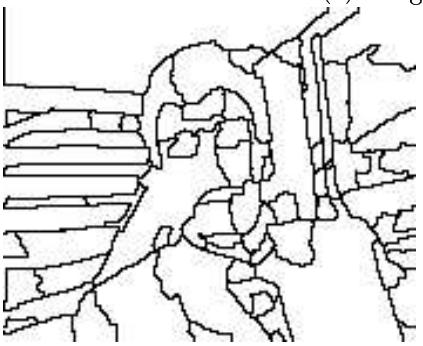
C. Vachier [53] propose une autre solution à ce problème. Elle combine le contraste d'un minimum avec la taille de la région qu'il représente, à travers une mesure de volume. Elle définit ainsi, ce qu'elle appelle la "valeur d'extinction volumique". Cette mesure permet d'utiliser l'information de taille d'un marqueur sans passer par plusieurs étapes de segmentation. Une description complète de cette notion, ainsi que des exemples d'application peuvent être trouvés dans [51].

Par ailleurs la dynamique est une mesure fragile, puisqu'elle évalue une frontière à partir d'une valeur minimale. La réévaluation de la dynamique sur plusieurs étapes de sur-segmentation donne une estimation du contraste plus fiable.

D'autre part, le gradient n'est pas suffisamment précis pour représenter les détails fins. Cela pose un problème de résolution: la segmentation est dépourvue de détails. Nous proposons deux solutions: soit le sur-échantillonnage, soit la décomposition de l'image en deux composantes: une qui peut être traitée avec le gradient, et une autre qui contient les détails qui sont en dessous de la résolution du gradient.

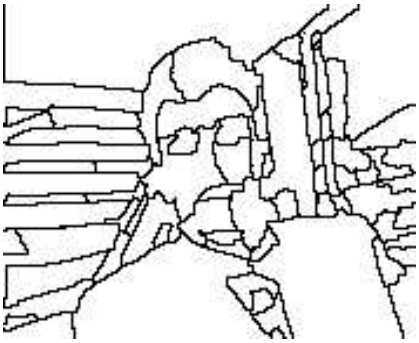
Les solutions que nous avons apportées sont néanmoins des solutions partielles. Pour éviter la fragilité de la segmentation nous avons proposé de calculer la moyenne du gradient le long des frontières, en plusieurs étapes de sur-segmentation. Vient alors la question: combien d'étapes sont nécessaires et

(a) Image originale



(b) Dynamique sans réévaluation

(c) Image (b) codée avec 5 coefficients de Fourier.



(d) Dynamique avec réévaluation

(e) Image (d) codée avec 5 coefficients de Fourier.

Figure 4.17: Réévaluation de la dynamique

combien de régions doit-on garder à chaque étape? En ce qui concerne la décomposition de l'image en deux composantes (grandes structures et détails), la question qui reste à résoudre est: quel est la proportion de détails que nous allons inclure par rapport au nombre de grandes régions?

Ce sont des questions qui auront une réponse plus générale au chapitre 6.

Chapter 5

Segmentation tridimensionnelle basée sur la dynamique

5.1 Introduction

Ce chapitre est consacré à l'extension aux séquences des idées exposées pour une segmentation bidimensionnelle.

Nous avons vu dans le chapitre 3 plusieurs approches pour traiter la dimension temporelle d'une séquence:

- une approche bidimensionnelle appliquée à chaque instant de la séquence, suivie d'une mise en correspondance des régions a posteriori ;
- une approche tridimensionnelle ($x \times y \times t$) ;
- une approche récursive.

Nous en avons conclu que l'approche récursive est la plus adaptée à notre problème, parce qu'elle impose une stabilité temporelle, n'introduit pas de retard et ses besoins de mémoire sont réduits. Voyons maintenant l'implémentation précise de notre application, expliquant les problèmes rencontrés ainsi que les solutions apportées.

5.2 Transmission de l'information du passé

Une séquence d'images décrit l'évolution d'une scène dans le temps. En général les changements entre deux trames successives sont mineurs par rapport à l'information contenue dans une seule trame. Autrement dit, la redondance temporelle est grande. Dans le cadre d'un système de codage orienté objet, la segmentation qui nous mène à un haut taux de compression est celle qui garde une cohérence temporelle, celle qui suit les objets présents dans la scène tout au long du temps. La raison en est qu'une segmentation cohérente dans le temps peut être prédite par une compensation de mouvement, tandis qu'un nouvel

objet dans la scène doit être codé entièrement, sans exploiter la redondance temporelle.

L'application d'un algorithme bidimensionnel à deux instants différents de la séquence, malgré la ressemblance entre les deux trames, produit des segmentations différentes pour lesquelles une mise en correspondance des régions a posteriori n'est pas envisageable. C'est pourquoi la transmission de l'information du passé au présent est nécessaire pour obtenir une segmentation cohérente dans le temps. Nous avons vu dans le chapitre 3 que l'utilisation de la segmentation précédente comme marqueurs du temps courant (dans une image tridimensionnelle) est un bon moyen de transmettre l'information du passé au présent. Appliquons maintenant un algorithme récursif à la segmentation d'une séquence.

Commençons par un cas simple. Imaginons deux instants d'une séquence, entre lesquels les objets présents ont bougé mais aucun nouvel objet n'est apparu dans la scène. Dans ce cas, l'étape d'extraction de marqueurs n'est plus nécessaire, la trame précédente nous transmet l'information des objets existants et nous devons simplement suivre l'évolution de ces objets. Pour ce faire nous générons une image 3D avec le temps précédent et le temps courant en les plaçant côte à côte comme l'illustre la figure 5.1

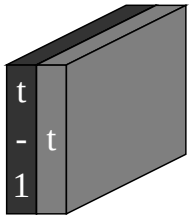


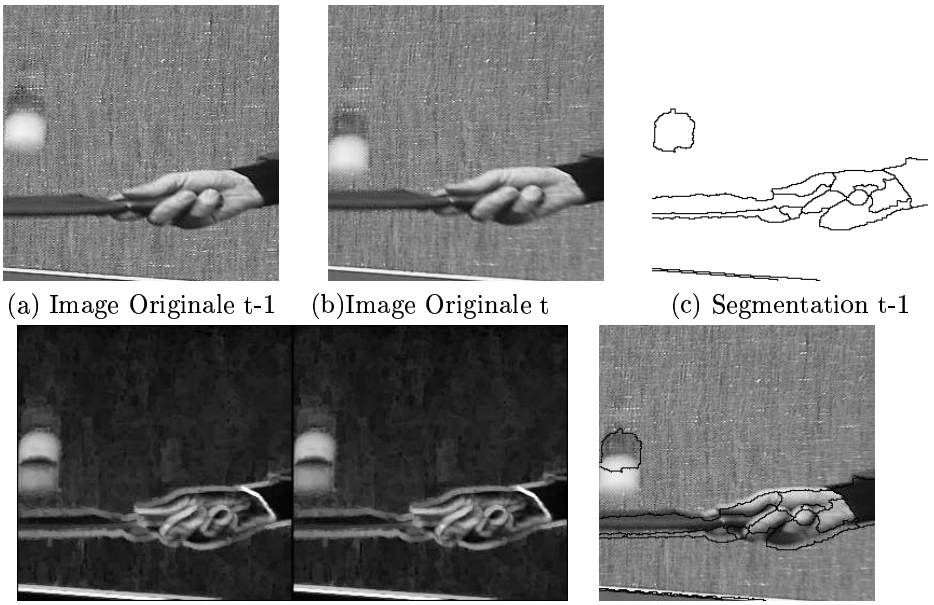
Figure 5.1: Fenêtre glissante

La procédure que nous avons suivie est la suivante:

Nous commençons par simplifier l'image avec un filtre aréolaire, mais cette fois-ci tridimensionnel. Pour que le filtrage soit équivalent à celui appliqué en deux dimensions nous multiplions la taille du filtre par le nombre des plans de l'image 3D, c'est-à-dire deux. Ensuite nous calculons le gradient morphologique tridimensionnel de cette image. Les fortes valeurs du gradient correspondent soit à des zones contrastées bidimensionnelles soit à des zones de mouvement. C'est de cette image gradient que nous allons extraire l'information de contour, moyennant un algorithme de ligne de partage des eaux qui devient une surface de partage des eaux sur une image 3D. L'eau provenant des marqueurs, dans notre cas des objets du temps t-1 (qui se trouvent dans le premier plan de la fenêtre), inonde le relief du gradient jusqu'à ce que la totalité de l'image soit couverte. A ce moment là, tout pixel du temps courant est atteint par l'eau provenant d'un des marqueurs et en conséquence on sait à quelle région il appartient. L'image courante est segmentée et les régions sont cohérentes avec celles du passé. Illustrons cette procédure avec un exemple.

Les deux trames étudiées (t-1 et t) sont sur la figure 5.2(a) et 5.2(b) et la segmentation t-1 sur 5.2(c). Nous voulons obtenir les contours des régions de t-1 au temps courant. Nous pouvons observer sur la fig. 5.2(d) que le gradient

morphologique 3D du groupement des temps précédent et courant possède des fortes valeurs sur toute la zone de mouvement: la balle, la raquette, la main... . C'est sur ces bandes épaisses que l'algorithme de croissance de marqueurs doit placer les frontières; elles constituent des zones d'incertitude. Voyons sur la figure 5.2(e) le résultat de l'application de l'algorithme de ligne de partage des eaux sur cette image gradient. Nous observons que les contours obtenus ne correspondent pas à l'emplacement des objets, à cause de l'incertitude que leur mouvement rapide provoque.



(a) Image Originale t-1 (b)Image Originale t (c) Segmentation t-1

(d) Gradient tridimensionnel. (e) Segmentation t

Figure 5.2: Segmentation sur un gradient tridimensionnel.

En effet, un objet de mouvement rapide est un objet fin dans la dimension temporelle et en conséquence le gradient n'est pas suffisamment précis pour représenter son contour (voir fig. 5.3). C'est pourquoi nous sur-échantillonnons le gradient dans l'axe temporel. La figure 5.4 nous montre la procédure utilisée pour calculer le gradient sur-échantillonné. Nous introduisons une trame supplémentaire entre les trames $t-1$ et t, sur laquelle se placeront les contours temporels. Nous assignons à la trame introduite la valeur minimale (resp. maximale) pour réaliser la dilatation (resp. érosion) de façon à ce que sa valeur n'interfère pas avec la procédure. La différence entre la dilatation et l'érosion, c'est-à-dire le gradient, contient le contour temporel sur la trame rajoutée, tandis que l'incertitude du contour spatial sur les trames $t-1$ et t est réduit à un seul pixel. Pour plus de détail sur cette procédure le lecteur peut consulter [43].

Reprenons l'exemple de la figure 5.2, qui nous avait posé des problèmes. Nous calculons le gradient tridimensionnel suréchantillonné dans l'axe temporel (voir fig. 5.5). L'étape de décision appliquée sur ce nouveau gradient donne le résultat de la figure 5.5. Nous pouvons voir que les contours sont maintenant adaptés aux objets de la scène.

Cette méthode donne des bons résultats tant que les conditions suivantes

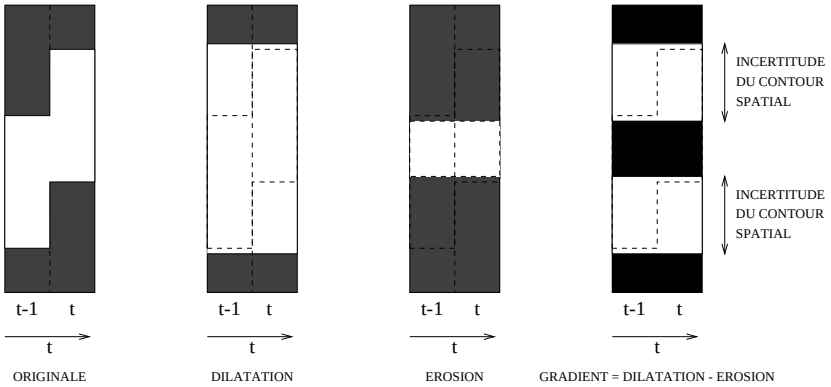


Figure 5.3: Gradient tridimensionnel d'un objet en mouvement rapide.

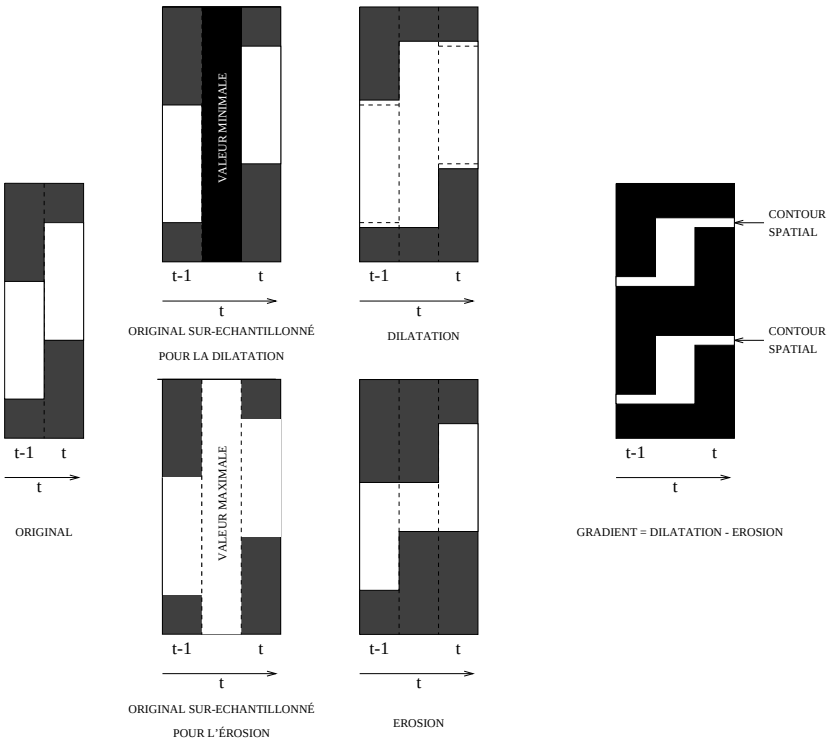
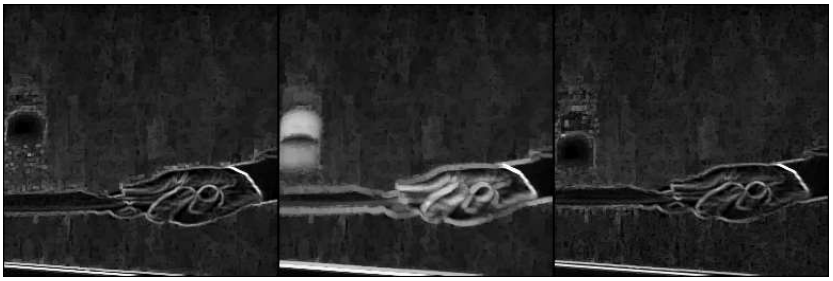
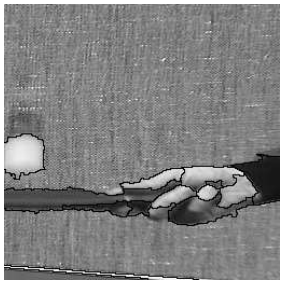


Figure 5.4: Gradient tridimensionnel sur-échantillonné.



(a) Gradient tridimensionnel sur-échantillonné.



(b) Segmentation t

Figure 5.5: Gradient tridimensionnel sur-échantillonné.

sont remplies:

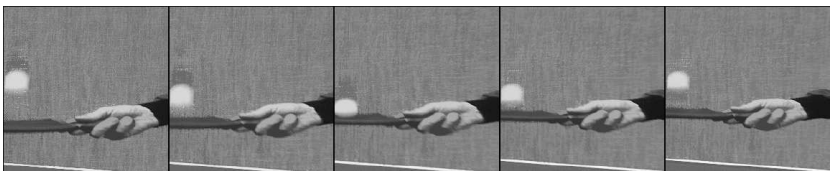
- Tous les objets de la séquence sont présents dans la scène dès la première image.
- L'emplacement d'un objet à deux instants consécutifs a toujours une intersection spatiale qui nous permet de passer le label (étiquette qui identifie une région) d'un temps à l'autre et de le reconnaître comme étant le même objet.

En général ces deux conditions ne sont pas remplies par les séquences réelles. C'est pourquoi nous devons compléter l'algorithme de façon à tenir compte des variations dans la scène.

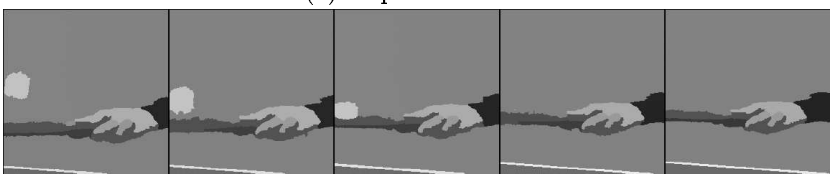
5.3 Inclusion de nouvelles régions

L'algorithme présenté dans la section précédente donne des bons résultats quand tous les objets de la séquence sont présents dès la première image et quand leur mouvement ne provoque pas de déconnexions temporelles (un objet sans intersection spatiale entre deux temps consécutifs est traité comme deux objets différents). Dans ce cas, après une segmentation *intra*, l'extraction de marqueurs n'est plus nécessaire: la segmentation précédente donne un bon jeu de marqueurs pour segmenter l'image courante. Cette procédure est capable de suivre des objets présents sur plusieurs images, de faire disparaître un objet de la scène (si son marqueur n'atteint pas le temps courant dans le processus d'inondation de la ligne de partage des eaux), mais il échoue de façon intrinsèque quand un

nouvel objet apparaît (les seuls marqueurs dont on dispose sont ceux du passé). Sur la figure 5.6 nous avons un exemple de cette situation. La déconnexion temporelle de la balle de ping-pong, à cause de son mouvement rapide, fait disparaître l'objet de la segmentation.



(a) Séquence filtrée



(b) Séquence segmentée

Figure 5.6: Segmentation sans nouvelles régions

L'algorithme doit être complété pour résoudre les cas d'objets déconnectés ou, de façon plus générale, les cas de nouveaux objets. Nous avons réalisé l'inclusion de nouvelles régions tout en gardant une stabilité de la façon suivante:

1. **Extraction indépendante de marqueurs:** les objets visuellement importants de la trame t sont sélectionnés avec un critère de contraste-taille comme présenté dans la section 4.2. Cette sélection est réalisée sans prendre en compte la segmentation précédente et son résultat est pris comme un jeu provisoire de marqueurs. Ces marqueurs provisoires peuvent être de trois types:

   (a) marqueurs déjà existants dans le passé et qui ne doivent pas être retenus comme de nouveaux marqueurs

   (b) marqueurs signalant les nouveaux objets apparus dans la scène. Ce sont les marqueurs que nous cherchons pour les ajouter aux marqueurs du passé.

   (c) marqueurs qui proviennent des fluctuations dans le classement de l'importance visuelle des régions, que nous voulons éliminer pour garder une bonne stabilité temporelle.

   La figure 5.7 montre les marqueurs sélectionnés pour le temps ou la balle de ping-pong disparaît.

2. **Validation des nouveaux marqueurs:** des trois types de marqueurs provisoires, seulement ceux qui correspondent à des changements importants dans la scène nous intéressent. Ce critère est introduit moyennant un masque de changement qui élimine tout marqueur associé à des régions qui n'ont pas changé dans le temps. Le masque est généré de la façon suivante: la différence entre deux plans successifs simplifiés est calculée (fig. 5.8(a)). Les aires de changement apparaissent avec un haut niveau

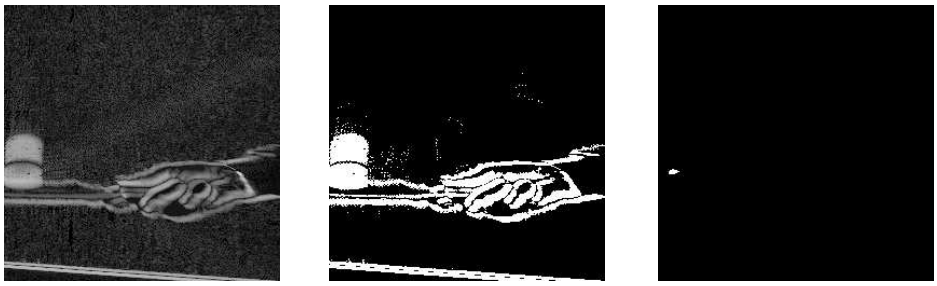
Figure 5.7: Marqueurs indépendants.

de gris. Un seuil de contraste est nécessaire pour obtenir le masque binaire qui signale les zones de changement. Pour cela, l'étape d'extraction de marqueurs nous fournit un seuil de dynamique qui correspond au contraste requis par une région pour être considérée comme visuellement importante. Nous appliquons ce seuil à l'image différence pour déterminer le masque de changement (5.8(b)). Les marqueurs que nous cherchons sont ceux qui sont sous le masque (fig. 5.8(c)) pour l'une des deux raisons suivantes:

(a) ce sont des marqueurs d'objets importants car ils ont été sélectionnés dans l'étape 1;

(b) ce sont de nouveaux objets parce qu'ils sont dans une zone de fort changement.

3. **Marqueurs finaux** Le jeu de marqueurs final est l'union des marqueurs précédents et nouveaux: sur le premier plan de la fenêtre glissante nous avons la segmentation précédente (comme marqueurs des objets permanents) et sur le plan qui correspond au temps courant nous avons les marqueurs provisoires sous le masque de changement (comme marqueurs de nouvelles régions).



(a) Image différence *(b) Masque de changement* *(c) Marqueurs nouveaux*

Figure 5.8: Masque de changement.

Cette technique nous permet d'obtenir une segmentation stable dans le temps et avec de nouvelles régions quand un changement important dans la scène le justifie. La figure 5.9 contient un diagramme de cette technique.

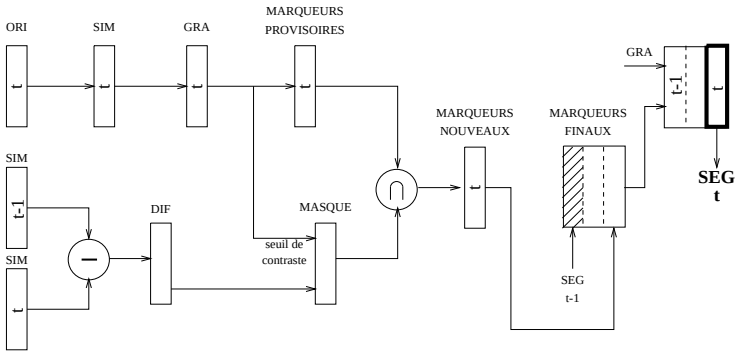


Figure 5.9: Diagramme de l'algorithme d'inclusion de nouvelles régions.

Sur l'image 5.10 nous avons le résultat de l'application de cette méthode à l'image ping-pong.

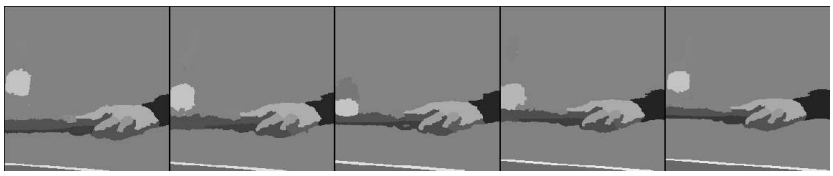


Figure 5.10: Inclusion de nouvelles régions.

5.4 Conclusions

Nous avons présenté un algorithme de segmentation morphologique de séquences d'images qui peut être décomposé en trois étapes:

1. **Simplification d'image**

   Un filtre aréolaire tridimensionnel simplifie l'image courante en éliminant les petites structures qui correspondent essentiellement au bruit. Les structures qui résistent au filtrage conservent leur contour puisque le filtre aréolaire est un filtre connexe.

2. **Sélection de marqueurs**

   - intra:
     Les minima du gradient sont candidats à être marqueurs de régions homogènes. A chaque minimum on associe une valeur, en fonction du contraste avec ses minima voisins (dynamique) et de sa taille, qui indique son importance visuelle. Les marqueurs sont ordonnés selon cette valuation et les plus visibles sont sélectionnés.
   - inter: Les mesures de l'importance des marqueurs calculées en INTRA peuvent être très proches les unes des autres. Par conséquent leur classement calculé indépendamment pour deux temps successifs peut ne pas être identique, même si les deux images se ressemblent

beaucoup. La sélection indépendante de marqueurs pour chaque temps de la séquence rend la segmentation instable. Pour imposer la stabilité nécessaire, qui nous mène à un haut taux de compression, nous utilisons la segmentation précédente comme jeu de marqueurs du temps courant. L'algorithme est complété par l'inclusion de marqueurs de nouvelles régions quand ils sont validés par un masque de changement important.

3. **Localisation des contours**

   Une fois que les marqueurs ont été sélectionnés nous devons localiser les contours précis de chacun d'eux. Le contour entre deux marqueurs est situé sur la ligne de plus grand contraste qui les sépare, c'est-à-dire sur la ligne de partage des eaux de l'image gradient.

Cet algorithme (gradient + marqueurs + ligne de partage des eaux) a été utilisé dans des nombreuses applications avec des résultats satisfaisants. Pour citer quelques exemples nous pouvons parler des applications médicales pour déceler des cellules cancéreuses [51], des applications à l'analyse des matériaux pour étudier par exemple les caractéristiques d'une mousse de polyuréthane [14], et même des applications qui ont abouti à une implémentation hardware en temps réel comme celle du projet PROMETHEUS, qui a permis d'avancer dans la réalisation d'un système de conduite automatique. Néanmoins les succès obtenus pour une application de codage sont plus modestes. Les raisons sont les suivantes:

- L'utilisation du gradient morphologique sans perte d'information nécessite la vérification des conditions d'échantillonnage morphologique: l'image doit être invariante par ouverture et fermeture de taille 1. Ces conditions ne sont pas remplies par les images réelles ce qui mène à la perte des objets fins. Cette perte de détails, qui a priori peut être négligeable, nous mène à une perte de qualité importante quand ces détails ont une signification (comme par exemple les yeux ou le nez d'un personnage).

- En ce qui concerne la segmentation tridimensionnelle le gradient a dû être sur-échantillonné dans l'axe temporel pour placer correctement les contours des objets rapides (objets fins dans la dimension temporelle). Cette solution réduit l'incertitude du gradient tridimensionnel à un pixel (comme le bidimensionnel) mais en même temps elle augmente considérablement le temps de calcul.

- Le fait d'utiliser un gradient tridimensionnel pour segmenter une séquence, exige une connexion d'au moins deux pixels d'épaisseur pour passer un label d'un temps à l'autre. Cela rend difficile la continuité des petites régions, autrement dit, le problème de résolution du gradient, déjà vu en deux dimensions, est aggravé en 3 dimensions.

- La dynamique, mesure de contraste entre deux régions, est définie comme la valeur minimale du gradient tout au long de la frontière entre les deux régions. Cette mesure n'est pas robuste puisqu'un passage fin de bas niveau sur une longue frontière contrastée implique une petite valuation de la frontière. Une valuation plus globale de la frontière serait mieux

adaptée qu'une valeur minimale, mais cette idée est difficile à mettre en œuvre dans cet algorithme parce que les frontières ne sont pas disponibles au moment de choisir les marqueurs.

A cause de ces handicaps, nous allons explorer d'autres chemins pour chercher des solutions mieux adaptées au problème du codage.

Chapter 6

Discussion

Nous avons commencé le travail de cette thèse par l'application de l'algorithme classique de la morphologie mathématique au codage. L'algorithme dont on parle, tel qu'il est présenté dans le chapitre 4, consiste à calculer la ligne de partage des eaux de l'image gradient à partir des marqueurs des régions que l'on veut extraire. L'algorithme est ainsi décomposé en deux étapes indépendantes: la détermination des marqueurs (principale difficulté de la segmentation) et l'extraction des contours (étape automatique réalisée par la ligne de partage des eaux).

Ce schéma a été utilisé avec succès dans de nombreuses situations. Néanmoins son application au codage mène à des difficultés importantes. Dans ce chapitre nous analysons ces difficultés et nous proposons d'autres méthodes qui évitent les problèmes inhérents au gradient.

6.1 Inconvénients de la méthode basée sur le gradient

Revoyons rapidement les caractéristiques de cette méthode qui la rendent inadaptée pour une application de codage.

- L'utilisation du gradient nous mène à une image segmentée dépourvue de détails. Ce manque de détails peut nous sembler a priori négligeable mais, en pratique, il nous mène à une perte de qualité importante. En effet, quand ces détails fins correspondent à des objets qui ont une signification (par exemple, les yeux ou le nez d'un personnage), leur absence est vraiment marquante.

  Pour éviter ce problème nous avons proposé de décomposer l'image en deux parties. Ces parties sont traitées séparément et combinées à la fin du processus pour générer la segmentation finale. La décomposition consiste à séparer les détails fins, qui sont en dessous de la résolution du gradient, des grandes régions. Le problème de cette approche est de trouver un bon équilibre entre la quantité d'information choisie dans chacune des deux composantes. Combien de détails fins par rapport au nombre des grandes structures devons nous préserver? Du fait que leur classement est indépendant nous ne savons pas quelle composante a plus d'information.

Le même problème est rencontré dans [44]. Il est question d'un algorithme de segmentation hiérarchique qui commence par générer une segmentation grossière et qui au cours de plusieurs étapes rajoute des régions de plus en plus fines à l'intérieur des régions précédentes. Cet algorithme doit choisir à chaque étape un certain nombre de régions. Or, le choix du nombre de régions à chaque étape reste un problème difficile, qui est dépendant des caractéristiques de l'image. En effet, si les premiers niveaux de la hiérarchie sont trop pauvres la modélisation des régions obtenues sera de mauvaise qualité et la recherche de régions dans son résidu sera difficile. Si par contre, le nombre de régions dans les premiers niveaux de la hiérarchie est trop élevé, on ne disposera pas de suffisamment de bits pour coder les niveaux bas de la hiérarchie.

Une autre solution à ce problème consisterait à sur-échantillonner l'image de manière à générer l'espace nécessaire pour placer un contour entre deux détails fins. Dans le cas bidimensionnel, cette solution multiplie par 4 (par 8 dans le cas tridimensionnel) le temps de calcul et les besoins en capacité mémoire.

- En ce qui concerne la segmentation tridimensionnelle, le gradient 3D des objets rapides présente une grande zone d'incertitude qui mène à des défauts encore plus visibles qu'en 2D. Pour placer correctement les contours des objets rapides (objets fins dans la dimension temporelle), le gradient a dû impérativement être sur-échantillonné dans l'axe temporel. Cette solution réduit l'incertitude du gradient tridimensionnel à un pixel (comme le bidimensionnel) mais en même temps elle augmente considérablement le temps de calcul.

- La dynamique, mesure de contraste entre deux régions, est définie comme la valeur minimale du gradient tout au long de la frontière entre les deux régions. Cette mesure n'est pas robuste puisqu'un passage fin de bas niveau sur une longue frontière contrastée implique une petite valuation de la frontière. Une valuation plus globale de la frontière serait mieux adaptée qu'une valeur minimale, mais cette idée est difficile à mettre en œuvre dans cet algorithme parce que les frontières ne sont pas disponibles au moment de choisir les marqueurs.

  Dans la section 4.6 nous avons proposé de réévaluer la dynamique en plusieurs étapes de sur-segmentation. Cela nous permet d'évaluer le contraste moyen entre deux régions tout au long de leur frontière, ce qui donne une valeur plus fiable que la valeur minimale.

  La question qui se pose alors est la suivante : combien d'étapes de sursegmentation sont nécessaires et avec quel nombre de régions chacune? Un nombre élevé d'étapes intermédiaires améliorerait le résultat, mais en même temps ralentirait l'algorithme (à chaque étape on doit calculer la dynamique des marqueurs et effectuer une ligne de partage des eaux).

  L'idée de réévaluer les frontières d'une sursegmentation nous semble intéressante. Nous la retenons pour la développer de manière plus efficace dans le cadre des algorithmes que nous présentons dans la suite.

6.2 Autres méthodes de segmentation

L'algorithme de segmentation basé sur la ligne de partage des eaux présente des inconvénients qui rendent la méthode peu adaptée à une application de codage. Voyons d'autres méthodes qui évitent les problèmes inhérents au gradient. La figure 6.1 illustre le fonctionnement de ces trois méthodes.

6.2.1 Croissance de régions à l'échelle du pixel

Cette méthode [32, 40], de la même façon que la ligne de partage des eaux, consiste à assigner itérativement les pixels autour des marqueurs aux marqueurs, suivant un critère de ressemblance, jusqu'à ce que tout pixel soit assigné à un marqueur. Contrairement à l'algorithme de la ligne de partage des eaux, le support pour faire croître les marqueurs n'est plus une image gradient mais l'image originale (ou filtrée) elle même. Cela permet de suivre les objets fins, pourvu qu'ils contiennent un marqueur.

6.2.2 Croissance de régions à l'échelle des zones plates

Cette méthode [9, 8] applique le concept d'opérateur connexe à la segmentation. Il ne travaille pas à l'échelle du pixel mais de la zone plate. Les zones plates sont les plus grandes composantes connexes pour lesquelles la valeur de la fonction (en l'occurrence le niveau de gris, la couleur ...) est constante.

L'algorithme consiste à signaler certaines zones plates comme marqueurs de régions importantes. Les zones non marquées sont assignées, une à une, à un marqueur de son voisinage jusqu'à ce que toute l'image soit recouverte. Cette méthode assure l'inclusion d'une zone plate de l'image d'entrée dans une zone plate de l'image de sortie. Par conséquent les contours finaux correspondent à des contours qui existaient déjà sur l'image de départ.

Toute zone plate, indépendamment de sa taille, est traitée, ce qui permet de représenter les objets fins.

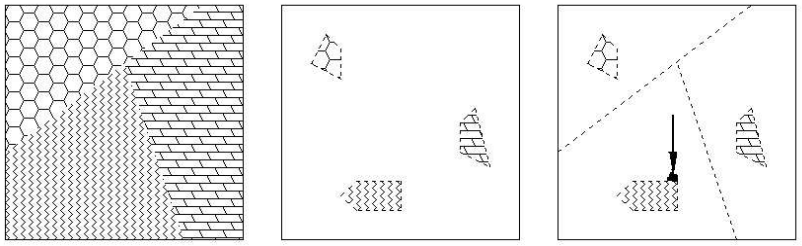
Remarquons qu'un jeu de marqueurs pour démarrer le processus de croissance est encore nécessaire.

6.2.3 Fusion de régions

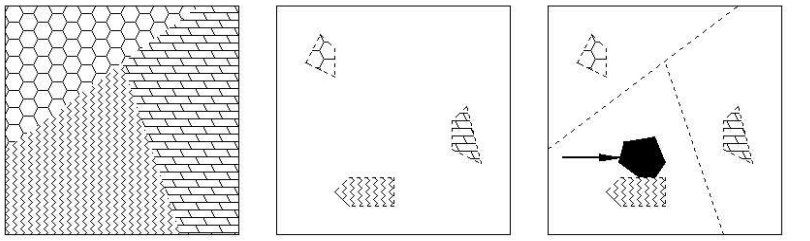
Cette méthode [36] travaille aussi à l'échelle des zones plates, mais contrairement à la méthode précédente, le jeu de marqueurs signalant les régions importantes n'est plus nécessaire pour démarrer le processus. L'algorithme consiste à fusionner itérativement les deux régions (zones plates) les plus similaires de l'image jusqu'à ce qu'un critère d'arrêt soit atteint.

6.3 Discussion

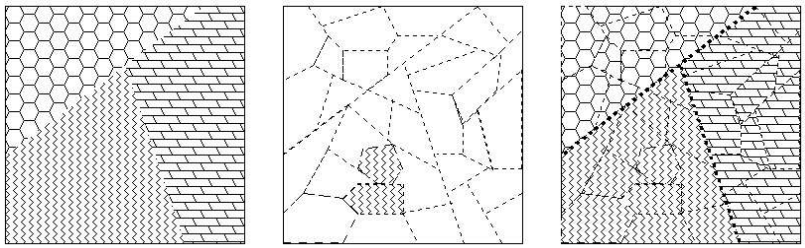
Analysons, à la lumière des connaissances acquises dans les chapitres précédents, les avantages et les inconvénients des méthodes que nous avons décrites dans la section 6.2. En particulier, étudions si elles apportent une solution aux problèmes posés par la segmentation basée sur le gradient.



CROISSANCE DE RÉGIONS À L'ÉCHELLE DU PIXEL.



CROISSANCE DE RÉGIONS À L'ÉCHELLE DE ZONES PLATES.



FUSION DE RÉGIONS.

Figure 6.1: Méthodes de segmentation évitant la perte de résolution

N'oublions pas que ces méthodes sont valables aussi bien pour segmenter des images bidimensionnelles que tridimensionnelles.

6.3.1 Résolution

Les trois méthodes décrites dans la section 6.2 évitent le problème de résolution inhérent au gradient. Le support pour faire croître les marqueurs (ou réaliser des fusions) n'est plus une image gradient, mais l'image originale elle même. Cela permet de suivre les objets fins sans avoir besoin de sur-échantillonner l'image.

6.3.2 Problèmes de fuites

Les trois méthodes décrites dans la section 6.2 évitent la perte de résolution inhérente au gradient. En contrepartie de cette flexibilité pour représenter n'importe quelle forme (y compris les détails fins), elles présentent dans certaines situations des problèmes de "fuites", comme l'illustre l'exemple suivant.

Prenons l'image de la figure 6.2(a). Elle comporte trois zones: une blanche, une noire et une grise, séparées par des zones de transition. Choisissons un marqueur à l'intérieur de chacune de ces zones. Si nous faisons croître ces marqueurs selon le contraste indiqué par le gradient, nous obtenons la segmentation de la figure 6.2(b). Par contre, si nous les faisons croître sur l'image originale le résultat est celui de la figure 6.2(c). Dans ce dernier cas, M_1 envahit la zone de transition entre les régions 2 et 3, générant un long contour qui ne correspond à aucun détail de l'image. Il s'agit d'une fuite du marqueur M_1 à cause de la ressemblance entre son niveau de gris et celui de certains pixels de la zone de transition $M_2 - M_3$.

Dans cette situation la segmentation basée sur le gradient est plus performante. Pour utiliser les autres méthodes, nous devrons prendre des précautions pour éviter ces fuites.

(a) Image originale (b) Croissance de marqueurs sur l'image gradient (c) Croissance de marqueurs sur l'image originale

Figure 6.2: Problème de fuites

6.3.3 Du pixel à la zone plate

L'algorithme de ligne de partage des eaux fait croître des marqueurs pixel à pixel, jusqu'à ce qu'ils occupent la totalité de l'image. Si, au lieu de considérer des pixels, on s'intéresse aux zones plates et à leurs relations de voisinage, on aboutit à des méthodes qui ont les avantages suivants:

- Les contours finaux correspondent à des contours qui existaient déjà sur l'image de départ. Les formes des objets peuvent alors être préservées.
- Travailler à l'échelle de la zone plate (et non pas du pixel) permet une analyse de l'image à plus haut niveau; on peut associer à une zone plate des caractéristiques plus significatives qu'à un pixel, comme par exemple une représentation de texture, un vecteur de mouvement...

La croissance de régions à l'échelle des zones plates et la fusion de régions travaillent à l'échelle des zones plates, bénéficiant ainsi de ces avantages.

6.3.4 Sélection de marqueurs

Dans le cadre du codage, la segmentation doit être adaptée au taux de compression visé. Or, il est difficile de prévoir un coût de codage à partir d'un jeu de marqueurs. Par contre, la méthode par fusion de régions connaît à chaque instant l'état de la segmentation, ce qui lui permet de mieux estimer le coût de codage.

Par ailleurs la méthode de fusion de régions offre un cadre flexible pour développer des critères évolués de fusion, tels que la texture ou le mouvement. Cette caractéristique est un avantage considérable par rapport aux autres méthodes parce qu'il n'est pas facile de dériver un jeu de marqueurs à partir de ces critères évolués. Nous reviendrons sur ce point dans le chapitre 9.

Dans le cadre du codage la meilleure segmentation est celle qui nous donne la meilleure qualité avec le plus petit coût. Il n'est pas facile de dériver un jeu de marqueurs à partir de ce critère. Les techniques de fusion nous permettent d'obtenir une segmentation de façon naturelle, fusionnant itérativement les deux régions qui se ressemblent le plus, sans introduire artificiellement un jeu de marqueurs.

6.3.5 Généralisation de la réévaluation de la dynamique

Nous avons vu dans la section 4.6 que la réévaluation de la dynamique en plusieurs étapes de sur-segmentation permet de mieux estimer le contraste entre régions. L'importance visuelle des régions est ainsi plus pertinente et en conséquence les résultats obtenus sont de meilleure qualité.

Les étapes successives de sur-segmentation sont en fait, des fusions de régions basées sur une évaluation associée aux frontières. Sur une sur-segmentation, nous associons une valeur à chaque frontière, dépendante de la ressemblance des deux régions qu'elle sépare. Certaines de ces frontières sont éliminées générant ainsi une segmentation plus grossière. Les frontières de cette segmentation sont réévaluées à nouveau et une nouvelle simplification est obtenue. Le processus est itéré jusqu'à ce que la segmentation finale ait le nombre de régions visé.

L'algorithme de fusion de régions est une généralisation de ce principe. Il étudie de manière systématique toutes les frontières d'une segmentation. A chaque itération, il élimine une frontière (la moins importante, dans un sens à définir) et s'intéresse aux changements que son élimination implique aux alentours. Autrement dit, il s'intéresse à donner une évaluation globale aux frontières de la nouvelle segmentation. Ainsi, l'algorithme de fusion de régions

réévalue en permanence les frontières de la segmentation. L'évaluation des frontières correspond toujours à l'état courant de la partition, ce qui nous permet de leur donner une évaluation globale et pertinente de leur importance.

6.4 Conclusion

L'algorithme qui réunit les meilleurs caractéristiques pour l'application que nous développons est l'algorithme de fusion de régions. Récapitulons ses avantages:

- Il élimine le problème de résolution inhérent au gradient parce qu'il s'intéresse à toutes les zones plates, indépendamment de leur taille.
- Il travaille avec les zones plates d'une image, ce qui permet une analyse de l'image à plus haut niveau; on peut associer à une zone plate des caractéristiques plus significatives qu'à un pixel, comme par exemple une représentation de texture, un vecteur de mouvement...
- Il fusionne itérativement les deux régions qui se ressemblent le plus, sans introduire artificiellement un jeu de marqueurs.
- Il réévalue en permanence l'importance des frontières, ce qui lui permet de travailler avec une évaluation globale des frontières, évitant ainsi la fragilité de la dynamique.
- Il produit des partitions emboîtées sur lesquelles on peut appliquer un arbre de décision pour maximiser la qualité pour un coût donné.

L'inconvénient principal de cette approche est lié au problème des fuites à travers les zones de transition. Nous traiterons ce problème dans les chapitres qui suivent.

Chapter 7

Algorithme de Fusion de Régions

7.1 Introduction

L'algorithme de segmentation basé sur le gradient ne se révèle pas très adapté à une application de codage à cause de son incapacité à représenter les petits objets. L'impossibilité de représenter les objets fins mène à une perte importante de qualité visuelle et rend difficile le passage de labels à travers le temps.

Nous avons vu dans le chapitre 6, dans leurs grandes lignes, d'autres méthodes de segmentation et nous avons conclu que l'algorithme de fusion de régions est celui qui s'adapte le mieux à notre application. Détaillons dans ce chapitre le fonctionnement de cet algorithme et décrivons nos choix algorithmiques pour obtenir une implémentation efficace.

7.2 Description de l'algorithme de fusion de régions

Les zones plates d'une image sont les plus grandes composantes connexes de pixels pour lesquelles la valeur de la fonction (en l'occurrence le niveau de gris, la couleur ...) est constante. Considérer les zones plates comme des entités indivisibles, au lieu de traiter les pixels isolés, est le principe des opérateurs connexes. Ceci comporte plusieurs avantages:

- Tout pixel appartient à une zone plate (les pixels dont tous les voisins ont une valuation différente sont des zones plates à un pixel). Un algorithme qui agit sur les zones plates assure le traitement de toute structure de l'image, quelle que soit sa taille. Ainsi le problème de résolution associé au gradient n'existe plus.

- Le fait de fusionner des zones plates élimine des contours mais n'en génère jamais de nouveaux. Cela permet la préservation de la forme des objets.

- Une fois les premières fusions réalisées, avec un critère simple tel que le contraste, les nouvelles régions obtenues peuvent être munies de carac-

téristiques plus complexes, telles qu'une représentation de texture ou un vecteur de mouvement.

L'ensemble des zones plates de l'image constitue déjà une partition parce qu'il vérifie les deux conditions suivantes:

- l'union des zones plates recouvre entièrement le domaine
- l'intersection de deux zones plates différentes est toujours nulle; les zones plates sont disjointes.

Cette partition est néanmoins trop riche pour être codée. La segmentation consiste à élargir les zones plates, générant ainsi de zones plus grandes en plus petit nombre, correspondant le plus fidèlement possible aux objets présents dans la scène.

L'algorithme de fusion de régions [36] est un processus itératif. A chaque itération on obtient une nouvelle partition dans laquelle les deux régions adjacentes les plus similaires sont fusionnées. Formalisons cette notation:

Soit P_0 une partition initiale constituée de N_0 régions: $P_{01}, P_{02}, P_{03}...P_{0N_0}$. L'algorithme de fusion de régions construit une suite de partitions de plus en plus grossières; P_i est la partition obtenue après i étapes de fusion, $P_i = \{P_{i1}, P_{i2}, P_{i3}...P_{iN_i}\}$. Le nombre de régions de P_i est donc égal au nombre de régions de P_0 moins le nombre de fusions réalisées: $N_i = N_0 - i$. Il s'agit donc d'un algorithme de segmentation ascendant ("bottom-up"). Le processus de fusion est itéré jusqu'à ce qu'un critère d'arrêt soit atteint. La mesure de similitude entre régions adjacentes et le critère d'arrêt restent à définir.

Pour passer de la partition P_i à la partition P_{i+1}, on fusionne les deux régions adjacentes les plus similaires, P_{ij} et P_{ik}. Donc, l'opération de fusion se traduit par:

$$P_{i+1\ l} = P_{ij} \cup P_{ik}$$

Le pseudo-code de l'algorithme est le suivant:

```
procédure fusion{
  évaluer la ressemblance de tout couple de régions adjacentes;
  arrêt = vérifier si critère d'arrêt est atteint;
  tant_que(! arrêt){
    (P_ij, P_ik) = trouver le couple de régions le plus similaire;
    fusionner(P_ij, P_ik);
    actualiser les valuations des nouvelles relations de voisinage;
    arrêt = vérifier si critère d'arrêt est atteint;
  } /* fin_tant_que */
} /* fin_procédure */
```

Ainsi, une région de la partition P_i correspond à la réunion d'un certain nombre de régions adjacentes de P_0 (principe des opérateurs connexes). Dans ce qui suit, nous appellerons "macro-région" l'union connexe des régions d'une partition et nous dirons que les régions de P_i sont des macro-régions de P_0.

Dans la suite nous appellerons "frontière" entre deux régions adjacentes, P_{ij} et P_{ik}, leur contour commun et nous la noterons $(P_{ij} - P_{ik})$. Nous remarquons

que la frontière entre deux régions adjacentes P_{ij} et P_{ik} de la partition P_i, est une union de frontières du type $P_{0l} - P_{0t}$ de la partition initiale où:

$$P_{0l} \subseteq P_{ij}$$

$$P_{0t} \subseteq P_{ik}$$

Quand P_{ij} et P_{ik} fusionnent nous devons éliminer tous les maillons $(P_{0l} - P_{0t})$ de leur frontière. La raison en est simple: entre deux régions d'une même macro-région ne peut pas exister une frontière de séparation. Nous désignerons la frontière entre P_{ij} et P_{ik} par le terme macro-frontière, pour insister sur le fait qu'elle est constituée d'une ou plusieurs frontières entre régions de la partition P_0. Puisque toutes les frontières d'une macro-frontière sont dépendantes les unes des autres (l'élimination d'une frontière implique l'élimination de toutes) sur quel critère devons-nous baser la fusion de deux macro-régions P_{ij} et P_{ik} ? Il est évident qu'il ne faut pas se baser sur le maillon le plus faible ainsi que le montre l'exemple de la figure 7.1: P_{i2} et P_{i3} sont voisines et apparaissent comme très différentes. Pourtant elles contiennent deux régions adjacentes $P_{0l} \subseteq P_{i2}$ et $P_{0t} \subseteq P_{i3}$ qui, considérées individuellement, apparaissent comme très semblables. Baser la fusion sur le maillon le plus faible aurait conduit à fusionner P_{ij} et P_{ik}.

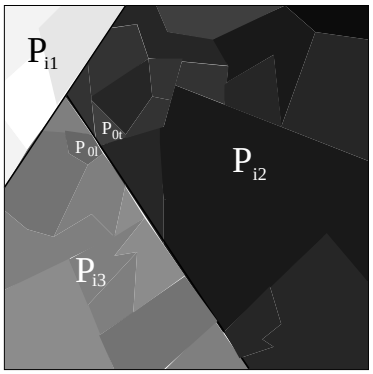


Figure 7.1: Réactualisation?

Ainsi la notion de macro-frontière joue un rôle important. Elle permet de prendre en compte les fusions déjà effectuées pour décider quelle est la fusion suivante à réaliser; la caractérisation globale des macro-frontières donne de la robustesse à l'algorithme.

7.3 Mise en œuvre

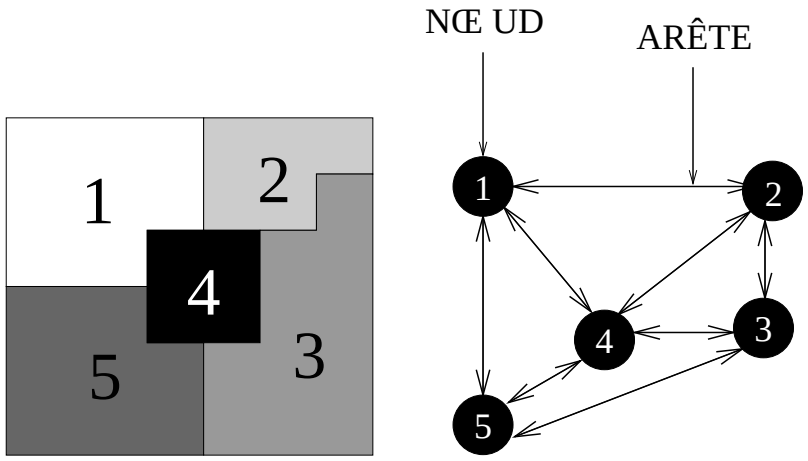
Pour mettre en œuvre un algorithme de fusion de régions, nous avons dû résoudre un certain nombre de problèmes que nous allons détailler dans la suite.

7.3.1 Représentation de l'information

Le premier problème à résoudre est la façon de représenter l'information relative à l'image. La manière courante de représenter une image est un tableau de

pixels. Or, l'algorithme doit tout au long de la procédure choisir un couple de régions à fusionner parmi tous les couples possibles, et gérer efficacement les relations de voisinage changeantes après chaque fusion. Pour cela il est nécessaire de disposer d'un accès rapide aux régions et aux relations de voisinage entre elles, ce qui n'est pas le cas sur un tableau de pixels. La structure de données la mieux adaptée à ce problème est une structure de graphe, composée de nœuds et d'arêtes. Les nœuds représentent les entités d'intérêt et les arêtes les relations entre les nœuds.

Si nous associons un graphe à une partition, les nœuds représentent les régions segmentées et les arêtes les relations de voisinage. La figure 7.2 illustre la relation entre une partition et son graphe associé.



(a) Partition (b) Graphe associé

Figure 7.2: Partitions et Graphes

Les arêtes correspondent aux couples de régions adjacentes, c'est-à-dire, aux fusions possibles. Leur valuation, une mesure de similitude, nous donne l'ordre de fusion: l'arête de plus petite valuation sera la première à être éliminée parce qu'elle sépare les deux régions les plus semblables.

7.3.2 Gestion de macro-régions

La procédure de fusion de régions commence par générer un graphe à partir de la partition initiale (P_0); les nœuds du graphe correspondent alors aux régions de P_0 (macro-régions initiales) et les arêtes aux frontières de P_0 (macro-frontières initiales). Ensuite, les deux régions qui se ressemblent le plus, P_{0j} et P_{0k}, fusionnent. Après cette fusion l'image comporte une région P_{1l} qui est l'union des deux régions initiales ($P_{1l} = P_{0j} \cup P_{0k}$) et dont les voisines sont $V_{1l} = (V_{0j} \cup V_{0k}) \backslash \{P_{0j} \cup P_{0k}\}$, où V_{0j} (resp. V_{0k}) est l'ensemble de voisines de P_{0j} (resp. P_{0k}).

Ainsi, après chaque fusion, les relations de voisinage changent et le graphe doit évoluer en conséquence. Une solution pour gérer les macro-régions à l'intérieur du graphe serait d'éliminer à chaque fusion une des deux régions fusionnées (un nœud). L'élimination d'un nœud implique l'élimination de toute référence à lui dans les relations de voisinage et la réorganisation de la liste d'arêtes. Cette so-

lution perd de vue toute région initiale qui a fusionné, ce qui pose des problèmes pour récupérer l'image segmentée issue du processus de fusion. C'est pourquoi nous avons préféré garder le graphe initial et associer une deuxième valuation (la première étant la mesure de ressemblance entre régions voisines) aux arêtes qui exprime leur statut. Cette solution permet de remonter éventuellement en arrière dans l'historique des fusions réalisées. Une arête peut avoir deux statut:

- soit l'arête est encore en vigueur, c'est-à-dire, elle établit une relation de voisinage entre deux régions.
- soit l'arête est déjà fusionnée, c'est-à-dire, elle indique que les deux régions adjacentes ont déjà fusionné.

Ainsi, à tout moment nous pouvons récupérer les macro-régions en cours, grâce à une reconstruction de grains [54] sur le graphe original. La reconstruction de grains consiste à regrouper à partir d'une région P_{0j}, toutes les régions auxquelles nous pouvons accéder traversant seulement des frontières déjà fusionnées. En d'autres termes, il s'agit de retrouver dans la partition P_i la macro-région P_{ik} associée à P_{0j}. Le pseudo-code de cette procédure est détaillé ci après:

```
procédure reconstruire_macro_région_et_trouver_ses_voisins(P_0n){
   macro_région            = P_0n; /* initialisation de la macro_région */
   voisins_macro_région    = null; /* initialisation des voisins */
   pour toute région R { déjà_traitée (R) = faux }
   R = P_0n;
   tant_que( R != null){
        pour toute région Q voisine de R{ /* information extraite
                                             du graphe initial */
          si ( déjà_traitée(Q) == faux ){ /* si Q n'a pas encore
                                              été traitée */
             déjà_traitée(Q) = vrai;
             si  (statut(R,Q) == fusionné){ /* le couple est déjà fusionné */
               /* Q fait partie de la macro_région */
               macro_région = add(macro_région,Q);
             }
             sinon { /* le couple de régions n'est pas encore fusionné */
               /* Q est un voisin de la macro_région parce qu'il est voisin
                  d'une de ses régions */
               voisins_macro_région = add(voisins_macro_région,Q);
             }
         } /* fin_si   (déjà_traitée (Q) == faux)  */
     } /* fin_pour toute région Q voisine de R */
     R = région_suivante_de_la_macro_région;
   } /* fin_tant_que */
} /* fin_procédure */
```

7.3.3 Gestion de macro-arêtes

Nous avons introduit précédemment la notion de macro-frontière; une macro-frontière est l'union de toutes les frontières de la partition initiale qui séparent

deux macro-régions. L'équivalent algorithmique d'une macro-frontière est une macro-arête. Nous avons vu aussi que toutes les frontières d'une même macro-frontière sont éliminées simultanément et qu'une valuation globale de l'ensemble donne de la robustesse à l'algorithme de fusion. Comment gérer alors la valuation exprimant la force de la frontière entre P_{i+1l} et P_{i+1t}? Faut-il mettre à jour toutes les frontières d'une macro-frontière? En fait, on va particulariser une des arêtes qui sera traitée comme la représentante de toutes les arêtes entre deux régions (macro-régions). Nous introduisons maintenant la notion d'"arête représentante".

Définition 7.1 *L'arête représentante d'une macro-arête est celle à laquelle on associe la nouvelle valuation globale d'une macro-arête.*

La notion d'arête représentante va nous aider à maintenir à jour la liste d'arêtes à éliminer, ainsi que leur classement par ordre de fusion. Précédemment nous avons associé une valuation binaire aux arêtes, afin de pouvoir reconstruire les macro-régions constituées. Maintenant cette valuation va admettre trois possibilités:

- REPRÉSENTANT: l'arête n'est pas encore éliminée et elle contient la valuation actualisée, c'est-à-dire, elle est à jour.
- NON-ACTUALISÉ: l'arête n'a pas encore été éliminée, mais sa valuation n'est plus à jour. Son élimination dépend d'une autre arête: son représentant.
- FUSIONNÉ: l'arête a déjà été fusionnée.

Après une fusion nous devons vérifier le statut des frontières qui entourent la nouvelle macro-région générée. Pour cela nous reconstruisons les macro-régions voisines. Pour chaque macro-arête il faudra choisir une arête représentante. Toutes les arêtes qui constituent une macro-arête ont la même importance. C'est pourquoi nous choisissons comme représentante d'une macro-arête, sa première arête rencontrée.

7.3.4 Classement dynamique des arêtes par ordre de valuation

Nous devons être capables de trouver de manière efficace l'arête de plus petite valuation parmi celles qui sont encore représentantes, sachant qu'après chaque fusion les arêtes autour de la nouvelle région peuvent changer de valuation.

Une solution est de parcourir toutes les arêtes à chaque fusion pour trouver celle de valuation minimale. Cette solution est la plus simple à programmer, mais elle est aussi la plus lourde en termes de temps de calcul.

Une autre solution est d'ordonner les arêtes dans une liste avec leur valuation initiale et de changer de place celles qui ont changé de valuation. Trouver l'emplacement d'une arête dans la liste et réorganiser la liste pour mettre l'arête à sa place sont aussi des procédures coûteuses en temps de calcul.

La file d'attente hiérarchique [30] offre une solution rapide à ce problème. Une file d'attente est tout simplement une structure linéaire avec une entrée et une sortie à des extrêmes opposées (voir fig. 7.3(a)). Les éléments sortent de la file dans leur ordre d'entrée. En termes informatiques il s'agit d'une

structure FIFO (First In - First Out). Une file d'attente hiérarchique est un ensemble de files d'attente simples. Les files sont ordonnées et chacune accueille des éléments d'une priorité différente. Des éléments de toute priorité peuvent arriver à n'importe quel moment et rentrer dans leur file correspondante. Par contre, à chaque instant, seul le premier élément de la file de plus haute priorité peut sortir de la file d'attente (voir fig. 7.3(b)).

Si on introduit toutes les arêtes dans une file d'attente hiérarchique selon leur valuation, l'ordre de sortie sera bien l'ordre de fusion. Pour traiter le cas de mise à jour des valuations après chaque fusion nous pouvons procéder de la façon suivante: une arête rentre dans la file d'attente autant de fois qu'elle est re-actualisée, chaque fois avec une priorité différente. Après, nous devons nous débarrasser des exemplaires multiples et traiter seulement l'arête qui correspond à la dernière valuation.

Il est intéressant de prendre en compte les macro-arêtes au moment d'actualiser la valuation des frontières après une fusion pour les raisons suivantes:

1. parce que cela nous permet de donner une évaluation globale à la nouvelle macro-arête;

2. pour ne pas introduire dans la file d'attente plusieurs fois ce qui est devenu "la même arête".

Pour nous débarrasser des exemplaires multiples d'une macro-arête, un test supplémentaire est rajouté à la sortie de la file d'attente. L'arête sera traitée seulement si:

- elle a le statut "REPRESENTANT" ce qui veut dire qu'elle contient la valuation actualisée.

- sa valuation correspond à la priorité de la file d'attente de laquelle elle sort; c'est bien la dernière valuation qu'elle a reçu.

Cette solution est plus simple et plus rapide que de maintenir la file à jour avec la présence unique de chaque arête.

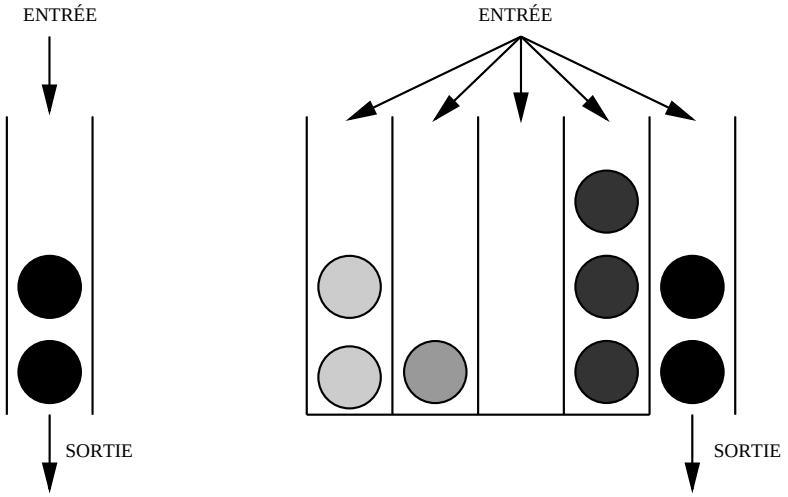


Figure 7.3: Files d'attente

7.3.5 Pseudo-code de l'algorithme.

```
procédure SEGMENTATION_PAR_FUSION(im) {
fah = générer_file_d'attente_hiérarchique;
gr = générer_graphe(im);
fah = introduire_arêtes_dans_fah(gr);
arrêt = vérifier si critère d'arrêt est atteint;
tant_que( (! arrêt) et (! fah_vide)){
  (P_0l,P_0t) = extraire(fah);
  si ( (priorité_en_cours(fah) == valuation(P_0l,P_0t))
       et (statut(P_0l,P_0t) == représentant)){
    pour toute région P_0n {      appartient_MR_à_jour (P_0n) = faux    }
    /* ``appartient_MR_à_jour'' indique si la région a déjà été mise à jour
          après la fusion réalisée. */
    P_ij = reconstruire_macro_région(P_0l);
    P_ik = reconstruire_macro_région(P_0t);
    v1  = trouver_voisins(P_ij);
    v2  = trouver_voisins(P_ik);
    pour toute région P_0n {
        /* éliminer macro_arête(P_0l,P_0t) */
       si ( (P_0n ∈ v1) et (P_0n ⊆ P_ik) ){
         pour toute région P_0m voisine de P_0n{
           si (P_0m ⊆ P_ij){ statut(P_0n,P_0m) = fusionné }
         } }
       si ( (P_0n ∈ v2) et (P_0n ⊆ P_ij) ){
         pour toute région P_0m voisine de P_0n{
           si (P_0m ⊆ P_ik){ statut(P_0n,P_0m) = fusionné }
         } }
       /* actualiser arêtes autour de la nouvelle macro_région */
       si ( ((P_0n ∈ v1) ou (P_0n ∈ v2))
            et (P_0n ∉ P_ij) et (P_0n ∉ P_ik)
            et (appartient_MR_à_jour(P_0n) == faux)){
         MR = reconstruire_macro_région (P_0n);
         pour toute région P_0m de MR { appartient_MR_à_jour (P_0m) = vrai }
         (R,Q) = représentante_de_macro_arête(MR, P_ij ∪ P_ik) ;
         autres_arêtes_de_macro_arête(R,Q) = non-actualisé;
         réévaluer(R,Q);
         fah = introduire_en_fah(R,Q);
       }
       arrêt = vérifier si critère d'arrêt est atteint;
  } /* fin_pour_toute_P_0n */
  } /* fin_si */
} /* fin_tant_que */
   im = générer_image(gr,statut);
   libérer_fah(fah);
   libérer_graphe(gr);
} /* fin_procédure */
```

7.4 Exemple pédagogique

Nous avons donc développé un algorithme de segmentation bottom-up, basé sur la fusion de couples de régions. Pour pouvoir l'utiliser nous devons définir une fonction qui mesure la ressemblance entre deux régions et un critère d'arrêt. Commençons par des critères simples: considérons comme distance entre deux régions la différence en valeur absolue de leur niveau de gris moyen, et comme critère d'arrêt un nombre de points de contours donné (en fonction du nombre de bits disponibles pour coder l'image). Nous prendrons comme critère d'arrêt 9 points de contour.

Voyons en détail l'évolution de l'algorithme sur une image simple. L'image originale est sur la figure 7.4(a). Ses dimensions sont 6×6 pixels. Elle contient 5 régions avec les caractéristiques suivantes:

REGIONS	R1	R2	R3	R4	R5
NIVEAU DE GRIS	255	180	150	160	210
SURFACE (en pixels)	8	5	11	4	8

Table 7.1: Régions de l'image 7.4(a)

Le graphe associé à cette image contient 5 nœuds et 8 arêtes. Chaque arête apparaît deux fois dans le graphe (R1-R2 $\leftrightarrow$ R2-R1) ce qui nous permet d'accéder directement à tous les voisins d'une région. La longueur des frontières entre régions apparaît sur le tableau 7.2.

R1-R2 2	R2-R1 2	R3-R2 3	R4-R1 2	R5-R1 2
R1-R4 2	R2-R3 3	R3-R4 3	R4-R2 1	R5-R3 2
R1-R5 2	R2-R4 1	R3-R5 2	R4-R3 3	R5-R4 2
			R4-R5 2	

Table 7.2: Longueur d'arêtes

La longueur totale de la frontière de la partition est la somme de la longueur individuelle de chacune des arêtes. Dans notre exemple la longueur totale est de 17 points de contour. Le contenu initial du graphe, si on prend comme critère de fusion la différence de niveau de gris moyen, apparaît sur le tableau 7.3.

Chaque arête est introduite dans la file d'attente hiérarchique, avec une priorité qui correspond à son évaluation.

L'arête R3-R4 sort de la FAH L'arête R3-R4 est la première à sortir parce qu'elle a l'évaluation minimale. Son évaluation correspond à la priorité en cours de la file d'attente et son statut est REPRÉSENTANT. Les deux conditions sont remplies, alors on fusionne les régions R3 et R4.

Après chaque fusion nous devons:

NŒUDS (ÉVALUATION)	R1 (255)	R2 (180)	R3 (150)	R4 (160)	R5 (210)
ARÊTES	R1-R2 (75) [REPR]	R2-R1 (75) [REPR]	R3-R2 (30) [REPR]	R4-R1 (95) [REPR]	R5-R1 (45) [REPR]
ASSOCIÉES	R1-R4 (95) [REPR]	R2-R3 (30) [REPR]	R3-R4 (10) [REPR]	R4-R2 (20) [REPR]	R5-R3 (60) [REPR]
(ÉVALUATION)	R1-R5 (45) [REPR]	R2-R4 (20) [REPR]	R3-R5 (60) [REPR]	R4-R3 (10) [REPR]	R5-R4 (50) [REPR]
[STATUT]				R4-R5 (50) [REPR]	

Table 7.3: Contenu initial du graphe

1. vérifier si d'autres arêtes dépendent de l'arête fusionnée. Cela n'arrive jamais avec la première fusion parce qu'à l'origine toutes les macro-arêtes sont des arêtes simples.

2. actualiser les arêtes autour. Les voisins de la nouvelle macro-région (R3-R4) sont R1, R2 et R5. Le niveau de gris moyen de la nouvelle macro-région est 153 et les nouvelles macro-arêtes avec leur nouvelle évaluation sont:

   R2-R3, R2-R4 $\rightarrow$ 27

   R1-R4 $\rightarrow$ 102

   R3-R5, R4-R5 $\rightarrow$ 57

   Les arêtes R2-R4 et R4-R5 sont mises en statut NON-ACTUALISE et ses représentantes en vigueur sont R2-R3 et R3-R5 respectivement. R2-R3, R1-R4 et R3-R5 re-rentrent dans la file d'attente avec leur nouvelle évaluation. Remarquons que quand une arête est réévaluée, non seulement elle doit être réintroduite dans la file d'attente avec la nouvelle priorité, mais aussi il est nécessaire de changer son évaluation dans le graphe. En effet, imaginons une arête A qui sort de la file d'attente de priorité P. Si la valuation de A dans le graphe, V, est égale à P, cela veut dire que P correspond à la dernière évaluation de A et alors l'arête A est traitée. Si par contre, P est différent de V, cela implique que A a été réévaluée depuis sa rentrée dans la file d'attente avec la priorité P et par conséquent A est ignorée pour l'instant; nous attendons sa sortie de la file de priorité V. Ainsi, le contenu du graphe et le statut d'arêtes après cette fusion apparaît sur le tableau 7.4.

   La longueur de la frontière de la partition actuelle est l'initiale moins la longueur de l'arête éliminée, c'est-à-dire, 17 - 3 = 14. D'autres fusions sont nécessaires pour réduire ce chiffre à 9 (longueur souhaitée).

NŒUDS (ÉVALUATION)	R1 (255)	R2 (180)	R3 (153)	R4 (153)	R5 (210)
ARÊTES	R1-R2 (75) [REPR]	R2-R1 (75) [REPR]	R3-R2 (27) [REPR]	R4-R1 (102) [REPR]	R5-R1 (45) [REPR]
ASSOCIÉES	R1-R4 (102) [REPR]	R2-R3 (27) [REPR]	R3-R4 (10) [FUS]	R4-R2 (20) [N-A]	R5-R3 (57) [REPR]
(ÉVALUATION)	R1-R5 (45) [REPR]	R2-R4 (20) [N-A]	R3-R5 (57) [REPR]	R4-R3 (10) [FUS]	R5-R4 (50) [N-A]
[STATUT]				R4-R5 (50) [N-A]	

Table 7.4: Contenu du graphe après la première fusion R3-R4

L'arête R2-R4 sort de la FAH Le statut de R2-R4 est NON-ACTUALISE, alors on ne la traite pas, on attend la sortie de sa représentante. La longueur de la frontière de la partition actuelle n'a pas changé alors le processus de fusion continue ($14 > 9$).

L'arête R2-R3 sort de la FAH R2-R3 sort de la file de priorité 27 (qui est aussi son évaluation actuelle) et son statut est REPRÉSENTANT; elle est traitée.

- l'arête R2-R3 fait partie d'une macro-arête. Les autres composantes, en statut NON-ACTUALISE, doivent passer au statut FUSIONE. Pour ce faire nous vérifions si les régions voisines de R2 (R3 resp.) font partie en même temps de la macro-région R3 (R2 respectivement). On s'aperçoit que R4 est voisine de R2 et fait partie de la macro-région R3. Cela implique que l'arête R2-R4 est dépendante de R2-R3, elles sont donc fusionnées en même temps.

- Deux macro-arêtes doivent être actualisées:

  R1-R2, R1-R4 $\rightarrow$ 95

  R3-R5, R4-R5 $\rightarrow$ 50

Le contenu du graphe et le statut d'arêtes après cette fusion apparaît sur le tableau 7.5.

La longueur de la frontière de la partition actuelle est la précédente moins la longueur des arêtes éliminées (R2-R3 et R2-R4), c'est-à-dire, 14 - 3 -1 = 10. D'autres fusions sont nécessaires pour réduire ce chiffre à 9 (longueur souhaitée).

L'arête R2-R3 sort de la FAH L'arête R2-R3 sort de la FA de priorité 30, mais elle a déjà été fusionnée. La sortie de cet exemplaire d'arête ne déclenche aucune opération.

NŒUDS (ÉVALUATION)	R1 (255)	R2 (160)	R3 (160)	R4 (160)	R5 (210)
ARÊTES	R1-R2 (95) [REPR]	R2-R1 (95) [REPR]	R3-R2 (27) [FUS]	R4-R1 (102) [N-A]	R5-R1 (45) [REPR]
ASSOCIÉES	R1-R4 (102) [N-A]	R2-R3 (27) [FUS]	R3-R4 (10) [FUS]	R4-R2 (20) [FUS]	R5-R3 (50) [REPR]
(ÉVALUATION)	R1-R5 (45) [REPR]	R2-R4 (20) [FUS]	R3-R5 (50) [REPR]	R4-R3 (10) [FUS]	R5-R4 (50) [N-A]
[STATUT]				R4-R5 (50) [N-A]	

Table 7.5: Contenu du graphe après la deuxième fusion R2-R3

L'arête R1-R5 sort de la FAH L'arête R1-R5 sort de la file de priorité 45, qui est aussi sa valuation actuelle. La macro-arête est une arête simple, alors elle ne provoque pas la disparition d'autres arêtes. La nouvelle macro-région a un niveau de gris moyen de 232. Une seule macro-arête reste en vigueur. Cette macro-arête est l'union de R1-R2, R1-R4, R4-R5 et R3-R5. Son arête représentante est R1-R2 et son évaluation est de 72. Le nouveau contenu du graphe apparaît sur le tableau 7.6.

NŒUDS (ÉVALUATION)	R1 (232)	R2 (160)	R3 (160)	R4 (160)	R5 (232)
ARÊTES	R1-R2 (72) [REPR]	R2-R1 (72) [REPR]	R3-R2 (27) [FUS]	R4-R1 (102) [N-A]	R5-R1 (45) [FUS]
ASSOCIÉES	R1-R4 (102) [N-A]	R2-R3 (27) [FUS]	R3-R4 (10) [FUS]	R4-R2 (20) [FUS]	R5-R3 (50) [N-A]
(ÉVALUATION)	R1-R5 (45) [FUS]	R2-R4 (20) [FUS]	R3-R5 (50) [N-A]	R4-R3 (10) [FUS]	R5-R4 (50) [N-A]
[STATUT]				R4-R5 (50) [N-A]	

Table 7.6: Contenu du graphe après la troisième fusion R1-R5

La longueur de la frontière de la partition actuelle est la précédente moins la longueur des arêtes éliminées (R1-R5), c'est-à-dire, 10 - 2 = 8. La longueur actuelle est alors inférieure à la longueur désirée ($8 \not> 9$), ce qui arrête le processus de fusion.

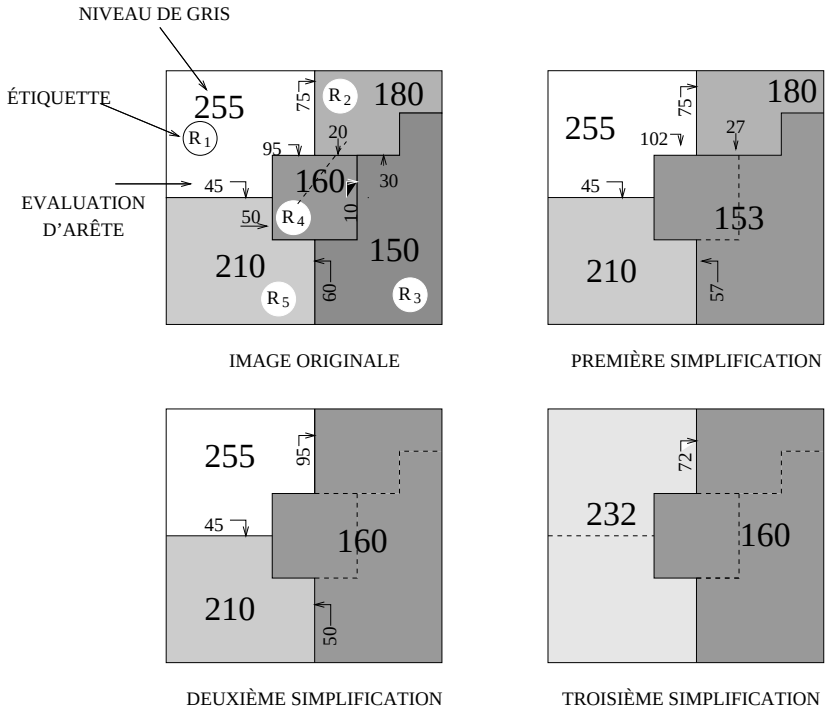


Figure 7.4: Exemple de fusions successives.

Chapter 8

Segmentation 2D par fusion de régions

Dans le chapitre précédent nous avons présenté un algorithme de segmentation basé sur les fusions successives de couples de régions adjacentes. Nous avons pris comme critère de fusion la différence de niveau de gris moyen entre deux régions et nous l'avons appliqué à un exemple simple, en suivant pas à pas l'évolution de l'algorithme. Maintenant nous allons expérimenter l'algorithme avec des images réelles. Nous allons aussi développer d'autres critères de fusion pour aboutir à une segmentation adaptée à une application de codage.

8.1 Application directe de l'algorithme de fusion à une image réelle

L'algorithme de fusion de régions fait cristalliser les structures existantes sur une image. De manière spontanée, le couple de régions qui se ressemble le plus fusionne, définissant ainsi une série de partitions avec un niveau de précision décroissant. Un critère de complexité ou de qualité sur la partition finale définit le critère d'arrêt des simplifications successives.

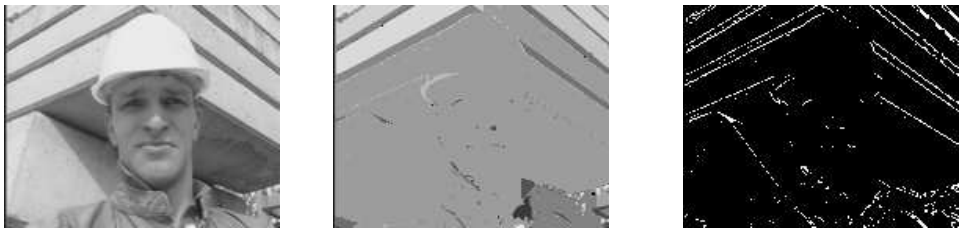
Expérimentons la méthode sur une image réelle. Prenons comme critère de ressemblance entre deux régions la différence de niveau de gris moyen et comme critère d'arrêt 5000 points de contour sur la partition finale et comparons les performances de l'algorithme sans et avec re-actualisation des évaluations des frontières après chaque fusion.

8.1.1 Fusion de régions sans actualisation

Considérons que la priorité de fusion d'un couple de régions est donnée par la différence de leurs niveaux de gris initiaux, calculée sur l'image originale. Fusionnons jusqu'à atteindre le critère d'arrêt sans modifier ces priorités initiales. Cela implique qu'un contour (qui, après quelques fusions, comprend un ou plusieurs contours initiaux) est éliminé dès que sa composante d'évaluation minimale est traitée.

Voyons sur la figure 8.1 le résultat de l'application de l'algorithme de fusion

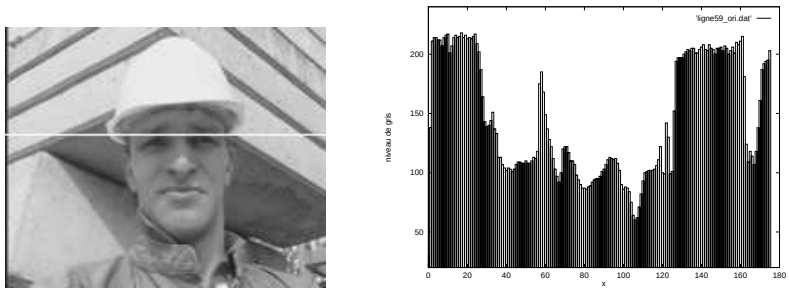
de régions sans actualisation, à l'image déjà bien connue de la figure 8.1(a). La figure 8.1(b) montre la segmentation obtenue avec cette méthode. Malgré le nombre élevé de régions (1262), des régions pertinentes pour la vision humaine ont été fusionnées. En même temps, parmi les 1262 régions, 1080 sont des pixels isolés (voir fig. 8.1(c)). Ces pixels ne sont pas importants pour l'interprétation de l'image et pourtant ce sont ceux qui résistent après une forte simplification. Analysons les causes des défauts de cette segmentation.



(a) Image originale (b) Mosaïque avec 1262 régions et 5000 points de contour. (c) Pixels isolés de (b).

Figure 8.1: Fusion de régions sans actualisation.

Pour illustrer le problème nous allons étudier plus en détail une image réelle. Prenons comme exemple la ligne 59 de l'image 8.2(a). La fonction des niveaux de gris de cette ligne apparaît sur la figure 8.2(b). Nous reconnaissons sur ce profil les différentes régions de l'image que cette ligne traverse: d'abord elle traverse une bande blanche du bâtiment avec un niveau de gris autour de 210, après une bande plus étroite avec un niveau de gris de 140, puis une zone plus sombre du bâtiment avec un niveau de gris compris entre 100 et 110. Le pic blanc aux alentours du pixel 60 correspond au bas du casque. Ensuite, cette ligne traverse la figure de foreman à la hauteur des sourcils (les deux vallées sombres). Entre les pixels 125 et 160 s'étend une autre bande blanche du bâtiment, séparée par une bande étroite et sombre d'une autre bande blanche.



(a) Image Originale (b) Niveaux de gris de la ligne 59 sur l'image originale (a)

Figure 8.2: Profil d'une ligne extraite de l'image originale.

Analysons ce profil de plus près:

1. Nous observons des petites oscillations du niveau de gris sur les bandes blanches du bâtiment (visuellement bien homogènes) qui correspondent

au bruit de l'image. Aux alentours des pixels 35 et 120 le bruit est un peu plus important.

2. Etudions les caractéristiques du sourcil gauche (structure qui s'étend entre les pixels 70 et 90) sur la figure 8.2(b). Sur la coupe transversale, le sourcil gauche apparaît comme une vallée constituée de marches de petite amplitude, autrement dit, constituée de transitions locales douces.

3. Un autre trait à analyser est la transition aux alentours du pixel d'abscisse 126. Le pixel 126, avec un niveau de gris de 152, est entouré de pixels dont le niveau de gris vaut respectivement 101 et 194. Il s'agit une transition abrupte, le cas opposé au point précédent.

Rappelons maintenant les caractéristiques de l'algorithme:

1. Sur l'image originale on calcule la différence de niveaux de gris des régions adjacentes.

2. On fusionne les deux régions de niveau de gris le plus proche jusqu'à atteindre un certain critère d'arrêt.

Avec ces prémisses, le comportement de l'algorithme par rapport aux traits de l'image soulignés précédemment est le suivant:

1. Le bruit, qui se manifeste sous forme de pics contrastés, ne fusionne avec ses voisins qu'aux étapes finales de la simplification.

2. En présence de transitions locales douces, certains traits pertinents risquent de disparaître lors des fusions de proche en proche de régions non contrastées. Par exemple, les marches de petite amplitude aux alentours du pixel 80 (qui correspondent au sourcil gauche) fusionnent dans les étapes préliminaires de simplification, ce qui fait disparaître le sourcil de la segmentation.

3. Les transitions fortes contiennent des marches d'escalier de grande amplitude. Ces marches d'escalier ne représentent pas des traits caractéristiques de l'image, mais, étant donné qu'il s'agit de régions très contrastées, l'algorithme a une tendance intrinsèque à les préserver. C'est le cas du pixel 126.

Finalement, une méthode qui semble cohérente pour une simplification d'image (fusionner le couple de régions qui se ressemble le plus) aboutit à des résultats non satisfaisants. Les grandes régions importantes sont fusionnées tandis que de petites régions insignifiantes restent isolées. En effet, les grandes régions sont souvent reliées par des chemins constitués de marches de petite amplitude tandis que les petites régions, correspondant soit à du bruit soit à des zones de transition abruptes, sont très contrastées.

Le fait que les évaluations initiales ne sont pas actualisées après chaque itération de l'algorithme, explique en partie ce résultat. Dans une étape intermédiaire, une frontière est l'union d'un ou de plusieurs contours initiaux. Si les évaluations ne sont pas mises à jour, c'est la portion de contour initial de plus faible évaluation qui commandera les fusions ultérieures. L'importance visuelle d'une frontière est donc mesurée en lui donnant la valuation de son maillon le plus faible et non pas par une mesure globale sur toute sa longueur, ce qui ne correspond pas aux critères visuels.

8.1.2 Fusion de régions avec actualisation

Examinons maintenant le résultat de l'algorithme quand on calcule la nouvelle valeur moyenne d'une région (à la suite d'une fusion) et qu'on re-évalue ses frontières en conséquence. La figure 8.3(b) montre la partition obtenue à partir de l'image 8.3(a). Cette fois ci, avec le même nombre de points de contour, nous avons seulement 550 régions (à la place de 1262).

(a) Image originale

(b) Mosaïque avec 550 régions et 5000 points de contour.

(c) Pixels isolés de (b).

Figure 8.3: Fusion de régions avec actualisation après chaque fusion.

Si nous comparons les images 8.1(b) et 8.3(b) nous observons que, pour le même nombre de points de contour, l'algorithme qui re-actualise les évaluations après chaque fusion donne une qualité de représentation nettement meilleure. La raison est que l'importance visuelle d'une frontière est calculée de manière globale sur toute sa longueur ce qui:

1. peut réduire l'effet du bruit. Si le bruit se manifeste sous forme des pics alternant autour d'une valeur moyenne, le calcul de la moyenne sur les régions, tenant compte des fusions déjà réalisées, réduit son effet.

2. évite une sous-estimation de l'importance d'une frontière à cause d'une transition locale douce.

3. n'évite pas les marches contrastées sur les zones de transition fortes. Effectivement, parmi les 550 régions obtenues, 422 (76 % des régions) sont des pixels isolés. La figure 8.3(c) nous montre où sont placées ces régions. Puisque l'image n'est pas très bruitée les pixels isolés se placent majoritairement sur les zones des fortes transitions.

Nous constatons donc, que le bruit et les zones de transition sont les régions qui ont la plus forte chance d'être contrastées par rapport à leurs voisines (voir fig. 8.4) et en conséquence sont celles qui restent après une simplification importante de la partition. Puisque ces régions ne sont pas pertinentes pour l'interprétation de l'image, un prétraitement qui les élimine s'avère nécessaire.

8.2 Prétraitement

L'application de l'algorithme de fusion de régions à une image réelle donne une partition avec un pourcentage élevé de régions d'un seul pixel. Ces régions ne correspondent pas aux détails importants de l'image mais aux pics de bruit et

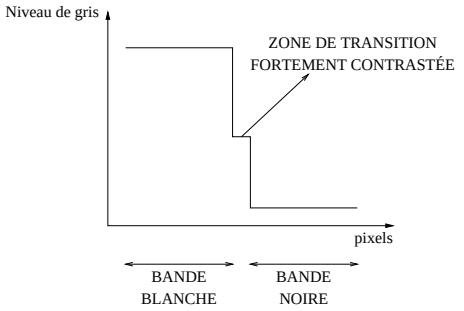


Figure 8.4: Problème des zones de transition.

aux régions des transition fortes. Leur présence augmente le coût de codage sans pour cela améliorer la qualité de la représentation. C'est pourquoi nous cherchons à nous en débarrasser. Pour cela nous allons appliquer un prétraitement à l'image, de sorte à enlever les caractéristiques non perceptibles pour la vision humaine que l'algorithme a tendance à préserver.

Le but du prétraitement est d'éliminer les traits de l'image qui perturbent le bon fonctionnement de l'algorithme, afin de générer une image simplifiée qui respecte les détails visibles. Les caractéristiques qu'on cherche à éliminer sont:

- les pics de bruit
- les zones de transition

parce que ce sont des traits très contrastés mais non significatifs de l'image mais très contrastés.

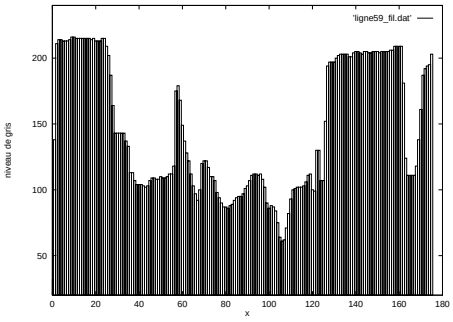
8.2.1 Élimination du bruit

Pour ce prétraitement nous utilisons les filtres connexes, qui conservent la forme des objets qui résistent au filtrage. Plus précisément nous utilisons les filtres aréolaires. Le filtre aréolaire alterné, c'est-à-dire, une ouverture aréolaire suivie d'une fermeture aréolaire, fait que tous les extrema de l'image (maxima et minima) aient une surface égale ou supérieure à un certain seuil, la taille du filtre.

Nous utilisons un filtre aréolaire de taille 10 pour une image QCIF (176×144 pixels). Le nombre de zones plates diminue (les zones plates s'élargissent) tandis que l'aspect visuel de l'image reste très proche de l'original. Certes, un filtre implique une perte de résolution: quelques petits détails peuvent disparaître. Néanmoins cette perte de résolution est beaucoup moins importante que celle associée au gradient (voir section 4.1). Le gradient fait disparaître systématiquement toute structure dont une des dimensions est inférieure à trois pixels. Maintenant cette contrainte est moins sévère; les structures fines sont préservées si elles sont longues.

Voyons l'effet du filtre aréolaire sur la ligne 59 (voir fig. 8.5). Ce filtre efface majoritairement le bruit, qui se manifeste sous forme de pics. Par contre, les filtres aréolaires n'agissent pas sur les zones de transition; les marches pour monter d'un plateau à un autre ne sont pas modifiées, même si elles sont de petite taille. C'est pourquoi après un filtrage aréolaire nous devons nous occuper d'éliminer les escaliers entre deux plateaux contrastés. Autrement ils risquent

d'apparaître sur la partition finale provoquant une augmentation du coût de codification sans apporter une amélioration à la qualité de représentation.



Niveaux de gris de la ligne 59 sur l'image après un filtrage aréolaire de taille 10

Figure 8.5: Effet du filtre aréolaire sur la ligne 59.

8.2.2 Élimination des régions de transition

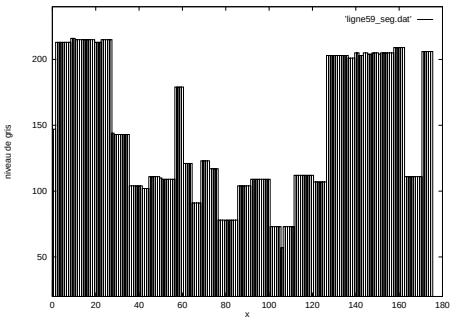
Les zones de transition sont des régions qui faussent l'évaluation de ressemblance entre régions importantes. Soit elles fournissent un chemin de transitions douces entre régions contrastées, soit elles restent isolées à cause d'une transition abrupte.

Le filtre aréolaire ne modifie pas les zones de transition mais nous aide à les repérer. Comme le montre [8] les extrema d'une image sont des régions avec une importante signification visuelle. Le filtre aréolaire assure une surface minimale des extrema. Ainsi, toute région qui à la sortie du filtre a une taille inférieure à celle du filtre peut être considérée comme une zone de transition entre deux plateaux extrémaux. Néanmoins certaines régions de transition peuvent être significatives; il n'est pas interdit à une image de comporter des marches d'escalier réelles et significatives. C'est pourquoi nous avons considéré comme zone de transition (non significative) toute région de taille inférieure à la moitié de la taille du filtre aréolaire. De cette façon la partition obtenue (sans zones de transition) préserve les détails visibles et ne comporte plus de pixels isolés.

Une fois les zones de transition repérées elles sont éliminées en utilisant l'algorithme de croissance de régions: les régions de transition non significatives sont assignées à une de leurs voisines suivant un critère de contraste jusqu'à ce que toute région appartienne à une région considérée significative au niveau de l'étape de prétraitement.

Le profil des niveaux de gris de la ligne 59 prétraitée apparaît sur la figure 8.6. Nous observons que les zones de transition abruptes ont été éliminées et les transitions douces remplacées par une transition plus représentative du contraste.

Illustrons cette procédure avec un exemple. Prenons l'image de la figure 8.7(a). Elle est au format QCIF (176×144 pixels) et contient 19342 zones plates. Si on lui applique un filtre aréolaire de taille 10 l'aspect de l'image ne change pas et le nombre de zones plates est réduit à 11171 (voir fig. 8.7(b)). Parmi ces 11171 régions, seulement 735 ont une taille égale ou supérieure à



Niveaux de gris de la ligne 59 sur l'image sur-segmentée après prétraitement.

Figure 8.6: Effet du prétraitement sur la ligne 59.

la moitié de la taille du filtre. Elles sont considérées comme marqueurs dans une procédure de croissance de régions et les régions restantes sont assignées au marqueur voisin de niveau de gris le plus proche. La figure 8.7(c) montre les marqueurs choisis et la figure 8.7(d) le résultat de la croissance des régions.

Cette procédure a fortement réduit (d'un facteur 26) le nombre de régions de l'image, tout en préservant ses traits caractéristiques. Le résultat est une image avec 735 régions qui:

- simplifie la recherche des zones homogènes.
- accélère le processus de fusion. Au lieu d'avoir 82474 frontières sur l'image originale (51114 sur l'image après l'application du filtre aréolaire de taille 10) qu'il faut évaluer, ordonner et gérer après chaque fusion, il en reste seulement 4060.

8.2.3 Lissage des contours

L'élimination des régions de transition, décrite dans la section 8.2.2, se réalise en utilisant l'algorithme de croissance de régions qui comprend deux étapes:

- une étape de sélection des régions significatives, les marqueurs M.
- une étape d'assignation des régions non significatives aux marqueurs. Les marqueurs grandissent région à région selon un critère de connexité (à tout instant un marqueur est une composante connexe) et de similitude.

Le critère de similitude que nous avons utilisé pour guider la croissance des marqueurs est le contraste. Ce critère produit des contours irréguliers qui se codent plus difficilement que les contours lisses. Voyons l'exemple de la figure 8.9. L'image 8.9(a) a deux régions importantes (M_1 et M_2) et une région de transition (R). Le contraste $R-M_1$ est de $C_1 = 51$ tandis que $R-M_2$ est de $C_2 = 49$. Si on s'intéresse seulement au contraste, la région de transition sera assignée au marqueur M_2 (fig. 8.9(b)), parce que le contraste est plus petit. Pourtant, cette solution produit un contour irrégulier qui est plus difficile à coder. Puisque la différence entre C_1 et C_2 est petite et que nous cherchons à générer des contours réguliers (qui sont codés plus facilement), il serait souhaitable d'assigner la région de transition à M_1 (fig. 8.9(c)).

(a) Image Originale (b) Image filtrée

(c) Zones plates de taille supérieure à *taille_du_filtre / 2* (d) Image sur-segmentée

Figure 8.7: Prétraitement

Dans [39], Pardàs et Salembier proposent une solution qui régularise le contour. Au lieu d'utiliser le contraste comme critère de similitude ils utilisent une somme pondérée du contraste et de la complexité de contour. La similitude entre un pixel p et un marqueur M_i est donné par la formule suivante:

$$similitude(p, M_i) = \alpha \times contraste(p, M_i) + (1-\alpha) \times complexité\_de\_contour(p, M_i) \quad (8.1)$$

où la complexité du contour est donnée par le nombre de pixels voisins de p qui n'appartiennent pas au marqueur candidat M_i. Remarquons que plus le pixel et la région sont similaires, plus la similitude est proche de zéro. α est un paramètre qui permet de donner plus ou moins d'importance au contour par rapport au contraste. Plus α est petit, plus les contours sont lisses, mais moins le contraste est important, ce qui peut mener à une perte significative de la qualité de la segmentation.

Nous avons utilisé une variante de cette formule, adaptée à notre système. Le critère que nous avons utilisé se différencie de celui de [39] sur les deux points suivants:

- Le premier, déjà cité, est que nous travaillons à l'échelle des zones plates et non pas du pixel.
- Le deuxième concerne l'estimation de la complexité du contour. Nous n'utilisons pas la définition donnée précédemment mais nous nous basons

plutôt sur l'augmentation de contour du marqueur due à la fusion de R avec M_i. Nous entendons par augmentation de contour, la différence entre le contour extérieur de R (nombre de pixels voisins de R n'appartenant pas à M_i) et le contour commun à R et M_i (nombre de voisins de R appartenant à M_i). L'avantage de cette nouvelle définition est illustré par la figure 8.8. Deux régions R et R' dans le voisinage d'un marqueur ont la même longueur de contour extérieur, c'est-à-dire, la même complexité de contour selon la formule 8.1. Néanmoins R a une frontière commune avec M_i plus longue que R'. L'augmentation de contour du marqueur nous permet de favoriser la fusion des régions qui se rajoutent au marqueur de manière longitudinale à sa frontière (comme $R - M_i$) par rapport à celles qui se rajoutent de manière transversale (comme $R' - M_i$).

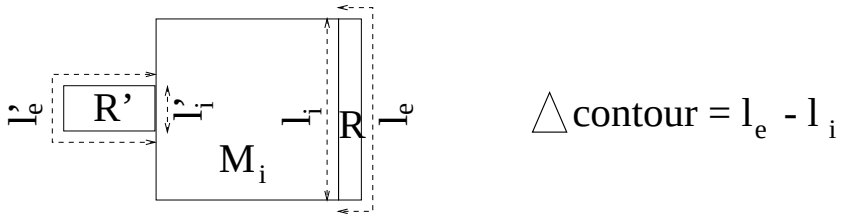


Figure 8.8: Augmentation de contour.

Donc la formule qui donne la similitude entre une région (R) et un marqueur (M_i) devient:

$$similitude(R, M_i) = \alpha \times contraste(R, M_i) \; + \; (1 - \alpha) \times \Delta \; contour(R, M_i)$$

Une fois les marqueurs repérés, leur croissance s'effectue en deux étapes:

- on détecte les régions voisines des marqueurs, qui sont candidates à être absorbées par un marqueur et on les ordonne selon le critère de similitude qu'on vient de présenter.
- le couple marqueur-non marqueur le plus similaire est fusionné, c'est-à-dire le non marqueur est assigné au marqueur, devenant ainsi une partie du marqueur et en conséquence capable d'absorber ses régions voisines.

Nous utilisons la somme pondérée de contraste et augmentation de la complexité de contour pour classer les régions non marqueurs, mais nous affinons la décision d'assignation quand une région a plusieurs marqueurs dans son voisinage. Quand plusieurs marqueurs sont en concurrence pour absorber une région, le contour final de la segmentation (celui qui sépare deux marqueurs) est en jeu. Dans ces situations nous appliquons une idée utilisée dans les algorithmes de codage, but final de notre segmentation. Les algorithmes de codage ne corrigent que les erreurs de prédiction supérieures à un certain seuil (en l'occurrence 5 niveaux de gris). Supposons donc qu'une région de transition R ait plusieurs marqueurs dans son voisinage. Soit M_1 le marqueur dont le contraste C_1 avec R est le plus petit (par rapport aux autres marqueurs), et soit L_1 la longueur de leur frontière commune $R - M_1$. S'il existe un deuxième marqueur M_2 dans le voisinage de R, tel que $|C_1 - C_2| < 5$ et que la frontière $R - M_2$ est plus

longue que L_1, alors R sera assignée au marqueur M_2. Sinon, nous considérons que lisser le contour pénalise de manière importante le contraste et la région R est assignée au marqueur M_1.

Par exemple, sur l'image de la figure 8.9(a) le contraste de R avec M_1 est de 49 et la longueur de sa frontière 3. Avec M_2 le contraste est 51 et la longueur 1. Puisque $|51 - 49| = 2$ est inférieur à 5, R est assignée au marqueur dont la frontière avec R est la plus longue, c'est-à-dire M_1, générant ainsi un contour lisse. Nous avons utilisé un seuil fixe de 5, ce qui donne des bons résultats pour les images testées. Dans le cas d'images très contrastées ou, au contraire, très peu contrastées, ce seuil devra être adapté en conséquence.

Cette solution rend les contours plus lisses sans trop pénaliser l'influence du contraste.

En utilisant cette stratégie nous avons obtenu la sur-segmentation de la figure 8.10(b) qu'on peut comparer à celle de la figure 8.10(a) obtenue avec un critère de contraste. Le nombre de points de contour est de 11862 (voir fig. 8.10(d)) au lieu de 13593 (voir fig. 8.10(c)).

(a) Image originale (b) Croissance de régions: contraste. (c) Croissance de régions: contraste + régularisation du contour.

Figure 8.9: Croissance de régions sans et avec régularisation du contour.

8.3 Critères de fusion

L'application directe de l'algorithme de fusion de régions à une image originale donne des résultats qui font ressortir des traits contrastés mais non pertinents, comme le bruit et les zones de transition abruptes. Nous avons développé une étape de prétraitement qui élimine les éléments de l'image qui font échouer l'algorithme. Le résultat de ce prétraitement est une sursegmentation que nous allons simplifier dans la suite avec l'algorithme de fusion de régions.

Notre objectif est maintenant de développer des critères de ressemblance entre régions qui fournissent des segmentations adaptées à une application de codage.

8.3.1 Critère de contraste

Ce critère consiste à évaluer les frontières avec la différence des niveaux de gris moyens des régions adjacentes. C'est le critère le plus simple que nous pouvons imaginer. Voyons déjà les résultats auxquels on aboutit.

(a) Croissance de régions

(b) Croissance de régions avec lissage de contours

(c) 13593 points de contour

(d) 11862 points de contour

Figure 8.10: Régularité de contour.

Comme dans la section 8.1, nous nous proposons d'obtenir une segmentation avec 5000 points de contour. L'algorithme utilisé dans 8.1.2 est appliqué maintenant à l'image qui résulte du prétraitement (fig. 8.10(b)) et non pas à l'image originale. Le résultat obtenu (voir fig 8.11(a)) comporte 101 régions. Si on le compare avec le résultat précédent (voir fig. 8.3(b)) on constate une augmentation de la qualité de représentation pour une complexité équivalente (le même nombre de points de contour).

Le contraste est bien un attribut qui exprime l'importance visuelle d'une région. C'est pourquoi les régions que nous avons obtenues sont pertinentes. Néanmoins, cette méthode génère parfois des faux contours, c'est-à-dire des contours qui ne correspondent à aucune frontière visible. Ceci arrive en présence d'un changement de luminosité progressif, d'un éclairage non uniforme. Voilà ce qui arrive: plusieurs germes de cristallisation apparaissent à l'intérieur de la même région, c'est-à-dire, plusieurs groupements de régions se forment à différents endroits de la région. Quand ces germes grandissent, leurs niveaux de gris moyens s'écartent à cause du changement de luminosité, ce qui génère un contraste entre eux qui ne correspond pas à ce qu'on perçoit.

Ce défaut nous pousse à essayer un autre critère de fusion basé, cette fois-ci non sur la différence de niveaux des gris moyens, mais sur un contraste local.

(a) Mosaïque avec 101 régions.

(b) Contours de l'image (a).

Figure 8.11: Segmentation obtenue en utilisant comme critère de fusion le contraste.

(a) Mosaïque avec 99 régions.

(b) Contours de l'image (a).

Figure 8.12: Segmentation obtenue en utilisant comme critère de fusion le contraste local.

8.3.2 Critère de contraste local

Après les résultats obtenus avec le critère de contraste, donné par la différence des niveaux de gris moyens des régions, nous nous demandons si le contraste local n'est pas une mesure d'importance perceptive plus pertinente. Le contraste local éviterait de générer de faux contours et garderait les frontières qui correspondent aux contours visibles.

Nous calculons le contraste local de la manière suivante: pour chaque paire de régions (R_1, R_2) nous calculons les niveaux de gris moyens de R_1 et de R_2 sur une bande (d'épaisseur e) qui longe la frontière de part et d'autre, obtenant les valeurs N_1 et N_2 respectivement. Le contraste local est donné par la différence (en valeur absolue) de N_1 et N_2 (voir fig. 8.13).

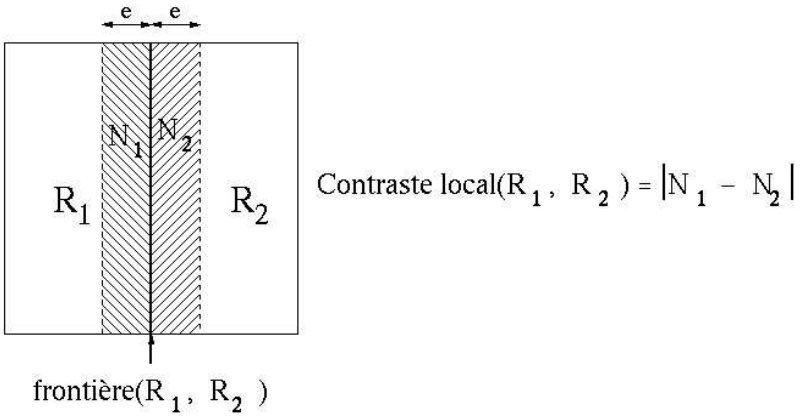


Figure 8.13: Calcul de contraste local.

Après chaque fusion, l'évaluation des frontières autour de la nouvelle région doit être mise à jour avec le nouveau contraste local tout au long de la frontière. Cette nouvelle évaluation est donné par la formule suivante:

$$nouvelle\ évaluation = \left| \frac{\sum_i N_{1i}\ l_i}{\sum_i l_i} - \frac{\sum_i N_{2i}\ l_i}{\sum_i l_i} \right| \tag{8.2}$$

où i correspond à chacune des composantes de la nouvelle frontière, l_i la longueur initiale de la composante i, et N_{1i} et N_{2i} les niveaux de gris moyens de chaque côté de la frontière.

Pour des raisons de simplicité, nous avons utilisé comme formule de réévaluation:

$$nouvelle\ évaluation = \frac{\sum_i v_i l_i}{\sum_i l_i} \tag{8.3}$$

où v_i est la valuation initiale de la composante i, ce qui nous permet d'avoir une seule valuation par arête (v_i), au lieu de deux $(N_{1i}\ et\ N_{2i})$. L'évaluation obtenue avec 8.3 est égale à celle obtenue avec 8.2 quand $N_{1i} \geq N_{2i} \forall i$ ou quand $N_{1i} \leq N_{2i} \forall i$, ce qui en pratique est souvent vrai. Cet algorithme est appliqué à l'image qui résulte du prétraitement avec l'objectif d'obtenir une segmentation de 5000 points de contour. Le résultat est sur la figure 8.12. Cette segmentation a 5000 points de contour comme prévu (voir fig. 8.12(b)) et 99 régions.

8.3.3 Discussion

Le critère de contraste génère de faux contours dans les régions d'éclairage non uniforme. Ceci est évité par le critère de contraste local. Par contre, les décisions locales sont plus sensibles aux éventuelles zones de transition qui n'ont pas été éliminées dans l'étape de prétraitement. Ceci fait que les résultats obtenus avec le contraste local sont différents mais de qualité similaire à ceux obtenus avec la différence de niveau de gris entre régions. Il est difficile de choisir une des deux méthodes; les critères de qualité sont subjectifs. Afin de juger de la pertinence de la segmentation il faudrait idéalement introduire un critère sémantique. Or les séquences que nous traitons peuvent être très variées. C'est pourquoi nous ne disposons pas de formule mathématique permettant de mesurer la qualité de la segmentation.

Nous pouvons faire appel à des critères standard comme la relation signal sur bruit (SNR), mais pour cela le contenu des régions doit être codé. A ce propos, un grand éventail de modèles de texture sont à notre disposition. Si on considère le codage le plus simple, le niveau de gris moyen, et on calcule la relation signal sur bruit des deux images codées, on voit que la méthode qui utilise la totalité de la région pour calculer le contraste donne de meilleurs résultats. Cette conclusion est logique, parce que la méthode essaie de minimiser les différences entre l'image originale et la valeur moyenne sur l'ensemble de la région et non seulement au voisinage de la frontière.

Néanmoins, quand on utilise des méthodes de texture plus évoluées cette relation s'inverse: le contraste local donne de meilleurs résultats.

Nous utiliserons dans la suite les résultats obtenus avec le contraste local parce que:

- le niveau de gris moyen est un modèle de texture très pauvre qui ne donne pas des résultats satisfaisants pour le codage
- pour les expériences que nous avons réalisées, quand on utilise des modèles de texture plus complexes, le contraste local donne des qualités légèrement supérieures

Pour un résultat plus rigoureux il serait nécessaire de faire appel à un groupe d'experts qui, dans des conditions très précises, réaliserait des expériences, demandant l'opinion d'un échantillon représentatif de gens. Une conclusion pourrait être tirée des calculs statistiques sur cet échantillon.

8.3.4 Critère de texture

Les destinataires finaux de nos images codées sont des spectateurs humains. La sémantique joue un rôle important pour évaluer la qualité d'une image: en général il est plus important d'avoir un personnage avec deux yeux, un nez et une bouche que tous les boutons de sa chemise, quoique un capitaine inspectant ses troupes pourrait penser le contraire. Cependant, nos critères de qualité ne peuvent pas tenir compte des informations sémantiques parce que les images peuvent être très variées et nous ne pouvons pas savoir quels sont les éléments indispensables pour une interprétation correcte. C'est pourquoi nous utilisons un critère de qualité standard dans le domaine du traitement du signal: la relation signal sur bruit (SNR).

Nous avons vu que la qualité de la segmentation dépend du modèle de texture utilisé pour remplir le contenu des régions. En particulier si on veut comparer deux segmentations données en leur appliquant un modèle de texture, le classement obtenu dépendra du modèle de texture choisi. Dans notre cas, nous voulons obtenir la segmentation qui optimise la qualité de représentation pour le modèle de texture utilisé par le codeur.

Généralement la segmentation et les algorithmes de codage sont deux étapes indépendantes du système de codage: la segmentation produit une image de sortie (avec ses propres critères) et ensuite les algorithmes de codage sont appliqués au résultat de la segmentation. Cette dictature de la segmentation, avec ses prises de décision sans tenir compte du reste du système, mène à des situations inefficaces: on segmente une région qui était bien rendue sans coder ses contours et on ne segmente pas une région qui fusionnée avec ses voisines n'est pas bien représentée. C'est pourquoi nous allons introduire une boucle de rétroaction entre segmentation et codage: construire la segmentation qui donnera les meilleurs résultats visuels après avoir été codée, de façon à améliorer le rapport qualité / coût du codeur.

Le problème peut être posé pour viser deux objectifs différents:

- maximiser la qualité finale pour un taux de compression donné.
- minimiser le coût de codage pour une qualité donnée.

Dans les deux cas il s'agit de réaliser des fusions, la différence étant le critère d'arrêt. Soit la segmentation est simplifiée jusqu'à ce que sa complexité permette un taux de compression donné, soit jusqu'à ce qu'une fusion supplémentaire provoque une perte de qualité importante.

Il s'agit donc de réaliser des fusions basées sur un critère de texture. Pour cela nous devons trouver une façon d'évaluer la ressemblance de régions adjacentes. Examinons différentes mesures possibles.

8.3.4.1 Mesures de ressemblance des régions adjacentes basées sur la texture

Ressemblance des paramètres de texture

Un premier critère de fusion consisterait à comparer les paramètres de texture des régions adjacentes. Si les paramètres sont proches les régions doivent être bien représentées ensemble et en conséquence la fusion est intéressante. Plus les paramètres se ressemblent, plus la fusion est prioritaire.

Cette évaluation a l'avantage d'être rapide, comparer des paramètres étant une opération simple. Par contre elle n'est pas optimale. Le fait que les paramètres de deux régions soient différents ne veut pas dire qu'il n'existe pas une optimisation globale de bonne qualité sur l'ensemble des deux régions. Par ailleurs nous travaillons avec des modèles de texture qui fournissent des coefficients dépendant de la forme de la région, comme par exemple les polynômes orthogonaux ou la transformée de Fourier (voir [10]). Cela empêche de comparer les paramètres de régions voisines.

Perte de qualité provoquée par la fusion des régions adjacentes

Un autre critère peut utiliser le fait que fusionner deux régions entraîne une perte de qualité de représentation. En effet, quand deux régions fusionnent,

un seul modèle de texture (au lieu d'un modèle par région) doit représenter l'union des deux régions. Le fait de prendre comme évaluation des frontières cette perte de qualité, nous permet de choisir à chaque instant la fusion qui réduit le moins possible la qualité de l'image. Si les fusions se font par ordre de perte croissante, on obtient une segmentation optimisée du point de vue de la qualité de ses régions, pour le modèle de texture utilisé.

Cette méthode comporte néanmoins deux inconvénients:

1. Elle nécessite de modéliser la texture pour chaque couple de régions adjacentes, ainsi que sa mise à jour après chaque fusion. Le temps de calcul nécessaire pour réaliser ceci est prohibitif.

2. Elle est basée sur un critère de qualité relatif, sans faire intervenir la valeur réelle de qualité elle même. De cette façon, si un région subit N fusions, elle peut perdre N fois la perte de qualité permise, ce qui peut la dégrader de manière importante.

   Par ailleurs, ce critère génère des partitions avec des régions de qualité hétérogène. Imaginons, d'une part, une région représentée avec une très bonne qualité, on peut même imaginer une reconstitution parfaite ce qui donne un rapport signal sur bruit infini. Quand on essaie de fusionner une telle région avec une de ses voisines la perte de qualité calculée est très grande (voire infinie), même si la qualité après fusion est suffisamment bonne. Par conséquence l'algorithme aura tendance à garder intactes les régions de très bonne qualité. D'autre part une région de qualité moyenne peut se dégrader et donner une impression de très mauvaise qualité, surtout à côté des régions très bien représentées.

   Ce contraste entre régions de qualités différentes produit un effet très négatif sur l'ensemble de l'image. Tout se passe comme si l'impression de qualité globale n'est pas la qualité moyenne sur l'image mais plutôt qu'elle se rapproche du minimum; ce sont les défaut qui se remarquent le plus. Autrement dit, entre deux images qui globalement ont le même SNR, celle qui a une qualité uniforme donne une impression bien plus agréable qu'une autre avec certaines régions très bien représentées et d'autres moins bien.

La perte de qualité n'est donc pas le critère de ressemblance que l'on cherche.

Qualité des régions adjacentes après fusion

Ce critère évalue l'importance d'une frontière en fonction de la qualité de représentation quand on code ensemble les deux régions adjacentes. Nous réalisons en premier les fusions telles que la qualité de la région obtenue soit la meilleure possible. C'est la loi du plus faible: les régions les mieux représentées perdent leurs privilèges, se rapprochant de la qualité des autres et générant une image de qualité homogène. De cette façon les catastrophes, c'est-à-dire les fusions de régions très mal représentées ensemble, sont évitées.

La qualité après fusion peut être considérée de plusieurs façons. La première solution à laquelle on pense est de calculer le rapport signal sur bruit sur l'union de deux régions fusionnées. Cependant cette estimation présente un problème quand on juge la fusion d'une région grande avec une petite. La contribution au bruit de la petite région par rapport au bruit global est négligeable. De cette

façon elle peut être fortement endommagée tandis que l'estimation de qualité globale est bonne.

Pour éviter ces situations, nous avons préféré estimer la qualité d'une région fusionnée comme le minimum des qualités de chacune d'elles quand on les code avec un seul modèle de texture. C'est-à-dire, si R_1 et R_2 sont deux régions adjacentes, $L(i,j)$ la luminance originale du pixel (i,j), $L'(i,j)$ la luminance du pixel (i,j) codé avec les paramètres de texture de la région $R = R_1 \cup R_2$, la qualité de la région R_1 après fusion est donnée par la formule:

$$Q_1 = 10 \times log_{10} \left(\frac{255^2 \times Aire_1^2}{\Sigma_{(i,j)\epsilon R_1} |L(i,j) - L'(i,j)|^2} \right)$$

Celle de la région R_2 est:

$$Q_2 = 10 \times log_{10} \left(\frac{255^2 \times Aire_2^2}{\Sigma_{(i,j)\epsilon R_2} |L(i,j) - L'(i,j)|^2} \right)$$

et l'évaluation de la qualité de la région fusionnée (évaluation qui exprime l'importance de la frontière: meilleure qualité, frontière moins importante):

$$Qualité \ = \ min(Q_1 \ , \ Q_2)$$

Cette estimation de qualité, calculée de manière indépendante de part et d'autre de la frontière, nous permet de garder un certain nombre des détails qui jouent un rôle important pour l'impression de qualité.

8.3.4.2 algorithme exhaustif

L'algorithme exhaustif qui maximise la qualité finale pour un taux de compression donné serait:

1. évaluer tous les couples de régions adjacentes avec le minimum des ses qualités quand elles sont codées ensemble
2. trouver le couple qui a la plus grande qualité après fusion
3. fusionner ce couple
4. réévaluer les frontières autour de la nouvelle région
5. itérer à partir du point 2 jusqu'à atteindre le critère d'arrêt.

Cette solution qui serait optimale du point de vue de la qualité de représentation finale, est très coûteuse en temps de calcul. Elle nécessite le calcul d'un modèle de texture pour chaque couple de régions adjacentes, ainsi que la mise à jour après chaque fusion. L'implémentation de cet algorithme n'est pas envisageable. Nous sommes obligés de chercher des solutions sous-optimales mais réalisables en pratique.

8.3.4.3 algorithme sous-optimal

Le temps de calcul de l'algorithme exhaustif est très élevée. Une solution sub-optimale (et pratique) est mise en place.

Nous voulons éviter le calcul d'un modèle de texture pour chaque couple de régions adjacentes et en même temps nous voulons faire intervenir dans la

segmentation les capacités de l'algorithme de codage de texture pour rendre le système global plus cohérent et augmenter ainsi le rapport qualité / coût de l'image codée.

L'algorithme sous-optimal que nous avons implémenté consiste à évaluer les frontières avec un critère simple, par exemple le contraste local. Ensuite nous traitons les frontières une à une par ordre croissant d'évaluation. Jusqu'à maintenant traiter une frontière voulait dire l'éliminer. Maintenant avant de réaliser la fusion nous vérifions si elle respecte un certain critère de qualité. Le critère consiste à coder la région et vérifier si après fusion elle préserve ou non une qualité supérieure à un certain seuil. Si le critère de qualité est vérifié, les régions fusionnent. Autrement, elles restent comme deux régions différentes dans la segmentation finale.

Fusionner des régions sachant qu'elles gardent une bonne représentation, augmente l'efficacité du système parce que:

- Seulement un modèle de texture par ensemble de régions fusionnées (au lieu d'un modèle par région initiale) doit être codé.
- Les contours communs aux régions fusionnées, qui auraient dû être codés, disparaissent dans le processus de fusion.
- La qualité de représentation n'est pas affectée de manière significative.

8.3.4.4 Validation de l'algorithme sous-optimal

Dans la section 8.3.4.2, nous avons décrit un algorithme de fusion de régions basé sur un critère de texture qui examine systématiquement toutes les frontières d'une segmentation et élimine à chaque fois la frontière qui préserve la meilleure qualité après fusion. Cet algorithme étant très coûteux en temps de calcul, nous avons présenté, dans la section 8.3.4.3, un algorithme sous-optimal: au lieu d'examiner systématiquement toutes les frontières d'une segmentation, nous les ordonnons selon un critère simple et, avant de les fusionner, nous vérifions si la fusion respecte un critère de qualité basé sur la texture.

Dans cette section nous allons comparer les résultats des deux approches, dans le but de valider l'approche sous-optimale. Prenons la segmentation de la figure 8.14(a), qui comporte 99 régions, et un modèle de texture, par exemple les polynômes orthogonaux d'ordre 4. Fusionnons toutes les régions qui codées ensemble préservent une qualité de 29 dB. L'algorithme exhaustif produit une segmentation avec 43 régions (voir fig. 8.14(b)) tandis que l'algorithme sous-optimal garde 52 régions (voir fig. 8.14(c)). Nous observons que, pour une qualité donnée, l'algorithme exhaustif fournit une segmentation plus simple. D'un autre point de vue, nous pouvons constater sur le tableau 8.1 que l'algorithme sous-optimal avec un paramètre de qualité de 27 dB fournit une segmentation de complexité équivalente à celle obtenue avec l'algorithme exhaustif en prenant comme paramètre de qualité 29 dB, mais avec un qualité légèrement inférieure (29.46 dB au lieu de 29.94 dB). En effet, puisqu'il examine systématiquement toutes les frontières, pour le même paramètre de qualité il trouve plus de régions à fusionner, ou, pour une segmentation de complexité équivalente il choisit de manière plus pertinente les frontières à éliminer. En contrepartie, le temps de calcul est plus élevé; l'initialisation de l'algorithme exhaustif nécessite le calcul d'un modèle de texture pour chacune des frontières de

la segmentation initiale, puis il calcule des paramètres supplémentaires pour actualiser les frontières après chaque fusion. Par contre, l'algorithme sous-optimal calcule au plus, autant de jeux de paramètres de texture que de frontières initiales. En pratique, le nombre de paramètres calculés par l'algorithme exhaustif est de 4 à 6 fois plus élevé que ceux calculés par l'algorithme sous-optimal. Par conséquent, même si l'algorithme sous-optimal produit un nombre de régions plus élevé, ou une qualité légèrement inférieure, il constitue un bon compromis entre performance et temps de calcul.

Nous avons effectué le même test avec des qualités différentes (voir tableau 8.1) et nous avons obtenu des résultats analogues. C'est pourquoi dans ce qui suit nous allons utiliser l'algorithme sous-optimal.

(a) Segmentation de départ

(b) Résultat de l'algorithme exhaustif.

(c) Résultat de l'algorithme sous-optimal.

(d) Image codée à partir de (b)

(e) Image codée à partir de (c).

Figure 8.14: Comparaison entre l'algorithme exhaustif et l'algorithme sous-optimal.

	Algorithme exhaustif		**Algorithme sous-optimal**	
paramètre de qualité	nombre régions	qualité	nombre régions	qualité
25 dB	25	27.51 dB	40	28.42 dB
27 dB	35	28.68 dB	45	29.46 dB
29 dB	43	29.94 dB	52	29.93 dB

Table 8.1: Comparaison entre l'algorithme exhaustif et l'algorithme sous-optimal

8.3.4.5 Résultats

Analysons le comportement de cet algorithme. Imaginons que nous avons une segmentation que nous voulons coder. Nous remplissons chaque région avec le modèle de texture disponible et nous calculons le rapport signal sur bruit (Q) sur l'ensemble de l'image. Ensuite nous ordonnons les frontières de la segmentation par ordre croissant de leur contraste local et nous les traitons une à une. Pour chaque frontière ($R_1 - R_2$) nous examinons la qualité de R_1 (Q_1) et de R_2 (Q_2) si les régions R_1 et R_2 avaient été codées comme une seule région. Si ces deux qualités (Q_1 *et* Q_2) sont supérieures à Q (qualité calculée sur l'ensemble de l'image) nous fusionnons les deux régions. La segmentation est ainsi simplifiée sans qu'on remarque visuellement une dégradation importante de l'image codée.

Nous avons vu précédemment l'importance de l'actualisation des frontières au fur et à mesure que les fusions se réalisaient. Nous continuons dans la même ligne, actualisant le contraste local des frontières après chaque fusion et aussi tenant compte des "macro-régions" déjà constituées, pour vérifier le critère de qualité avant de réaliser une fusion.

Illustrons cette procédure avec un exemple. Prenons la segmentation obtenue avec le critère de contraste local dans la section 8.3.2 (voir fig. 8.15(a)). Cette segmentation comporte 99 régions et 4966 points de contour. Si on utilise la transformée de Fourier avec 10 coefficients pour coder le contenu de chaque région on obtient l'image codée de la fig. 8.15(b) qui a une qualité de 30.75 dB. Appliquons à cette segmentation l'algorithme que nous venons de décrire: rejoignons toutes les régions qui après fusion préservent une qualité supérieure à la qualité initiale de l'image: 30.75 dB. La segmentation obtenue est sur la fig 8.15(c). Elle comporte 82 régions et 4550 points de contour et l'image correspondante codée (figure 8.15(d)) a une qualité de 30.59 dB. La segmentation comporte environ 10% de moins de régions et de points de contour et néanmoins la qualité visuelle n'a pas été dégradée.

Appliquons le même algorithme en supposant que le codeur de texture utilise des polynômes orthogonaux d'ordre 4 pour représenter l'intérieur de chacune des régions. La simplification obtenue comporte 71 régions et 4379 points de contour et la qualité est passée de 32.15 dB à 31.83 dB (voir figure 8.16).

La qualité globale après la simplification de texture a légèrement diminué. Nous pouvons imaginer d'itérer l'algorithme pour simplifier davantage la segmentation. Quelle est l'évolution de l'algorithme après quelques itérations? Est-ce que l'image se dégrade petit à petit jusqu'à réduire la segmentation à une seule région ou bien converge-t-il vers une segmentation optimale du point de vue du coût de codage pour le codeur utilisé?

Etant donné que le seuil de qualité utilisé est la qualité globale de l'image, la marge de dégradation permise est petite, ce qui fait que le seuil de la deuxième itération est très proche du premier. Ainsi, dans la deuxième itération à peine une ou deux régions fusionnent et dans les itérations suivantes aucune fusion ne vérifie la condition de qualité nécessaire. La convergence est ainsi atteinte en deux itérations et même la plupart du travail de simplification est fait dans la première étape.

Cet algorithme nous permet d'éliminer d'une segmentation les éléments qui, selon le codeur, ne comportent pas une amélioration significative de la qualité visuelle, augmentant ainsi le rapport qualité / coût de l'image codée.

Cet algorithme impose un critère de qualité qui dépend de la qualité de

(a) Mosaïque avec 99 régions 4966 points de contour.

(b) Image codée à partir de (a). 30.75 dB

(c) Mosaïque avec 82 régions 4551 points de contour.

(d) Image codée à partir de (c). 30.59 dB

Figure 8.15: Simplification de texture utilisant la transformée de Fourier.

(a) Mosaïque avec 99 régions 4966 points de contour.

(b) Image codée à partir de (a). 32.15 dB

(c) Mosaïque avec 71 régions 4379 points de contour.

(d) Image codée à partir de (c). 31.83 dB

Figure 8.16: Simplification de texture utilisant les polynômes orthogonaux d'ordre 4.

(a) Mosaïque avec 90 régions 2972.

(b) Image codée à partir de (a). (38.01 dB)

(c) Mosaïque avec 69 régions 2690 points de contour.

(d) Image codée à partir de (c).(37.88 dB)

(e) Mosaïque avec 35 régions 1882 points de contour.

(f) Image codée à partir de (e).(36.02 dB)

Figure 8.17: Simplification de texture utilisant les polynômes orthogonaux d'ordre 4.

l'image qu'il a reçu en entrée. C'est un critère relatif, c'est-à-dire, la valeur absolue de la qualité n'intervient pas. Il est possible que cette qualité soit très élevée ou très basse et dans ce cas là, d'autres actions devraient être entreprises.

Actions à entreprendre si la qualité de l'image codée à partir de la segmentation d'entrée est trop basse

Si la qualité de l'image d'entrée est trop basse, cela voudrait dire que les fusions réalisées avec un critère de contraste sont allées trop loin. Nous devrions revenir en arrière et séparer les dernières fusions réalisées avec un critère de contraste, pour refusionner après avec un critère plus performant comme celui de la texture.

Actions à entreprendre si la qualité de l'image codée à partir de la segmentation d'entrée est trop élevée

Considérons la situation dans laquelle le codage de la segmentation à simplifier produit une image de très bonne qualité. Par exemple, prenons la segmentation de la figure 8.17(a) qui comprend 90 régions et 2972 points de contour. Son codage avec les polynômes orthogonaux d'ordre 4 produit l'image codée de la figure 8.17(b) dont la qualité est de 38.01 dB. Si nous fusionnons toutes les régions qui préservent une qualité supérieure à 38.01 dB, nous obtenons la segmentation de la figure 8.17(c) qui comprend 69 régions et 2690 points de contour. La qualité de représentation de cette simplification codée (fig. 8.17(d)) est de 37.88 dB. Comme nous l'avions déjà montré, cette opération simplifie la segmentation sans pénaliser de manière sensible la qualité visuelle de l'image. Néanmoins, l'heuristique nous a montré que 38 dB est un seuil de qualité élevé. Aux alentours de 30 ou 32 dB, nous obtenons des images de qualité satisfaisante. En conséquence, nous pouvons simplifier davantage la segmentation sans que cela ne se répercute de manière significative sur l'aspect visuel de l'image. Si nous relâchons cette contrainte de qualité et nous considérons qu'avec 32 dB, par exemple, la qualité de représentation est suffisamment bonne, la simplification est beaucoup plus importante (35 régions et 1882 points de contour, fig 8.17(e)). Malgré cette forte simplification de la segmentation, avec la réduction du coût de codage qui en découle, l'image codée résultante (fig. 8.17(f)) garde un aspect très proche de celui des images précédentes (fig 8.17(b) et (d)).

La stratégie de segmentation serait ainsi la suivante:

- nous obtenons avec le critère de contraste une image segmentée dont le coût de codage est supérieur au coût visé et dont la qualité de l'image codée est suffisante.
- l'image segmentée obtenue au point précédent est simplifiée avec le critère de texture. Le paramètre de qualité utilisé dans cet algorithme est a priori la qualité de l'image d'entrée. Si la simplification obtenue avec ce paramètre dépasse encore le coût de codage visé, le paramètre de qualité est réduit progressivement pour simplifier graduellement l'image segmentée jusqu'à atteindre le coût visé. On réduira aussi le paramètre de qualité, s'il est considéré trop élevé.

Pour nos tests, nous avons ajusté les paramètres (nombre final de points de contour et qualité nécessaire) manuellement. Un travail supplémentaire reste à faire pour automatiser le choix des paramètres, en fonction de la complexité de l'image et du taux de compression visé.

8.4 Conclusion

Dans ce chapitre nous avons appliqué l'algorithme de fusion de régions à des images réelles.

Nous avons vu que le fait de fusionner, deux par deux, les régions de niveaux de gris les plus proches, fait disparaître les régions pertinentes tout en préservant beaucoup d'autres qui ne sont pas importantes pour l'interprétation de l'image. En analysant de près le problème, nous avons développé une étape de prétraitement qui élimine les traits perturbant le bon fonctionnement de l'algorithme: le bruit et les zones de transition.

Nous avons illustré l'importance de l'actualisation des évaluations des frontières à la suite de chaque fusion. Il s'agit d'une généralisation de la re-actualisation de la dynamique en plusieurs étapes de sur-segmentation que nous avions décrite dans le chapitre 4. L'actualisation a été ainsi une caractéristique essentielle de tous nos algorithmes.

Ensuite nous avons décrit différents critères de fusion que nous avons combinés pour développer l'algorithme de segmentation. Les premières fusions (après le prétraitement) sont réalisées selon un critère simple: le contraste local. Cette opération nous fournit une segmentation qui:

- excède le coût de codage visé, pour réaliser par la suite la simplification basée sur le critère de texture.
- contient un nombre de régions pas trop élevé pour que le critère de texture (très coûteux en temps de calcul) soit applicable.

La segmentation ainsi obtenue est simplifiée par l'algorithme de fusion selon le critère de texture. Une stratégie sous-optimale a été mise en place pour accélérer le processus. A priori, le paramètre de qualité utilisé est la qualité initiale de l'image. Mais ce paramètre est réduit si le taux de compression n'est pas atteint, ou s'il est considéré trop élevé.

Nous avons introduit un critère de fusion pour adapter la segmentation à un système de codage. Nous nous sommes basés sur la qualité de représentation des régions après fusion, utilisant le modèle de texture disponible par le codeur. Cela donne déjà de bons résultats, mais ce n'est pas encore une stratégie optimale. Pour améliorer davantage les résultats il serait intéressant d'introduire dans le critère de fusion non seulement la qualité de représentation mais aussi le coût de codage. Ainsi, entre deux fusions qui préservent une bonne qualité, nous donnerons priorité à celle qui réduit davantage le coût de codage.

Par ailleurs, nous avons supposé un modèle de texture unique au codeur. Si le codeur dispose de plusieurs algorithmes de codage de texture, comment choisir pour chaque région celui qui donne le meilleur résultat? Encore une fois, la solution optimale serait trop lourde en temps de calcul. Elle consisterait à essayer tous les modèles de texture pour chaque couple de régions voisines.

Pour chaque frontière nous retiendrions le modèle qui donne le meilleur rapport qualité / coût et nous éliminerions les frontières par ordre décroissant de cette valuation.

Comme solution sous-optimale nous proposerions de commencer par fusionner avec le modèle de texture le plus simple, et aussi le moins coûteux en termes de codage. Tant que la qualité obtenue avec un modèle simple est bonne, nous ne faisons pas appel à un modèle plus compliqué. Au fur et à mesure que le nombre de régions diminue, nous appliquons des modèles de texture de plus en plus compliqués. Dans ce cas, le coût de codage de chaque modèle doit être introduit impérativement, parce que l'éventuelle amélioration en qualité qu'un modèle compliqué apporte peut être liée à une forte augmentation du coût de codage. Dans ces situations, nous réaliserons une fusion seulement si le rapport qualité / coût est amélioré.

Chapter 9

Segmentation de séquences d'images par fusion de régions

Ce chapitre est consacré à la segmentation de séquences par fusion de régions. Il est chronologiquement le dernier développé, ce qui lui donne l'avantage de profiter de l'expérience acquise tout au long des chapitres précédents ainsi que d'algorithmes plus performants d'estimation de mouvement et de codage, qui ont vu le jour dans d'autres laboratoires pendant le déroulement de cette thèse.

Résumons les connaissances acquises qui seront maintenant utilisées:

- Pour traiter la dimension temporelle, une approche récursive se révèle être la mieux adaptée à notre problème parce qu'elle impose une stabilité temporelle, n'introduit pas de retard et ses besoins en mémoire sont réduits. La phase d'initialisation de cette approche récursive sera réalisée par l'algorithme bidimensionnel décrit dans le chapitre 8.

- Nous avons vu que l'application directe de l'algorithme de fusion de régions à une image réelle fait ressortir des traits contrastés mais non pertinents comme le bruit et les zones de transition. Une étape de prétraitement a été développée dans le cas bidimensionnel; elle sera adaptée aux séquences dans ce chapitre.

- L'importance de l'actualisation des valuations des frontières après chaque fusion a été illustrée. L'actualisation sera donc une caractéristique essentielle de tous les algorithmes développés dorénavant.

- Nous avons implémenté les algorithmes de fusion de régions sur une structure de graphe. La seule modification nécessaire pour appliquer ces algorithmes à une image tridimensionnelle est d'utiliser un voisinage tridimensionnel pour engendrer le graphe.

- Certains critères de fusion, comme le contraste ou le contraste local, sont aussi valables en trois dimensions. D'autres critères vont être développés pour traiter les particularités des séquences. Par exemple nous allons implémenter un algorithme qui fusionne des régions si elles ont un mouvement cohérent.

- Nous avons vu que l'algorithme de fusion de régions a l'avantage, par rapport à l'approche morphologique classique, de produire une segmentation sans introduire un jeu de marqueurs de manière artificielle; de proche en proche, les régions voisines qui se ressemblent selon un certain critère sont fusionnées jusqu'à ce qu'un critère d'arrêt soit atteint. De cette façon nous évitons la phase de sélection de marqueurs a priori, qui est une opération complexe dans le cas général. Néanmoins dans le cadre du traitement de séquences il est intéressant d'imposer comme marqueurs les régions de la segmentation précédente de manière à obtenir une segmentation stable dans le temps. Nous allons voir comment combiner au mieux l'algorithme de fusion de régions et l'introduction d'une stabilité temporelle.

Nous allons décrire en détail une à une les modifications qui ont été nécessaires pour adapter l'algorithme de segmentation aux séquences.

9.1 Algorithme de fusion de régions: introduction de marqueurs et traitement de nouvelles régions

Les algorithmes de segmentation de séquences doivent trouver un équilibre entre l'imposition d'une stabilité temporelle et la possibilité d'introduire de nouvelles régions de façon à suivre correctement l'évolution de la scène.

La solution adoptée dans [39] réalise le travail en deux étapes: une première, appelée projection, qui comporte la localisation des régions précédentes au temps courant. Dans une deuxième étape, par différence entre l'image projetée et l'image originale on obtient un résidu, dans lequel on récupère des marqueurs de nouvelles régions (régions qui ne sont pas bien représentées dans la projection). Ces marqueurs sont introduits dans la procédure et des nouveaux contours sont rajoutés à la segmentation sans modifier les contours de la projection. La figure 9.1 montre le schéma de cette approche. Le problème de cette approche est qu'au moment de la projection tous les pixels du temps courant doivent être assignés à une région précédente, même les pixels des nouvelles régions. Les contours ainsi obtenus peuvent traverser une nouvelle région, changeant la valeur moyenne des marqueurs qui croissent, et influençant la procédure de croissance. Par ailleurs quand on calcule le résidu de la partition projetée codée on obtient l'information des régions qui n'ont pas été codées, c'est-à-dire des candidates à de nouvelles régions, mais aussi les erreurs de codage propres au codage avec pertes qu'on développe. Cela rend plus complexe la recherche de marqueurs de nouvelles régions dans le résidu que dans l'image originale.

Dans le chapitre 5 nous avons proposé une solution qui évite ces inconvénients. Il s'agit de chercher les marqueurs de nouvelles régions et de les combiner avec les régions de la segmentation précédente de façon à réaliser une seule croissance de marqueurs pour trouver les contours qui nous intéressent. Néanmoins cette solution nécessite une recherche de marqueurs de nouvelles régions a priori, problème qu'on avait évité avec l'algorithme de fusion de régions.

Voyons maintenant comment combiner les atouts des marqueurs qui imposent une stabilité temporelle et ceux des algorithmes de fusion de régions qui permettent l'apparition de régions sans avoir besoin de les marquer.

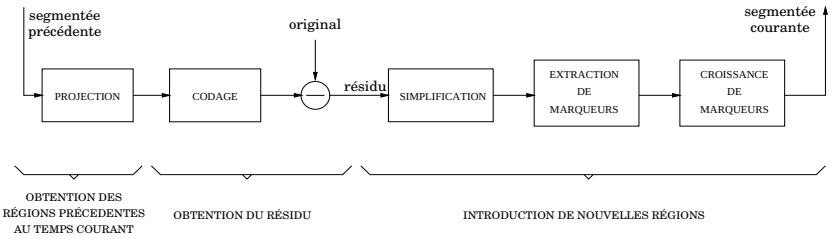


Figure 9.1: Schéma d'introduction de nouvelles régions après projection [39].

L'algorithme de fusion de régions décrit dans le chapitre 7 permet la réunion de n'importe quel couple de régions voisines. Si on applique cet algorithme à la fenêtre glissante de la procédure récursive, deux régions du passé peuvent fusionner. Cependant les régions du temps $t-1$ appartiennent à la segmentation précédente, segmentation qui a déjà été établie et qu'on ne remet pas en cause. Pour éviter l'altération de la segmentation précédente les fusions entre deux régions du passé sont interdites. Ainsi les fusions peuvent se produire entre deux régions du temps t ou entre une région du temps t et une autre du temps $t-1$, mais jamais entre deux régions du temps $t-1$.

Les conséquences de cette procédure sont:

- on retrouve les régions précédentes au temps courant (par fusion des régions $t-1$ avec des régions t) ce qui impose une cohérence temporelle.
- on obtient de nouvelles régions par cristallisation indépendante au temps courant (fusions entre régions du temps t).

Les régions du temps $t-1$ jouent le rôle de marqueurs mais avec une nuance par rapport aux marqueurs de la section 4. Précédemment un marqueur donnait lieu à une région dans la segmentation et seulement les régions marquées apparaissaient dans la segmentation, c'est-à-dire la segmentation avait au plus le même nombre de régions que de marqueurs initialement imposés. Maintenant une région marquée est toujours dans la segmentation finale mais d'autres régions peuvent aussi apparaître. Attention, on parle de segmentation tridimensionnelle. Toutes les régions marquées (les régions de la segmentation précédente) sont dans la segmentation 3D finale, ce qui ne veut pas dire que toutes aient réussi à passer au temps courant. Le tableau 9.1 résume le comportement des deux algorithmes :

	Croissance de marqueurs	Fusion de régions
Marqueur avec Non Marqueur	Fusion Permise	Fusion Permise
Non Marqueur avec Non Marqueur	Fusion Interdite	Fusion Permise
Marqueur avec Marqueur	Fusion Interdite	Fusion Interdite

Table 9.1: Comportement de l'algorithme de croissance de marqueurs par régions et de fusion de régions

L'algorithme de croissance de marqueurs par régions fusionne toujours un non-marqueur avec un marqueur, jusqu'à ce que tous les non marqueurs soient

assignés à un marqueur. L'algorithme de fusion de régions permet la fusion d'un non-marqueur non seulement avec un marqueur, mais aussi avec un autre non-marqueur. Le critère d'arrêt est donné soit par un critère de qualité ou de complexité de segmentation, soit par absence de régions non-marqueurs.

L'utilisation de l'algorithme de fusion de régions pour segmenter la fenêtre glissante de l'approche récursive, prenant comme marqueurs les régions de la segmentation précédente, impose une stabilité temporelle à la segmentation et en même temps permet d'inclure de nouvelles régions de manière spontanée, sans une sélection préalable de marqueurs pour ces régions.

9.2 Prétraitement

L'objectif du prétraitement est d'éliminer les éléments contrastés, comme par exemple le bruit et les zones de transition, que l'algorithme de fusion de régions tend à préserver et qui ne correspondent pas aux caractéristiques pertinentes de l'image.

Dans la section 8.2 nous avons développé une étape de prétraitement pour les images 2D que nous allons adapter au traitement de séquences.

Rappelons l'algorithme décrit en deux dimensions. Il peut être décomposée en deux étapes:

1. filtre aréolaire qui élimine le bruit et en même temps nous aide à repérer les régions de transition.
2. élimination des régions de transition qui, à son tour, est réalisée en deux étapes:
   (a) sélection des régions de transition: toutes les régions qui à la sortie du filtre aréolaire ont une taille inférieure à un certain seuil
   (b) assignation des zones de transition aux régions voisines suivant un critère de contraste et de simplicité de contour. Ceci est réalisé en utilisant l'algorithme de croissance de marqueurs par régions.

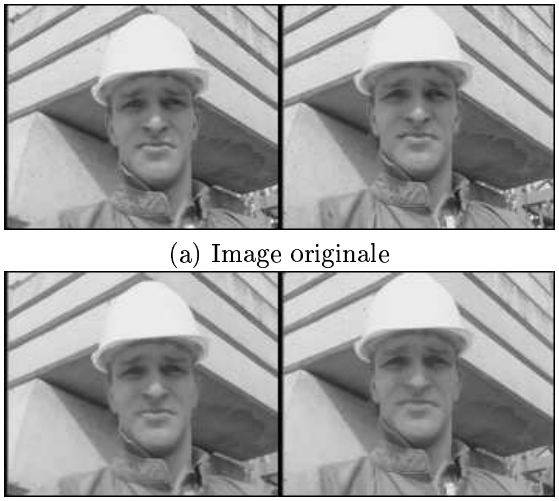
La procédure pour traiter les séquences utilise la même philosophie, mais au lieu de travailler sur une image 2D elle le fait sur une image 3D qui correspond à la fenêtre glissante de l'approche récursive. Elle introduit aussi la compensation de mouvement du plan précédent pour améliorer la correspondance des régions entre les deux plans et ainsi stabiliser davantage la segmentation [38] (voir aussi la section 9.2.2).

Le prétraitement 3D peut se décomposer en les étapes suivantes:

1. filtrage aréolaire
2. compensation de mouvement de l'image précédente
3. élimination des régions de transition

9.2.1 Filtrage aréolaire

Nous utilisons un filtre aréolaire, mais cette fois-ci tridimensionnel. Pour que le filtrage soit équivalent à celui appliqué en deux dimensions nous multiplions la taille du filtre par le nombre des plans de l'image 3D, c'est-à-dire deux (voir figure 9.2).



(a) Image originale

(b) Image filtrée

Figure 9.2: Filtrage aréolaire 3D de taille 20

9.2.2 Utilisation de l'information de mouvement pour améliorer la continuité temporelle

Comme nous l'avons déjà décrit précédemment, la procédure de segmentation que nous utilisons consiste à réunir dans une image tridimensionnelle deux plans consécutifs d'une séquence: le premier plan a déjà été segmenté et nous cherchons l'emplacement de ses objets dans le deuxième plan.

Pour ce faire nous travaillons avec deux images tridimensionnelles: une qui réunit les image originales I_{t-1}, I_t (I) et une autre qui rejoint les images segmentées S_{t-1}, S_t (S). A la fin de la procédure le deuxième plan de S contient le résultat de la segmentation du temps courant.

Nous réunissons deux temps successifs dans une image tridimensionnelle dans le but de transmettre l'information du passé au présent et d'obtenir ainsi une continuité temporelle. La transmission de l'information d'un plan au suivant se réalise à travers les zones de bas contraste temporel, qui correspondent aux zones de recouvrement entre les positions consécutives des objets. Le recouvrement temporel est donc une condition sans laquelle nous ne reconnaissons pas un objet comme existant dans le passé. Les grands objets vérifient sans difficulté cette condition, cependant les objets petits se déconnectent facilement à cause d'un mouvement rapide (par rapport à leur taille). La déconnexion temporelle implique l'absence d'une zone de bas contraste temporel dans l'image I, ce qui empêche la localisation de l'objet du passé dans le plan courant.

Pour éviter cette rupture temporelle nous allons utiliser des algorithmes d'estimation et de compensation de mouvement. Ces algorithmes ont déjà été utilisés dans [18] et [38].

9.2.2.1 Estimation de mouvement

Imaginons que nous avons deux images consécutives I_{t-1} et I_t, et que nous voulons estimer le mouvement de I_t par rapport à I_{t-1}. Nous allons utiliser

l'algorithme d'estimation de mouvement par mise en correspondance de blocs (en anglais "block-matching") en arrière [18]. Cet algorithme divise l'image I_t en petits blocs réguliers et considère que chaque bloc bouge de manière rigide.

L'estimation de mouvement du bloc b_k de l'image I_t, centré sur le point de coordonnés (x, y), consiste à déplacer le bloc b_k sur l'image I_{t-1}, dans une zone de recherche centrée sur le pixel de coordonnés (x, y), et à trouver la position (x', y') qui minimise la différence entre I_{t-1} et b_k. Ainsi le bloc b_k de I_t est supposé venir du bloc $b_{k'}$, centré sur (x', y') de l'image I_{t-1} et l'estimation de mouvement de ce bloc est donc $(x - x', y - y')$. La figure 9.3 illustre cette procédure.

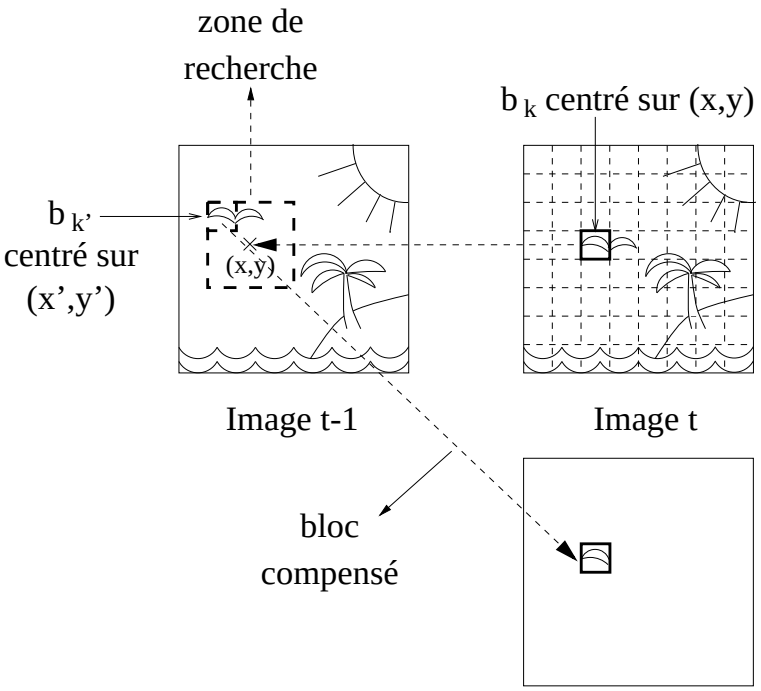


Figure 9.3: Compensation de mouvement par Block-matching en arrière

9.2.2.2 Compensation de mouvement

L'estimation de mouvement nous fournit pour chaque bloc b_k de l'image I_t un vecteur de déplacement qui pointe vers le bloc b'_k de I_{t-1} de façon à minimiser la différence $|b_k - b'_k|^2$.

Si les blocs b_k de I_t sont remplacés par les blocs b'_k correspondants de I_{t-1} nous obtenons une image I'_{t-1} avec l'information du passé mais dont les différences avec I_t dues au mouvement ont été réduites. Autrement dit, l'image I_{t-1} a été compensée en mouvement.

La compensation de mouvement a pour but d'aligner l'image I_{t-1} sur l'image I_t évitant les différences dues au mouvement entre les deux. La transmission d'information entre l'image compensée I'_{t-1} et I_t est plus directe (facile) qu'entre I_{t-1} et I_t. C'est pourquoi I'_{t-1} remplace I_{t-1} dans le premier plan de la fenêtre glissante de l'algorithme récursif. Pour que l'algorithme reste cohérent la même

compensation de mouvement appliqué à I_{t-1} doit être appliqué à S_{t-1} de façon à aligner les marqueurs de l'image précédent avec le nouvel emplacement de l'objet. Nous appellerons l'image résultante S'_{t-1}.

La compensation de mouvement que nous avons utilisée se fait en arrière. Elle cherche dans le passé un bloc qui ressemble au bloc courant. Cette technique a l'avantage, par rapport à la compensation de mouvement en avant (recherche du la nouvelle position d'un bloc du passé), d'éviter les conflits entre blocs compensés; les blocs se compensent les uns à coté des autres sans interférence entre eux, sans conflits de superposition de blocs ni de trous. Cette proposition reste vraie tant qu'on parle de l'image texturée (originale) compensée. Par contre la compensation de la segmentation peut produire plusieurs composantes connexes du même objet, situation qui n'est pas prévue dans les algorithmes de codage. Or, les doubles composantes connexes sont souvent dues au fait qu'on considère un mouvement rigide des blocs, ce qui n'est pas complètement vrai. Le déplacement rigide d'un bloc situé à cheval entre deux régions produit de petites composantes connexes isolées. Ces doubles composantes sont éliminées, gardant la composante connexe la plus grande de chaque région, pour éviter de fausses propagations. Puisque le rôle de cette segmentation compensée est de servir de marqueur du temps courant, le fait que S'_{t-1} ne soit pas une partition n'est pas un problème parce que son objectif est simplement d'assurer la connexion temporelle avec la nouvelle position de l'objet.

Voyons l'effet de la compensation de mouvement sur des images réelles. Prenons deux images successives de la séquence foreman (figures 9.4(a) et 9.4(b)). La différence entre les deux apparaît sur la figure 9.4(c). Nous pouvons quantifier cette différence à partir de la qualité de 9.4(a) comme approximation de 9.4(b) ce qui représente une relation signal sur bruit de 23.05 dB. Par contre, si on compense en mouvement l'image 9.4(a) avec des blocs de taille 8×8 pixels, nous obtenons l'image 9.4(d) dont la qualité comme approximation de 9.4(b) est de 32.68 dB. La figure 9.4(e) montre la différence entre l'image compensée (9.4(d)) et l'image courante (9.4(b)).

De cette façon, si une petite région se déconnecte de sa nouvelle position à cause d'un mouvement rapide, la compensation de mouvement nous permet de la placer en face de sa nouvelle position. Ainsi nous récupérons la connexion temporelle et avec elle la transmission de l'information du passé vers le présent devient possible.

9.2.2.3 Le sous-échantillonnage temporel et le mouvement

Parfois les systèmes de transmission à bas débit ne transmettent qu'une image sur p, ce qui fait gagner directement un facteur p en compression. En réception, pour reproduire la séquence à une vitesse correcte, soit l'image est répétée p fois, soit elle est interpolée.

Le sous-échantillonnage temporel rend plus difficile la stabilité dans la segmentation: les mouvements des p trames s'accumulent et les objets se déconnectent temporellement plus facilement. Pour éviter l'instabilité due au sous-échantillonnage temporel une possibilité est de segmenter toutes les trames, même si on ne transmet qu'une sur p; il est plus facile de suivre les objets trame à trame que de p trames en p trames.

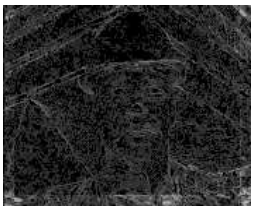
Néanmoins, l'utilisation de la compensation de mouvement nous permet de reconnecter les objets qui par un mouvement rapide n'avaient plus de recouvre-

(a) Image Originale 1

(b) Image Originale 2

(c) Différence entre (a) et (b)

(d) Image 1 compensée en mouvement par rapport à l'image 2



(e) Différence entre (b) et (d)

Figure 9.4: Compensation de mouvement par block-matching en arrière

ment entre leurs positions respectives au temps $t-p$ et au temps t, ce qui nous permet de segmenter avec la fréquence de l'image sous-échantillonnée dans le temps.

Voyons donc les performances de la compensation de mouvement entre deux images non consécutives. Prenons deux images correspondant respectivement aux temps $t-5$ et t (figures 9.5 (a) et 9.5 (b)). La compensation de mouvement de $t-5$ par rapport à t donne le résultat de la figure 9.5(c), dont la différence avec l'image t est sur la figure 9.5(d). La qualité de 9.5(c) comme approximation de 9.5(b) est de 27.36 dB. Cette qualité élevée est due au fait qu'une grande partie de l'image ne bouge presque pas et dans ce cas la compensation est correcte. Mais la compensation dans les zones qui bougent beaucoup est réellement de mauvaise qualité; le mouvement est tellement brusque qu'il est difficile de l'estimer. En fait, on essaye d'estimer le mouvement entre les trames $t-5$ et t, ce qui représente un fort changement, et on ignore toutes les trames intermédiaires qui donnent une continuité au mouvement.

Le résultat peut être amélioré si au lieu d'ignorer les images intermédiaires nous réalisons la compensation en plusieurs étapes: nous compensons l'image I_{t-5} par rapport à I_{t-4} et nous obtenons une image I'_{t-5} , ensuite I'_{t-5} est compensée par rapport à I_{t-3} et nous obtenons I''_{t-5} et ainsi de suite jusqu'à compenser l'image I^{IV}_{t-5} par rapport à I_t. L'image compensée finale est sur la figure 9.5(e) et sa différence avec l'image I_t est sur la figure 9.5(f). La qualité de cette compensation est de 29.0dB et l'impression visuelle est bien supérieure à la compensation directe ignorant les images intermédiaires.

En cas de sous-échantillonnage temporel, la compensation utilisera les plans intermédiaires pour suivre de plus près le mouvement. La figure 9.6 montre I^{IV}_{t-5} et I_t dans le cas de l'exemple que nous avons choisi pour illustrer le prétraitement

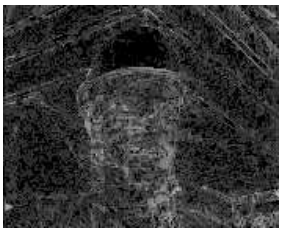
(a) Image Originale 1

(b) Image Originale 2

(c) Image (a) compensée en mouvement par rapport à (b) en ignorant les images intermédiaires

(d) Différence entre l'image (b) et l'image (c). 27.36 dB

(e) Image (a) compensée en mouvement par rapport à (b) en utilisant les images intermédiaires



(f) Différence entre l'image (b) et l'image (e). 29.0 dB

Figure 9.5: Comparaison de la compensation de mouvement par block-matching toutes les 5 trames et trame par trame.

(voir figure 9.2). Les zones qui apparaisent en blanc sur l'image compensée correspondent aux pixels qui ont été rejetés à cause de la déconnexion des labels de la segmentation. La figure 9.7 montre la segmentation compensée.

Figure 9.6: Image I qui réunit l'image I_{t-1} compensée par rapport à I_t et I_t.

Figure 9.7: Image S_{t-1} compensée par rapport à t. En noir les zones de conflit.

9.2.3 Elimination des régions de transition

La procédure d'élimination des régions de transition comporte deux étapes: sélection des régions de transition et élimination.

9.2.3.1 Sélection des régions de transition

Rappelons que la procédure récursive travaille sur des images tridimensionnelles constituée par la réunion de l'image précédente et de l'image courante. L'image S'_{t-1} contient la segmentation précédente (compensée) dans laquelle, par hypothèse, les régions de transition ont déjà été éliminées. Maintenant il s'agit d'éliminer les régions de l'image du temps courant. L'algorithme de sélection est le même que pour une image bidimensionnelle: toutes les zones plates de I_t qui après le filtrage aréolaire ont une aire inférieure à la moitié de la taille du filtre 2D (un quart du filtre 3D) sont considérées comme étant des zones de transition. Sur la figure 9.8 on peut voir d'un côté S'_{t-1} et de l'autre, les marqueurs en blanc et les zones plates de I_t qui ont été choisies comme zones de transition en noir.

9.2.3.2 Elimination des régions de transition

L'élimination des régions de transition consiste à les assigner aux régions voisines de taille supérieure, utilisant un processus de croissance de marqueurs par régions. La procédure est la même qu'en 2D, ce qui change sont les marqueurs:

Figure 9.8: A gauche S'_{t-1} avec les zones de conflit en noir. A droite les marqueurs en blanc et les zones de transition en noir.

ce ne sont plus seulement les régions du temps t de taille supérieure au seuil indiqué, mais aussi les régions de la segmentation précédente.

Cette combinaison de marqueurs, avec le critère de croissance basé sur le contraste et sur la simplicité de contours (le même que celui utilisé dans la section 8.2.3), nous fournit une sursegmentation du temps t dont les contours sont cohérents avec les contours de la segmentation précédente et par ailleurs la sursegmentation contient des germes de nouvelles régions au temps courant. Cela permet la cristallisation spontanée de nouvelles régions sans avoir besoin de les inclure a posteriori comme dans [39]. Sur la figure 9.9 nous pouvons voir le résultat du prétraitement appliqué à notre exemple (voir figure 9.2). A gauche on a S'_{t-1} et à droite la sursegmentation du temps t (on a mis dans chaque région la moyenne des niveaux de gris).

Figure 9.9: Résultat du prétraitement.

9.3 Critères de fusion

De la même façon que dans le cas bidimensionnel, les critères de fusion se succèdent en ordre croissant de complexité. Les premières fusions se réalisent sur des petites zones plates qui n'admettent pas des critères complexes comme la texture ou le mouvement. Au fur et à mesure que les zones plates croissent les critères utilisés peuvent se servir d'une information plus significative que le simple niveau de gris.

9.3.1 Critère de contraste

Dans le cas bidimensionnel, le résultat du prétraitement est simplifié par fusions selon le contraste local entre régions jusqu'à obtenir une segmentation avec un

nombre de points de contour prédéterminé. En 3D les contours sont des surfaces et le nombre de points de contour dépend du mouvement des objets. Or, un nombre élevé de points de contour ne signifie pas "complexe à coder". En effet, un objet peut subir un fort déplacement (générant un nombre élevé de points de contour) et pourtant être facilement codé grâce à une bonne modélisation de son mouvement. C'est pourquoi le critère d'arrêt utilisé en 3D n'est pas le nombre de points de contour, mais un seuil de contraste, celui utilisé en 2D pour obtenir le nombre de points de contour demandé. Ainsi toutes les régions dont le contraste local est inférieur au seuil calculé en 2D sont fusionnées.

Pour le traitement de séquences, l'image que nous avons à simplifier est une image tridimensionnelle et comme nous l'avons déjà décrit, les fusions que nous pouvons réaliser comportent soit deux régions du temps t, soit une région du temps $t-1$ et une autre du temps t, mais jamais deux régions du temps $t-1$ parce qu'elles appartiennent à une segmentation qui a déjà été établie.

Or, les fusions qui comportent une région du temps courant et une autre du temps précédent sont plus intéressantes du point de vue de codage, parce qu'elles contribuent à garder une stabilité temporelle. Si nous considérons toutes les frontières (exceptées celles qui ont un marqueur de chaque côté) et que nous les ordonnons selon leur contraste il est possible que nous réalisions d'abord des fusions spatiales (qui génèrent des contours temporels) alors que la fusion temporelle (plus intéressante du point de vue de codage) n'avait pas un contraste sensiblement supérieur. Pour éviter les instabilités dues à des petites différences d'évaluation des frontières nous commençons par fusionner toutes les régions qui sont complètement incluses dans une région précédente et qui ont un contraste inférieur au seuil calculé en 2D. Dans la pratique la contrainte d'inclusion complète est relâchée permettant la fusion des régions qui sont incluses à 90% dans une région précédente. Ainsi, les fusions intéressantes d'un point de vue du codage sont réalisées en premier, donnant priorité aux fusions qui gardent la stabilité temporelle; tant que le contraste dans l'axe temporel n'est pas fort la segmentation précédente est gardée. La figure 9.10 montre les fusions prioritaires qui ont été réalisées à partir de l'image de la figure 9.9.

Une fois que cette prmière phase de fusions est terminée, les autres frontières sont examinées à leur tour. L'algorithme de fusion de régions selon un critère de contraste local est appliqué à l'image tridimensionnelle résultante, le crière d'arrêt étant le seuil de contraste calculé en deux dimensions. Maintenant les régions du temps t peuvent fusionner avec le passé (sans la contrainte de 90% d'inclusion) ainsi qu'avec d'autres régions du temps t. Cette deuxième étape permet la déformation des régions et la cristallisation de nouvelles régions, c'est pourquoi elle est réalisée après les fusions purement temporelles qui sont plus intéressantes du point de vue de la stabilité. La figure 9.11(a) montre le résultat des fusions par contraste et la figure 9.11(b) montre les régions qui ont cristallisé au temps courant, sans s'associer à une région du passé.

9.3.2 Critère de texture

Jusqu'à maintenant les seules fusions réalisées ont été basées sur le contraste. Nous sommes ainsi arrivés à une segmentation qui comprend les régions du passé à leur nouvel emplacement ainsi que des candidates à nouvelles régions.

Dans le cas bidimensionnel, après les fusions par contraste, nous vérifiions si l'ensemble de plusieurs régions pouvait être représenté par le même modèle de

(a) En blanc, les fusions temporelles prioritaires

(b) Image codée avec le niveau de gris moyen après les fusions prioritaires

Figure 9.10: Fusions prioritaires à partir de l'image 9.9.

(a) Image segmentée codée avec le niveau de gris moyen

(b) Candidates à de nouvelles régions

Figure 9.11: Résultat après fusions par contraste.

texture, situation dans laquelle les régions étaient fusionnées.

Il est possible que certaines candidates à nouvelles régions proviennent des régions qui ont été fusionnées dans le passé par un critère de texture que nous n'avons pas appliqué pour l'instant au temps courant. C'est pourquoi nous vérifions si ces nouvelles régions peuvent être représentées par le même modèle de texture qu'une voisine. Dans le cas affirmatif les régions sont fusionnées, ce qui entraîne la disparition d'une nouvelle région présumée.

Ainsi, parmi les candidats à nouvelles régions de la figure 9.11(b), seulement deux ont été retenues (voir figure 9.12). Elles correspondent aux zones de l'image découvertes à cause du mouvement de caméra vers le bas et vers la droite.

Figure 9.12: Nouvelles régions après fusions par texture.

9.3.3 Critère de mouvement

Une séquence d'images montre l'évolution d'une scène au long du temps. Hormis les changements de scène, les changements entre deux images successives sont dus soit au mouvement des objets dans la scène, soit au mouvement de la caméra. Dans les deux cas, l'information de mouvement entre deux plans successifs est très pertinente et permet de coder de manière économe l'image t à partir de l'image $t - p$.

Dans le cadre du projet $MAVT$ [25, 7] il a été développé un algorithme d'estimation de mouvement orienté objet, qui a l'avantage, par rapport aux méthodes classiques, de travailler avec des objets au lieu de le faire avec des blocs. Les blocs ont une forme régulière, indépendante du contenu de l'image, tandis que les objets (régions de la segmentation) s'adaptent au contenu de l'image et il est plus probable qu'ils correspondent à des objets physiques, soumis à un mouvement homogène. Les estimations de mouvement calculées sur des objets sont ainsi plus performantes. Cette estimation de mouvement produit un ensemble de paramètres de mouvement affine (6 paramètres) pour chacune des régions. Dans la suite, quand nous parlerons de vecteur de mouvement pour une région, en fait nous ferons référence à cet ensemble de paramètres.

Utilisant cet algorithme d'estimation de mouvement orienté objet, le codage de l'image t d'une séquence consiste à estimer le mouvement pour chacune des régions de la segmentation S_t par rapport à la dernière image codée, l'image $t - p$. Ensuite, l'image codée du temps $t - p$ est compensée en utilisant le mouvement estimé et l'erreur de prédiction (différence entre l'image compensée et l'image originale) est corrigé.

Ainsi l'information nécessaire pour reconstituer l'image t en réception est: l'image $t - p$ codée, un vecteur de mouvement par région et la correction de l'erreur de prédiction (compensation). Or, il est possible (même courant) que

l'ensemble de plusieurs régions bouge de manière cohérente. Imaginons par exemple la figure d'un personnage: les yeux, le nez, la bouche bougent à l'unisson avec le mouvement de la tête. Le cas extrême est un mouvement de caméra, situation dans laquelle un seul vecteur de mouvement compense correctement l'image entière (exceptées évidement les zones nouvelles). Si le mouvement de plusieurs régions est cohérent, du point de vue du codage, il est intéressant de les fusionner parce que:

- l'ensemble des régions peut être compensé correctement par un seul vecteur de mouvement sans augmenter de manière significative l'erreur de prédiction.
- l'erreur de prédiction des contours communs aux régions fusionnées ne doit pas être codée.

Notre objectif est maintenant de trouver un critère qui nous permette de décider si le mouvement de deux régions est cohérent ou non.

9.3.3.1 Mesures de ressemblance entre régions adjacentes basées sur le mouvement

Les critères auxquels on peut penser ressemblent à ceux présentés pour la texture.

Ressemblance des paramètres de mouvement Premièrement nous pouvons penser à comparer les paramètres de mouvement eux-mêmes. Ce critère est basé sur l'hypothèse suivante: si deux régions bougent ensemble leurs vecteurs de mouvement sont similaires. Mais cette hypothèse n'est pas vérifiée. En effet, le calcul du vecteur de mouvement d'une région est le résultat d'une optimisation locale sur la région et cela peut nous mener à des résultats très différents pour des régions différentes. Par exemple, l'optimisation locale sur une petite région peut modéliser son mouvement par une forte rotation qui n'existe pas vraiment, simplement parce que, un peu par hasard, un pixel est ainsi mieux compensé, ce qui fait diminuer l'erreur de compensation. Ce vecteur sera très différent des vecteurs voisins. La différence avec un vecteur voisin sera importante, même si à grande échelle les deux régions adjacentes bougent ensemble. La comparaison des paramètres eux mêmes n'est donc pas un bon critère.

Perte de qualité provoquée par la fusion des régions adjacentes La fusion de deux régions implique leur codage conjoint, comme si elles étaient une seule région, ce qui logiquement entraîne une perte de qualité mais aussi un gain en compression. Si les deux régions bougent de manière cohérente le mouvement de l'ensemble peut être bien modélisé par un seul vecteur de mouvement. Par ailleurs, leur fusion implique un gain au niveau du codage des contours et de la texture. Tout ceci apporte un gain en compression sans affecter de manière importante la qualité de l'image. La perte de qualité que la fusion provoque est ainsi un critère pour détecter les régions homogènes au sens du mouvement. Néanmoins, de manière similaire à la discussion de la section 8.3.4 le critère de perte permet la même dégradation pour toute région, tant pour celles qui sont bien représentées que pour celles qui le sont moins bien. Cela génère des images avec des régions de qualité hétérogène.

Qualité des régions adjacentes après fusion L'impression de qualité visuelle d'une image avec des régions de qualité hétérogène est négative: on remarque les défauts, surtout à côté des régions de très bonne qualité. C'est pourquoi nous préférons des critères de fusion basés sur la valeur absolue de la qualité et non pas sur une valeur relative telle que la perte de qualité. De la même façon que dans les fusions basées sur la texture, l'estimation de qualité après fusion sera réalisée de part et d'autre de la frontière, de manière à assurer individuellement une qualité minimale sur chacune des régions fusionnées. Autrement une petite région risquerait d'être fortement endommagée lors de sa fusion avec une grande région voisine; en effet, sa contribution à l'erreur serait négligeable.

Plus la qualité (Q) de compensation de la région qui résulte d'une fusion est grande, plus la fusion est prioritaire.

9.3.3.2 Estimation de la qualité après une fusion par mouvement

Maintenant que nous avons choisi comme critère de fusion la qualité de compensation après fusion, intéressons nous à la manière de calculer cette qualité.

Estimation exhaustive Imaginons deux régions adjacentes R_1 et R_2 avec leurs vecteurs de mouvement respectifs $\vec{v}_1$ et $\vec{v}_2$. Les vecteurs $\vec{v}_1$ et $\vec{v}_2$ ont été calculés pour optimiser la compensation des régions R_1 et R_2 respectivement. Aucun des deux n'optimise la compensation de $R = R_1 \cup R_2$. Pourtant, après la fusion, la région qui doit être compensée est R. Pour estimer la qualité de compensation après fusion, il serait souhaitable de calculer le vecteur de mouvement $\vec{v}$ associé à la région R, région qui résulte de la fusion de R_1 et de R_2.

Néanmoins, cela nous oblige à calculer un vecteur de mouvement pour chaque couple de régions adjacentes, ainsi que pour chaque région autour d'une région fusionnée, de façon à actualiser les évaluations des frontières après chaque fusion. Cette solution est très coûteuse du point de vue du temps de calcul. C'est pourquoi nous allons utiliser une solution sous-optimale qui est un compromis entre la complexité algorithmique et l'efficacité.

Estimation sous-optimale L'obtention d'un vecteur de mouvement pour chaque fusion possible mène à un algorithme très coûteux en temps de calcul. Examinons une solution sous-optimale qui accélère la procédure tout en préservant un bon comportement de l'algorithme.

La solution sous-optimale que nous proposons consiste à considérer les vecteurs $\vec{v}_1$ et $\vec{v}_2$ (calculés de manière indépendante pour optimiser les régions R_1 et R_2 respectivement) comme deux approximations différentes du mouvement de la région R. Si un des deux vecteurs compense correctement les deux régions on pourra conclure qu'elles peuvent être compensées ensemble (avec un seul vecteur de mouvement) et en conséquence qu'elles sont soumises au même mouvement. Si par contre la qualité de compensation de R avec $\vec{v}_1$ et $\vec{v}_2$ est mauvaise, on ne peut pas conclure que les deux régions ne puissent pas être correctement compensées ensemble. Peut être qu'une optimisation globale sur l'ensemble des deux régions aurait donné une bonne compensation.

Détaillons la procédure sous-optimale que nous proposons. Imaginons deux régions R_1 et R_2 et leurs vecteurs de mouvement respectifs $\vec{v}_1$ et $\vec{v}_2$. Considérons d'abord $\vec{v}_1$ comme vecteur de mouvement de R. Les régions R_1 et R_2 sont

compensées avec $\vec{v}_1$ obtenant les qualités respectives Q_1^1 et Q_2^1. L'estimation de qualité globale est le minimum des deux ($Q_1 = min(Q_1^1, Q_2^1)$), assurant ainsi une qualité minimale de chaque côté de la frontière. Les mêmes opérations sont réalisées avec le vecteur $\vec{v}_2$ obtenant les qualités partiales Q_1^2 , Q_2^2 et l'estimation de qualité globale avec $\vec{v}_2$ est de $Q_2 = min(Q_1^2, Q_2^2)$. L'estimation de qualité que nous attribuons à la région fusionnée est le maximum des deux ($Q = max(Q_1, Q_2)$). La figure 9.13 illustre l'évaluation de la qualité d'une région après une fusion basée sur le mouvement. Elle correspond à la meilleure qualité de compensation que nous pouvons obtenir à partir des vecteurs de mouvement optimisés pour une des deux régions. Cette qualité est une borne inférieure qui peut être améliorée par une optimisation globale du vecteur de mouvement dans la région fusionnée.

Le vecteur de mouvement associé à R est $\vec{v}_1$ si Q_1 est supérieure à Q_2 ou $\vec{v}_2$ si Q_2 est supérieure à Q_1.

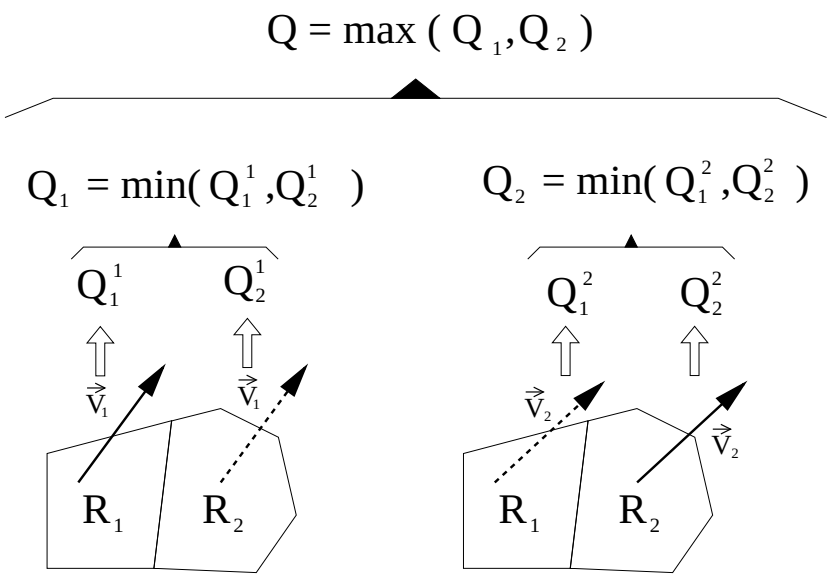


Figure 9.13: Evaluation de qualité après fusion basée sur le mouvement.

Conséquences d'une estimation sous-optimale L'avantage de ne pas calculer le vecteur de mouvement associé à la région qui résulte de la fusion est clairement la rapidité de calcul, mais quels en sont les inconvénients?

L'inconvénient est que la compensation de mouvement n'est pas aussi bonne qu'elle pourrait l'être. En effet, le vecteur de mouvement qu'on garde pour R n'est pas optimal puisqu'il avait été calculé pour R_1 ou R_2. De plus, étant donné qu'il s'agit d'une optimisation locale, $\vec{v}_1$ ou $\vec{v}_2$ peuvent ne pas exprimer le vrai mouvement, surtout dans le cas de petites régions. C'est pourquoi nous appliquons le vecteur $\vec{v}_1$ à la région R_2 et vice versa et nous choisissons le vecteur qui donne les meilleurs résultats. Cela réduit les possibilités d'échec de l'estimation sous-optimale. Si $\vec{v}_1$ est un vecteur faussé par l'estimation locale, nous espérons que $\vec{v}_2$ exprime le vrai mouvement. Si par contre, les deux vecteurs modélisent mal le vrai mouvement, l'estimation de qualité après fusion sera

fausse. Les régions ne seront pas fusionnées, même si elles sont soumises au même mouvement. C'est le prix à payer pour travailler avec une estimation sous-optimale.

Néanmoins l'estimation sous-optimale échoue seulement quand les deux vecteurs de mouvement des régions adjacentes sont faux, situation qui n'arrive pas souvent. La solution que nous avons adoptée est conservative: deux régions qui bougent ensemble peuvent ne pas être fusionnées, mais si deux régions sont fusionnées elles sont forcément bien représentées ensemble.

Dans ces conditions, l'utilisation de l'algorithme sous-optimal est un bon compromis entre la complexité algorithmique et l'efficacité. Voyons quelques résultats de l'application de ce critère à des images réelles.

9.3.3.3 Résultats de l'algorithme de fusion des régions selon un critère de mouvement

L'algorithme de fusion de régions par mouvement consiste à:

1. pour tout couple de régions adjacentes évaluer la qualité après fusion
2. trouver la frontière (R_1, R_2) qui garde la meilleure qualité après fusion.
3. fusionner les régions R_1 et R_2.
4. actualiser les frontières autour de $R = R_1 \cup R_2$
5. réitérer à partir du point 2 jusqu'à ce qu'une fusion supplémentaire génère une région de qualité inférieure à un certain seuil.

Voyons quelques résultats. Prenons deux images correspondant respectivement aux temps $t-5$ et t (figures 9.14 (a) et 9.14 (b)) et la segmentation correspondante au temps t (fig. 9.14(c)). Cette segmentation contient 92 régions et 4922 points de contour. Si on transmet un vecteur de mouvement pour chacune des régions de 9.14(c) la compensation de mouvement obtenue est sur la figure 9.14(d). La simplification par mouvement de l'image 9.14(c) donne le résultat de la figure 9.14(e) dont la compensation de mouvement obtenue est sur la figure 9.14(f).

9.4 Conclusion

Dans ce chapitre nous avons abordé le problème de la segmentation de séquences d'images par fusion de régions. Résumons les points principaux que nous avons traités.

Pour commencer, nous avons cherché à favoriser la stabilité temporelle de la segmentation. En effet, l'algorithme de fusion de régions permet la réunion de n'importe quel couple de régions voisines. En particulier, si nous travaillons avec une fenêtre glissante qui comprend le temps précédent et le temps courant, deux régions de la segmentation précédente pourraient fusionner, provoquant des instabilités au cours du temps. Pour éviter cela, nous avons introduit dans les algorithmes de fusion de régions le concept de "marqueur". Un marqueur est une région considérée comme importante, qui doit correspondre à une région de la segmentation finale. Ainsi l'algorithme de fusion de régions a été modifié

(a) Image Originale 1

(b) Image Originale 2

(c) Image (b) segmentée avec 89 régions et 4988 points de contour

(d) Image (a) compensée par rapport à (b) à partir des régions de l'image (c)

(e) Régions de mouvement homogène obtenues par fusion à partir de (c). 36 régions et 2951 points de contour

(f) Image (a) compensée par rapport à (b) à partir des régions de l'image (e)

Figure 9.14: Exemple de fusion de régions basées sur le critère de mouvement

pour interdire les fusions entre régions marquées. C'est pourquoi la segmentation finale comporte au moins autant de régions que de marqueurs initialement sélectionnés. Pour segmenter la fenêtre glissante de l'approche récursive nous utilisons comme marqueurs les régions de la segmentation précédente. Cette approche nous permet d'imposer une stabilité temporelle (nous récupérons les régions précédentes au temps courant) et en même temps elle nous permet d'inclure de nouvelles régions par cristallisation indépendante au temps courant (sans sélection préalable de marqueurs pour ces nouvelles régions).

Ensuite, nous avons développé une étape de prétraitement qui simplifie l'image avant sa segmentation. Le prétraitement comporte les étapes suivantes:

- Un filtre aréolaire de petite taille qui élimine essentiellement du bruit et qui nous aide à repérer les zones de transition.
- Une compensation de mouvement entre l'image à segmenter et celle qui vient d'être segmentée pour réduire les différences dues au mouvement entre les deux trames. Ceci contribue aussi à la stabilité temporelle de la segmentation (surtout dans le cas des petites régions).
- Elimination des zones de transition qui faussent l'estimation de contraste entre régions voisines. Pour cela nous utilisons l'algorithme de croissance de marqueurs par régions prenant comme marqueurs les régions de la segmentation précédente ainsi que les régions du temps courant de taille supérieure à un certain seuil.

Dans la section suivante nous avons présenté différents critères de fusion utilisés pour segmenter les séquences. Les premières fusions concernent des zones plates petites et nombreuses qui n'admettent pas d'attributs complexes tels qu'une représentation de texture ou un vecteur de mouvement. C'est pourquoi nous commençons par un critère simple qui est le contraste local. Nous avons distingué trois types de fusions dans la fenêtre glissante de l'approche récursive:

1. les fusions temporelles qui concernent une région du passé et une autre du présent, avec une forte intersection temporelle. Ces fusions sont les plus intéressantes du point de vue du codage parce qu'elles gardent une stabilité temporelle.
2. les fusions entre une région du passé et une région du présent qui se trouve à cheval sur plusieurs régions du passé. Ces fusions sont celles qui permettent la déformation des objets présents dans la scène.
3. les fusions entre deux régions du présent. Ce sont les fusions qui permettent l'apparition de nouvelles régions.

Pour favoriser la stabilité temporelle nous avons donné la priorité aux fusions entre régions qui ont une forte intersection temporelle (fusions du type 1). Une fois toutes les fusions du type 1 réalisées, nous opérons les fusions du type 2 et 3. Elles sont moins intéressantes du point de vue du taux de compression, mais elles sont nécessaires pour suivre correctement l'évolution de la scène.

Ensuite, de la même façon que dans le cas bidimensionnel, nous utilisons un critère de fusion de régions basé sur la texture.

Finalement nous avons introduit un critère de fusion de régions basé sur la ressemblance de mouvement de régions voisines. Ce critère est d'une grande

importance pour le traitement de séquences parce que les différences entre deux plans successifs sont essentiellement dues au mouvement des objets présents dans la scène. L'analyse des objets en vue de repérer ceux qui ont un mouvement cohérent permet d'obtenir des taux de compression plus intéressants. Néanmoins, l'utilisation de ce critère dans la segmentation provoque des disfonctionnements au niveau d'autres étapes de la boucle de codage. Pour un fonctionnement efficace du système global ces étapes ont besoin d'être adaptées. Les modifications nécessaires pour obtenir un système cohérent sont présentées dans le chapitre suivant.

Chapter 10

Introduction de l'algorithme de fusion de régions par mouvement dans la boucle de codage

La fusion des régions qui ont un mouvement cohérent permet de coder l'évolution d'une scène de manière plus comprimée. Les raisons sont les suivantes:

- seulement un sous-ensemble de vecteurs de mouvement doit être codé
- l'erreur de prédiction des contours communs aux régions fusionnées ne doit pas être codé
- l'erreur de prédiction du contenu des régions n'augmente pas de manière significative

C'est pourquoi il est intéressant de détecter dans la segmentation les régions qui bougent ensemble.

A ce propos nous avons développé, dans le chapitre précédent, un critère de fusion de régions basé sur la ressemblance de mouvement des régions voisines. Son introduction dans la boucle de codage améliore (augmente) le rapport qualité / coût du système. Mais, malheureusement, le fait de travailler avec des régions homogènes au sens mouvement trouble le bon fonctionnement d'autres étapes de la boucle. Voyons les causes précises des modifications nécessaires, ainsi que les solutions que nous avons apportées.

Les étapes qui ont besoin d'une adaptation sont:

- la segmentation
- le codage de contour
- la compensation de mouvement

10.1 Modifications de la segmentation

Commençons par rappeler les lignes générales de l'algorithme de segmentation que nous avons développé.

L'algorithme de segmentation suit une approche récursive, c'est-à-dire, la segmentation précédente est utilisée pour segmenter le temps courant. La méthode consiste à générer une image tridimensionnelle qui réunit le temps précédent (déjà segmenté) et le plan courant (à segmenter).

Avec la segmentation précédente dans le premier plan de l'image tridimensionnelle et l'image courante légèrement simplifiée dans le deuxième, nous réalisons des fusions spatio-temporelles. Les critères de fusion sont la ressemblance de niveau de gris, de texture ou de mouvement des régions adjacentes. Dans la procédure de fusion, les régions de la segmentation précédente jouent le rôle de marqueurs pour imposer une stabilité temporelle.

Or, si les régions de la segmentation précédente proviennent des fusions par mouvement, leur contenu n'est pas homogène au sens de la texture. En conséquence, leur niveau de gris et leurs paramètres de texture ne sont pas représentatifs des régions. C'est pourquoi les fusions basées sur la ressemblance de niveau de gris ou de texture, ne peuvent pas utiliser comme marqueurs les régions de la segmentation précédente si celles-ci contient des fusions par mouvement.

Pour éviter le problème dû au fait que les régions homogènes au sens du mouvement peuvent ne pas l'être au sens de la texture, nous proposons de travailler à deux niveaux de résolution:

- un niveau de segmentation fine qui contient des régions homogènes au sens de la texture.
- un niveau de segmentation grossière qui contient des régions homogènes au sens mouvement. Par construction les régions de la segmentation grossière seront l'union d'une ou plusieurs régions de la segmentation fine.

La segmentation fine est utilisée côté émetteur pour aider à la projection des régions précédentes, tandis que la segmentation grossière est celle qui est envoyée au récepteur, nous permettant de coder l'évolution de la séquence de manière plus comprimée. La figure 10.1 montre un exemple de segmentation fine et grossière dans un cas réel. L'axe horizontal correspond au temps.

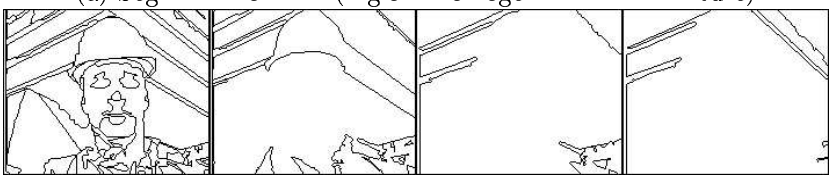
10.1.1 Génération de la segmentation fine

La segmentation fine est générée côté émetteur et elle n'est pas envoyée au récepteur. Elle contient des régions homogènes au sens texture, c'est-à-dire qu'elle n'admet pas les fusions par mouvement.

En mode INTRA l'émetteur doit envoyer des régions homogènes au sens texture, parce que le contenu des régions doit être codé, il n'est pas connu par le récepteur. De même les nouvelles régions qui apparaissent au long de la séquence doivent être homogènes en texture. Par contre, en mode INTER, le contenu des régions est codé par compensation de mouvement et un seul vecteur peut compenser des régions qui ont des textures différentes.

L'obtention de la segmentation fine consiste à suivre au cours du temps toutes les régions qui ont été codées en mode INTRA à un moment donné

(a) Segmentation fine (régions homogènes au sens texture)



(b) Segmentation grossière (régions homogènes au sens mouvement)

Figure 10.1: Deux niveaux de résolution

de la séquence, même si après elles ont été fusionnées par mouvement et, en conséquence, n'ont pas été envoyées au récepteur.

10.1.2 Génération de la segmentation grossière

Chaque région de la segmentation grossière est une réunion de régions de la segmentation fine. Les régions appartenant à cette réunion ont été choisies parce qu'elles bougent de façon similaire. Etudions plusieurs façons d'obtenir la segmentation grossière à partir de la segmentation fine.

Une première approche consisterait à appliquer, à chaque instant de la séquence, l'algorithme de fusion de régions par mouvement présenté précédemment. Cette approche réalise des fusions au temps t sans tenir en compte celles qui avaient été réalisées au temps $t-p$. Le résultat est une segmentation instable dans le temps qui est chère à coder et qui génère un effet de papillotement très gênant pour l'œil humain.

Une deuxième solution consisterait à fusionner au temps t les régions de la segmentation fine qui étaient déjà fusionnées au temps $t-p$. Cette approche génère des segmentations stables dans le temps, mais elle ne suit pas correctement l'évolution de la séquence. En effet, un objet qui au temps $t-p$ avait le même mouvement qu'un voisin, peut bouger de manière indépendante au temps t. Imaginons par exemple un personnage dans la scène qui est statique jusqu'au temps $t-p$ et qui au temps t commence à se déplacer. Dans ce cas là, la compensation du personnage avec le fond sera de bonne qualité jusqu'à ce que le personnage commence à se déplacer.

Pour être capables de suivre correctement l'évolution de la scène tout en gardant une segmentation stable, nous proposons une troisième solution. Elle permet de séparer dans la segmentation grossière, des régions qui précédemment bougeaient ensemble et qui au temps courant ont un mouvement indépendant. Cette troisième solution est réalisée en trois étapes:

- Premièrement les régions de la segmentation fine qui avaient été fusionnées au temps précédent sont fusionnées au temps courant. Précisons. Imaginons une région de la segmentation grossière précédente RG_1^{t-p} qui comprend trois régions de la segmentation fine (R_1^{t-p}, R_2^{t-p}, et R_3^{t-p}).

Au temps courant nous commençons par supposer que RG_1^t est toujours l'union des trois régions R_1^t, R_2^t et R_3^t, qui correspondent aux projections des trois régions précédentes.

Les paramètres de mouvement des régions grossières ainsi générées sont calculés et la compensation correspondante est obtenue.

- Ensuite, la qualité de la compensation qu'on vient d'obtenir est estimée sur chacune des régions de la segmentation fine ($R_1^t, R_2^t ...$). Les régions pour lesquelles la qualité est insuffisante sont rajoutées dans la segmentation grossière, de manière à améliorer leur compensation de mouvement. Imaginons que la compensation conjointe des trois régions fusionnées précédement produit une mauvaise qualité de compensation sur R_1^t. Dans cette situation, on rajouterait R_1^t à la segmentation grossière. Elle bougeait ensemble avec R_2^t et R_3^t au temps précédent mais au temps courant elle bouge de manière indépendante. Sa compensation en mouvement est possible parce qu'il ne s'agit pas d'une région nouvelle. Elle existait déjà au temps précédent R_1^{t-p}. Seulement, au temps courant elle bouge de manière différente.

- Ensuite la segmentation grossière est mise à jour en utilisant l'algorithme de fusion de régions par mouvement. La mise à jour consiste à:

  - fusionner les régions qui étaient séparées au temps précédent et qui au temps courant bougent ensemble
  - fusionner certaines régions qui ont été rajoutées parce qu'elles étaient mal représentées. Il est possible qu'un sous-ensemble de ces régions soit bien représenté par un seul vecteur de mouvement, par exemple, toutes les régions qui composent le personnage qui commence à bouger.

Cette procédure donne une stabilité à la segmentation parce que les fusions prioritaires sont celles qui avaient été faites dans le passé et, en même temps, permet le mouvement indépendant d'objets (rajoutant dans la segmentation grossière les régions de la segmentation fine qui sont mal représentées ensemble).

La figure 10.2 illustre l'évolution des deux segmentations (fine et grossière) dans un exemple dessiné à la main. Au temps t, nous obtenons une segmentation (A-1) avec des régions homogènes au sens texture (segmentation INTRA): la segmentation fine. Cette segmentation est projetée (A-2) sur le temps suivant (t+p). Les fusions par mouvement précédentes (pour l'instant aucune) sont appliquées sur A-2 et nous obtenons B-2. Les régions de A-2 qui sont mal compensées à partir de B-2 sont rajoutées à B-2, obtenant C-2 (pour l'instant A-2 = B-2 = C-2). Ensuite nous appliquons sur C-2 l'algorithme de fusion par mouvement et nous obtenons la segmentation grossière correspondant au temps $t + p$ (D-2). C'est D-2 qui est envoyée au récepteur, mais l'émetteur projette au temps $t + 2p$ la segmentation fine (A-2). La segmentation fine au temps $t + 2p$ est donc sur la figure A-3. La première approximation de la segmentation grossière au temps $t + 2p$ consiste à réaliser les fusions existantes entre A-2 et D-2 et nous obtenons B-3. A partir de B-3 nous pouvons resegmenter les régions si les macro-régions précédentes ne bougent plus ensemble, ce qui n'est pas le cas. C'est pourquoi C-3 = B-3. A partir de C-3 nous pouvons réaliser

des fusions supplémentaires si plusieurs macro-régions précédentes ont un mouvement cohérent. Nous obtenons ainsi D-3 où le personnage a fusionné avec le fond. La segmentation fine au temps $t + 3p$ (A-4) est obtenue par projection de A-3. Les fusions existantes entre A-3 et D-3 sont appliquées sur A-4 et nous obtenons B-4. Nous supposons ainsi que le personnage continue à bouger de manière cohérente avec le fond, mais il a entrepris un mouvement indépendant vers la droite. Quand nous réalisons la compensation de mouvement à partir de B-4, nous nous apercevons que la compensation du personnage est de mauvaise qualité. C'est pourquoi il est resegmenté sur la figure C-4. Toutes les régions du personnage sont rajoutées sur C-4, parce qu'elles sont toutes mal compensées. Néanmoins le personnage entier continue à bouger de manière cohérente. C'est pourquoi l'algorithme de fusion par mouvement refusionne le personnage en une seule région (D-4). Nous pouvons observer que la nouvelle région qui est apparue dans la segmentation fine, au coin en bas à gauche, apparaît aussi sur la segmentation grossière.

Voyons maintenant, sur la figure 10.3, toutes les images intermédiaires générées lors du passage d'une segmentation fine à une grossière, à un moment donné d'une séquence réelle. Prenons l'image codée précédente 10.3(a) et l'image courante 10.3(b). La segmentation fine précédente 10.3(c) est projetée au temps courant 10.3(d). Nous commençons par supposer qu'au temps courant, le groupement de régions de la segmentation fine pour obtenir la segmentation grossière est le même qu'au temps précédent. Ainsi, nous réalisons sur 10.3(d) les mêmes fusions qui ont été réalisées entre 10.3(c) et 10.3(e). Nous obtenons ainsi l'image 10.3(f). Ensuite nous calculons les vecteurs de mouvement associés à chacune des régions de 10.3(f) et nous obtenons l'image compensée 10.3(g). Nous avons superposé l'image compensée et la segmentation fine pour montrer que l'estimation de mouvement n'a pas réussi à modéliser correctement le mouvement de la région du fond. La qualité de cette compensation est étudiée sur chacune des régions de la segmentation fine 10.3(d). Les régions de la segmentation fine qui ont une qualité de compensation insuffisante sont rajoutées à la segmentation grossière 10.3(h). Finalement nous appliquons à 10.3(h) l'algorithme de fusion de régions basé sur le mouvement et nous obtenons la segmentation grossière courante 10.3(i). Cette dernière étape a été réalisée pour:

- fusionner des régions qui étaient séparées au temps précédent et qui maintenant bougent ensemble. Par exemple, plusieurs régions du visage qui étaient séparées dans les passé sont maintenant fusionnées.
- fusionner des régions qui ayant été rajoutées par mauvaise compensation, appartiennent à un objet de mouvement cohérent. Par exemple, certaines bandes du bâtiment ont été refusionnées après avoir été rajoutées.

La compensation en mouvement de l'image grossière est illustré dans l'image 10.3(j). Nous pouvons constater l'amélioration de qualité par rapport à l'image 10.3(g), sur les régions rajoutées. Par exemple nous pouvons observer les bandes du bâtiment sur la gauche; maintenant elles coïncident avec les régions de la segmentation fine. En même temps les régions qui ont fusionné (par exemple les yeux) gardent une bonne qualité de compensation.

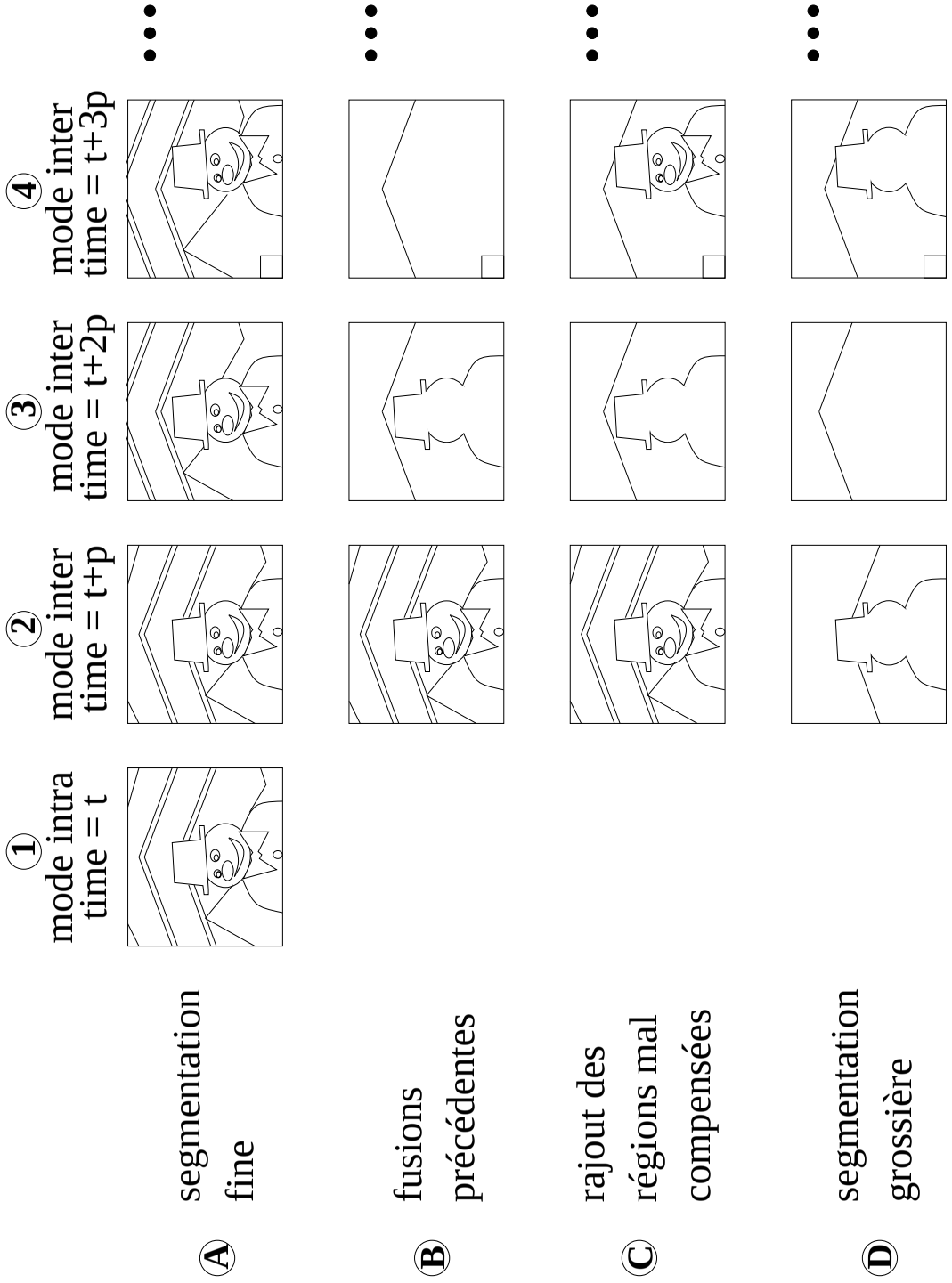


Figure 10.2: Evolution de la segmentation fine et de la segmentation grossière. Etapes intermédiaires pour passer de l'une à l'autre.

(a) Segmentation codée précédente

(b) Image courante

(c) Segmentation fine précédente

(d) Segmentation fine courante

(e) Segmentation grossière précédente

(f) Segmentation courante avec les fusions précédentes

(g) Compensation orientée objet à partir de l'image (f).

(h) Segmentation (f) plus régions mal compensées

(i) Segmentation grossière courante

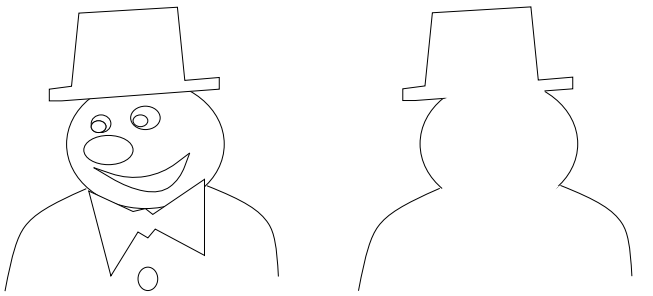
(j) Compensation orientée objet à partir de l'image (i)

Figure 10.3: Exemple du passage de la segmentation fine à la segmentation grossière

10.2 Modifications du codage de contour

Les algorithmes de codage de contour exploitent la ressemblance de la forme des régions au cours du temps, pour coder les contours de manière économe. La procédure consiste à compenser en mouvement les régions de la segmentation précédente et ne coder que les erreurs de prédiction. Cela permet de réduire par un facteur 3 ou 4 le coût de codage des contours en mode INTER par rapport au coût en INTRA.

Par contre, si entre deux plans successifs nous réalisons les fusions des régions qui bougent ensemble, la forme des régions change énormément. En effet, la forme de la région $R = R_1 \cup R_2 \cup R_3$ ne ressemble à la forme d'aucune des trois régions initiales. Du coup l'économie que nous nous attendions à obtenir grâce aux fusions par mouvement est perdue du fait qu'on n'arrive pas à bien compenser le changement de forme.



(a) Segmentation précédente (b)Segmentation courante avec fusions par mouvement

Figure 10.4: Changement de forme provoquée par les fusions par mouvement

La figure 10.4 illustre la problématique du changement de forme. Nous avons au temps t une macro-région qui est l'union de plusieurs régions précédentes. Soient $R_1^{t-p}, R_2^{t-p}, ...$ les régions de la segmentation précédente (fig. 10.4(a)) et $R_1^t, R_2^t...$ leur projection au temps courant. Si $R_1^t, R_2^t...$ sont fusionnées au temps courant, la région ainsi obtenue $R^t = R_1^t \cup R_2^t \cup ...$ (fig. 10.4(b)) ne ressemble individuellement à aucune des régions fusionnées, mais à leur réunion.

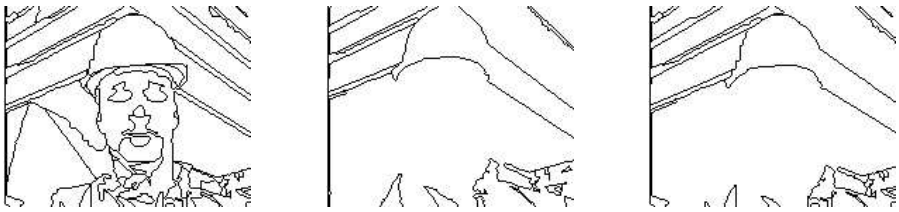
Pour pouvoir bien compenser la forme de la région obtenue par fusion, son vecteur de mouvement doit être appliqué non pas à une région individuelle, mais à l'ensemble des régions réunies ($R^{t-p} = R_1^{t-p} \cup R_2^{t-p} \cup ...$). Cette information a priori n'est pas connue par le récepteur. L'émetteur doit lui envoyer la liste des régions qui ont été fusionnées. De cette manière le récepteur peut aussi les mettre ensemble avant de coder les contours de la segmentation courante.

Quand la segmentation contient des fusions par mouvement, l'algorithme de codage de contours consiste à:

- générer, à partir de la segmentation précédente et la liste de fusions, une segmentation du temps précédent dont les régions sont proches de celles qui vont être codées au temps courant.
- coder la segmentation courante par rapport à la segmentation obtenue au point précédent.

Ainsi l'erreur de prédiction des contours n'augmente pas à cause des fusions par mouvement.

Le récepteur devra utiliser de la même manière la liste de fusions pour produire une segmentation de l'image précédente cohérente avec celle de l'image courante, avant de décoder le buffer de contour. L'image 10.5 illustre cette procédure. Soit l'image 10.5(a) la segmentation précédente et 10.5(b) la segmentation courante. Le codage de contour commence par générer l'image 10.5(c) qui est la segmentation précédente avec les fusions courantes et ensuite les contours de 10.5(b) sont codés en INTER par rapport à ceux de 10.5(c).



(a) Segmentation précédente

(b) Segmentation courante

(c) Segmentation précédente avec les fusions courantes.

Figure 10.5: Codage de contours avec des fusions par mouvement

10.3 Modification de la compensation de mouvement

Les fusions par mouvement introduisent de forts changements dans la forme des régions segmentées. La forme de $R^t = R_1^t \cup R_2^t \cup R_3^t$ est très différente de la forme de chacune des régions R_1^{t-p}, R_2^{t-p} et R_3^{t-p}. Nous avons vu que la liste des régions fusionnées côté émetteur doit être envoyée au récepteur. Cela permet d'appliquer le vecteur de mouvement de R^t à $R^{t-p} = R_1^{t-p} \cup R_2^{t-p} \cup R_3^{t-p}$, ce qui donne une bonne estimation de la forme de R^t. L'envoi de cette information permet de ne pas augmenter le coût de codage des contours à cause des fusions par mouvement. Voyons maintenant les modifications que nous devons inclure pour que la compensation du contenu des régions soit efficace.

Commençons par voir un peu plus en détail les caractéristiques de l'algorithme de compensation. Le mouvement d'une région est modélisé par un jeu de paramètres $(a_1, a_2, a_3...., b_1, b_2, b_3...)$. L'expression du vecteur de déplacement (dx, dy) de chaque pixel (x, y) est donnée par les deux polynômes suivants:

$$dx = a_1 + a_2 \times x + a_3 \times y + ...$$

$$dy = b_1 + b_2 \times x + b_3 \times y + ...$$

Ce vecteur de déplacement est calculé pour chaque pixel (x, y) en utilisant les paramètres $(a_1, a_2..., b_1, b_2...)$ de la région à laquelle il appartient. Ensuite la compensation peut se faire en deux modes:

1. mouvement non-restreint
2. mouvement restreint

Le mode non-restreint compense chaque pixel (x, y) du temps t avec le pixel $(x - dx, y - dy)$ de l'image $t - p$. Aucune contrainte n'est rajoutée à cette compensation.

La compensation de mouvement en mode restreint d'une région du temps courant (R^t) utilise l'information de la position de la région au temps précédent (R^{t-p}). Détaillons la compensation de la région R^t en mode restreint. Elle s'opère en deux étapes:

- pour chaque pixel de la région R^t on calcule son vecteur de déplacement (dx, dy) à partir des paramètres de mouvement de R^t. Si le pixel $(x - dx, y - dy)$ appartient à la région R^{t-p} de la segmentation précédente, alors on effectue la compensation. Sinon, le pixel n'est pas compensé.
- après avoir compensé tous les pixels qui vérifient la condition précédente, une propagation de texture remplit les trous des pixels qui n'ont pas pu être compensés.

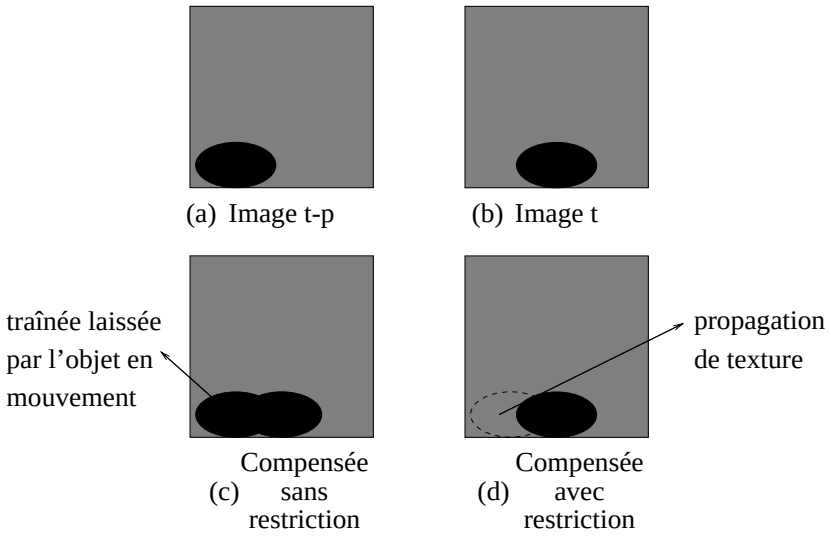


Figure 10.6: Problème de zones découvertes

L'avantage du mode de fonctionnement restreint est qu'il gère mieux le problème des zones découvertes à cause du mouvement. La figure 10.6 illustre les deux modes de fonctionnement. Imaginons un objet qui bouge sur un fond statique (fig. 10.6(a) et 10.6(b)) . La compensation de mouvement sans restriction place l'objet à sa nouvelle position, mais la compensation du fond (pas de mouvement) laissera des traces de l'ancienne position de l'objet (fig. 10.6(c)).

Par contre, le mode de fonctionnement restreint ne compensera pas les pixels qui appartiennent au fond au temps courant mais qui appartenaient à l'objet le temps précédent (zone découverte par le mouvement de l'objet). La zone découverte sera remplie par une propagation de texture des pixels qui ont réussi à être compensés. Ainsi il ne restera pas de trace de l'ancienne position de l'objet (fig. 10.6(d)).

Remarquons que si aucun pixel de la région ne coïncide avec la même région au temps précédent, nul pixel n'est compensé et la propagation de texture

n'est pas possible. Dans ce cas là, la compensation de la région est faite sans contrainte.

La compensation de mouvement en mode restreint donne de meilleurs résultats parce que les pixels compensés appartenaient déjà à la même région au temps précédent. Quand on inclut des fusions par mouvement, quelle région va contraindre la compensation de la région obtenue par fusion? La réponse est claire: l'union des régions qui ont été fusionnées. Cette information est la même que celle qui a été transmise pour coder les contours de manière efficace.

Mais dans la segmentation courante il n'y a pas que des fusions. Il y a aussi des resegmentations. Nous pouvons distinguer deux types de resegmentation qui vont subir deux traitements différents.

Le premier type de resegmentation correspond à l'apparition de nouvelles régions. Ce sont des régions qui ont cristallisé au temps courant, indépendamment du temps précédent. Les nouvelles régions n'ont pas de passé dans la séquence. C'est pourquoi elles sont codées en mode INTRA, sans compensation de mouvement par rapport au temps précédent.

Un deuxième type de resegmentation correspond à l'introduction dans la segmentation grossière des régions de la segmentation fine dont la qualité de compensation sans resegmentation n'est pas suffisante. Ces régions ont un passé dans la séquence. Elles ont été codées et ensuite fusionnées parce qu'elles bougeaient ensemble avec d'autres régions. Maintenant leur mouvement est indépendant. Elles doivent être séparées pour être correctement compensées. Dans cette situation, comment doit agir la restriction de mouvement? Le mouvement de la région rajoutée doit être contraint à l'intérieur de la région de la segmentation grossière précédente qui la contenait, c'est-à-dire, à l'intérieur de la région précédente qui est resegmentée. A priori cette information est disponible au récepteur. Il suffit d'assigner à la région rajoutée la même étiquette qu'elle avait avant d'avoir été fusionnée par mouvement. Le récepteur pourrait, grâce à l'histoire des fusions réalisées, savoir à quelle région grossière elle appartient. Mais pour l'instant, le récepteur n'est pas muni de mémoire. Il ne connaît que le temps précédent immédiat. C'est pourquoi nous devons envoyer pour chaque objet rajouté, l'étiquette de la région précédente dans laquelle il était inclus. Ainsi leur mouvement peut être contraint, améliorant de cette façon la qualité de la compensation.

Par conséquent, les informations nécessaires pour éviter que les fusions par mouvement troublent le fonctionnement d'autres étapes sont les suivantes:

- d'une part la liste des fusions qui ont été réalisées entre le temps t et le temps t+p;
- d'autre part, la liste des resegmentations.

Compatibilité entre la restriction de mouvement et la fusion de régions par mouvement Afin de pouvoir restreindre le mouvement des régions de la segmentation fine qui ont été rajoutées dans la segmentation grossière, l'émetteur doit envoyer au récepteur, pour chacune des régions rajoutées, l'étiquette de la segmentation grossière à laquelle elle appartenait. Mais après l'étape de resegmentation de la segmentation grossière nous réalisons des fusions par

mouvement (voir section 10.1). Par conséquent une nouvelle contrainte est introduite dans l'algorithme de fusions de régions par mouvement: deux régions courantes qui ont été rajoutées à cause de leur mauvaise compensation, peuvent fusionner seulement si elles proviennent de la même région grossière précédente. Autrement, une région rajoutée pourrait rester à cheval sur deux régions grossières précédentes et le récepteur ne saurait pas restreindre le mouvement d'une telle région. Si vraiment ces régions bougent de manière cohérente, elles seront fusionnées le temps d'après.

10.4 Résultats

Dans les chapitres précédents nous avons introduit un algorithme de fusion de régions que nous avons adapté aux particularités propres d'un système de codage. Nous avons mis au point différents critères de fusion que nous avons testés sur des exemples isolés. Maintenant il s'agit d'analyser le comportement de l'algorithme sur des séquences entières. Commençons par rappeler l'algorithme final auquel nous avons abouti.

Les algorithmes de codage de séquences ont deux modes de fonctionnement:

- mode INTRA ou mode d'initialisation. La scène est complètement inconnue par le récepteur et en conséquence elle doit être entièrement codée.
- mode INTER: la scène est déjà connue par le récepteur. On ne transmet que les différences par rapport à la dernière image codée.

Voyons les étapes de notre algorithme selon ces deux modes de fonctionnement:

Segmentation INTRA

Prétraitement

1. filtre aréolaire de petite taille.
2. élimination des régions de transition par croissance de région. Nous considérons comme régions de transition celles qui ont une taille inférieure à un certain seuil à la sortie du filtre aréolaire.

Fusions par contraste local

Les frontières de la segmentation obtenue après le prétraitement sont ordonnées selon leur contraste local et une à une, elles sont éliminées jusqu'à obtenir une segmentation avec un nombre de points de contour adapté au taux de compression visé. Après chaque fusion, les frontières sont ré-évaluées pour tenir compte de l'état courant de la partition. Le contraste de la dernière frontière éliminée est enregistré pour être utilisé en mode INTER.

Fusions par texture

Nous fusionnons les régions voisines qui ont une qualité suffisante lorsqu'elles sont codées avec un seul jeu de paramètres de texture.

Segmentation INTER

Prétraitement

1. filtre aréolaire tridimensionnel (volumique) de petite taille.
2. compensation de mouvement par block-matching de la dernière image codée par rapport à l'image courante
3. élimination des régions de transition de l'image courante.

Fusions par contraste local

Les frontières dont le contraste est inférieur à celui calculé en mode INTRA sont éliminées.

Dans ce processus, afin de rendre la segmentation plus stable, nous avons introduit dans l'algorithme de fusion de régions les deux améliorations suivantes:

- les régions de la segmentation précédente sont considérées comme marqueurs, c'est-à-dire, la fusion entre deux régions précédentes est interdite.

- parmi toutes les frontières dont le contraste est inférieur au seuil indiqué, nous commençons par éliminer les frontières temporelles qui séparent une région du passé et une autre du présent, telles que cette dernière soit incluse au moins à 90% en surface dans la première. Ensuite toute autre frontière (spatiale ou temporelle) qui vérifie la condition de contraste est éliminée.

Le résultat de cette étape est une segmentation qui comprend la projection de la segmentation précédente au temps courant, ainsi que des régions candidates à être de nouvelles régions qui ont cristallisé au temps courant, c'est-à-dire qui proviennent de la fusion entre régions du temps courant exclusivement.

Fusions par texture

De la même façon qu'en mode INTRA, nous vérifions si l'ensemble de plusieurs régions peut être codé avec un seul jeu de paramètres de texture. Pour stabiliser le résultat, les régions précédentes projetées jouent le rôle de marqueurs. Cette étape élimine certaines régions présumées nouvelles parce qu'elles peuvent être correctement codées avec une région voisine.

Fusions par mouvement

Le but de cette étape est de repérer dans l'image les différents plans de mouvement. Cela va nous permettre de ne coder qu'un sous-ensemble des vecteurs de mouvement ainsi qu'un sous-ensemble des corrections de contours et des corrections de texture. Le coût de codage est ainsi réduit, tandis que la qualité de l'image codée n'est pas affectée de manière significative.

Ce critère est le plus complexe à mettre en œuvre parce qu'il affecte le fonctionnement d'autres étapes du codage. Notamment les étapes préalables de la segmentation sont affectées. En effet, nous utilisons dans les étapes préliminaires de simplification des critères basés sur le contraste. Or, le contenu d'une région homogène au sens mouvement ne l'est pas au sens texture, ce qui empêche de calculer le contraste de cette région avec une autre. Pour éviter ce problème nous avons décidé de travailler à deux niveaux de résolution: une segmentation fine, utilisée dans les premières étapes de la segmentation et une segmentation grossière qui est envoyée au récepteur.

Le passage d'une résolution à l'autre est effectué après les fusions basées sur le critère de texture, c'est-à-dire, juste avant d'appliquer le critère de mouvement. Comme nous l'avons déjà vu dans les sections précédentes, le changement de résolution consiste à:

- fusionner les régions qui étaient fusionnées précédemment. Cela nous donne une première approximation de la segmentation grossière basée sur la supposition qu'aucun objet n'a entrepris un mouvement indépendant.

- calculer les vecteurs de mouvement associés aux régions de la segmentation grossière que nous venons de générer.

- compenser l'image précédente avec les vecteurs que nous venons de calculer.

- calculer la qualité de la compensation sur chacune des régions de la segmentation fine courante et repérer celles qui ont une qualité insuffisante.

- rajouter à la segmentation grossière les régions repérées au point précédent, ce qui nous donne une deuxième approximation de la segmentation grossière.
- recalculer les vecteurs de mouvement associés aux régions de la nouvelle segmentation grossière.
- fusionner les régions selon le critère de mouvement. L'effet de ce dernier point est le suivant:
  - fusionner les régions qui bougent de manière cohérente maintenant et qui avaient un mouvement indépendant dans le passé
  - fusionner les régions qui, ayant été rajoutées par mauvaise compensation, appartiennent à un objet de mouvement cohérent. En effet, différentes régions appartenant à un objet qui a entrepris un mouvement indépendant peuvent être compensées avec un seul vecteur de mouvement, une fois qu'elles ont été séparées des autres régions.

10.4.1 Séquence "Foreman"

La séquence est au format QCIF (176×144 pixels) à une fréquence de 30 images par seconde. Nous réalisons un sous-échantillonnage temporel et nous ne traitons qu'une image sur six, c'est-à-dire, cinq images par seconde. La quantité d'information initiale par seconde est donc:

$$5 \frac{images}{seconde} \times (176 \times 144) \frac{pixels}{image} \times 24 \frac{bits}{pixel} = 3041.28 \frac{Kbits}{seconde}$$

Les 24 bits par pixel proviennent du fait que nous traitons des images en couleur (dommage que l'impression soit en noir et blanc!). L'image originale sous-échantillonnée temporellement est sur les figures 10.7 et 10.8.

Nous prenons comme critères d'arrêt 5000 points de contour pour les fusions basées sur le contraste local et 29 dB pour les fusions basées sur la texture et le mouvement. Après les fusions par texture ou mouvement, la porte est ouverte à des re-segmentations dans les trames futures. Si le même seuil de qualité est imposé pour fusionner et pour re-segmenter, nous sommes exposés à avoir des petites oscillations de qualité d'une trame à l'autre; les régions qui ont une qualité aux alentours du seuil (tantôt un peu en dessus (fusion), tantôt un peu en dessous (re-segmentation)), fusionneraient et seraient re-segmentées sans arrêt, provoquant une forte instabilité temporelle. C'est pourquoi nous avons laissé une marge de sécurité en vue de la stabilité temporelle qui consiste à imposer un seuil pour re-segmenter qui est inférieur à celui utilisé pour fusionner, en l'occurrence 26 dB.

Sur les figures 10.9 et 10.10 nous avons la segmentation fine correspondante. Nous constatons que la partition correspond aux objets présents dans la scène: notamment , le bâtiment, le personnage avec ses yeux, son nez, sa bouche. Une fois que les objets ont été détectés dans la première image notre objectif est de les suivre au cours de la séquence pour les coder de manière économe à travers un vecteur de mouvement. Cet objectif a été rempli et nous observons que la segmentation est stable au cours du temps. Les régions se déforment convenablement, s'adaptant à l'évolution des objets dans la scène, ce qui fait

que les régions segmentées à un moment donné de la séquence peuvent être retrouvées de nombreuses images plus tard à leur emplacement correct.

D'autre part, l'algorithme introduit de nouvelles régions quand une zone du temps courant ne ressemble à aucune région du passé. Notamment il a introduit des régions nouvelles dans la première image codée en mode INTER, au coin en bas à droite de l'image. Elles correspondent à la zone des objets qui rentrent dans la scène.

A partir de la segmentation fine des figures 10.9 et 10.10 nous obtenons la segmentation grossière (voir figures 10.11 et 10.12). Elle contient les régions homogènes au sens du mouvement; elles correspondent à l'union d'une ou de plusieurs régions de la segmentation fine. Les fusions et les resegmentations sont basées sur la qualité de l'image compensée. Nous constatons que quand il n'y a pas de mouvement les régions fusionnent, et que quand le mouvement reprend, les régions sont resegmentées pour améliorer la compensation de mouvement.

Regardons un peu plus en détail cette segmentation grossière. Entre l'image 0 et l'image 6 la caméra bouger ver le bas à droite. Etant donné qu'il s'agit d'un mouvement de caméra, toutes les régions sont soumises à un mouvement similaire et l'application de l'algorithme de fusion de régions selon le critère de mouvement simplifie fortement l'image. Néanmoins, pourquoi reste-t-il des régions qui n'ont pas fusionné? La raison est que la notion de mouvement n'est pas la même pour nous que pour l'algorithme.

Voyons deux exemples de situations dans lesquelles l'interprétation du mouvement de l'algorithme diffère de celle de l'être humain:

- champ de vision limité

  Le fait que l'image correspond à un champ de vision limité dans l'espace, nous mène à un problème de bords. En effet, le bâtiment du fond est coupé par le cadre de vision ce qui fait que le mouvement de la caméra donne l'impression que la taille de celui-ci a changé. L'algorithme modélise ce mouvement par une réduction d'échelle qui n'est pas la même pour les régions voisines. C'est pourquoi les régions n'ont pas été fusionnées. Peut être que dans ces situations le calcul du vecteur de mouvement associé à l'union de deux régions apporterait une solution. Une piste à suivre pour résoudre ce problème avec un temps de calcul réduit est présenté dans [6].

- zones découvertes

  Quand un objet du premier plan bouge, il découvre une zone derrière lui. Dans cette séquence la zone découverte est le bâtiment et pour l'observateur humain elle bouge de manière cohérente avec le fond. Néanmoins l'algorithme voit une région qui s'élargit. La zone découverte a un mouvement apparent lié au mouvement des objets du premier plan. Pour éviter ce problème, il faudrait une analyse plus approfondie des différents plans de mouvement de l'image. Les lecteurs intéressés par ce sujet peuvent consulter les travaux de Wang et Adelson [57].

Dans cette séquence il y a un mouvement de caméra constant. Ceci, ajouté aux problèmes de bords, fait que nous ne réussissons pas à fusionner complètement le fond, même si en général il est très simplifié.

Pour illustrer le problème lié aux zones découvertes nous pouvons observer que dans l'image 36 nous avons rajouté la région qui est sur la gauche de foreman.

Cette région, complètement incluse dans le cadre de vision, est rajoutée à cause de son mouvement apparent.

Nous avons détecté aussi un problème au niveau de la compensation de mouvement des petites régions et, plus généralement, des régions fines. Quand le vecteur de mouvement d'un pixel pointe au milieu de deux pixels, une interpolation est réalisée pour estimer la valeur du point intermédiaire. Or, s'il s'agit d'une région fine cette interpolation utilise des pixels des régions voisines et le résultat est une mauvaise estimation de la vraie valeur d'un pixel. C'est la raison pour laquelle nous retrouvons régulièrement des petites régions dans la segmentation grossière.

L'image codée correspondante à cette segmentation grossière est sur présentée sur les figures 10.13 et 10.14. La qualité est de 30 dB et le coût de codage est de 46.31 kbits/s. Le détail de ce coût est le suivant:

Information des fusions et des resegmentations	0.65 kbits/s
Information de mouvement	6.10 kbits/s
Information de contour	11.85 kbits/s
Information de texture	27.71 kbits/s
TOTAL	46.31 kbits/s

Table 10.1: Détail du coût de codage de la séquence foreman

Le taux de compression est ainsi de $\frac{3041.28}{46.31} = 65.67$. Ce taux est calculé en tenant compte qu'on ne transmet qu'une image sur 6 de la séquence et qu'en réception on répète la même image 6 fois pour reproduire la séquence à la vitesse correcte. Si on utilise un algorithme d'interpolation performant comme celui de [] ce taux peut être considéré comme étant 6 fois plus grand, c'est-à-dire 394.

10.4.2 Séquence "News"

La séquence est au format QSIF (176 $\times$ 128 pixels) à une fréquence de 30 images par seconde. Nous réalisons un sous-échantillonnage temporel et nous ne traitons qu'une image sur six, c'est-à-dire, cinq images par seconde. La quantité d'information initiale par seconde est donc:

$$5\ \frac{images}{seconde} \times (176 \times 128)\ \frac{pixels}{image} \times 24\ \frac{bits}{pixel} =\ 2703.36\ \frac{Kbits}{seconde}$$

L'image originale sous-échantillonnée temporellement est sur les figures 10.15 et 10.16. Cette séquence est plus simple que la précédente; la caméra est fixe, et il y a deux personnages qui bougent relativement peu. Mais l'information pertinente (yeux, bouche etc) apparaît sous la forme de petits détails, faciles à perdre dans un codage avec pertes. Une difficulté supplémentaire vient du fait qu'il y a un écran sur le fond avec plusieurs changements de scène.

La segmentation fine que nous avons obtenue est sur les figures 10.17 et 10.18. Nous constatons un suivi correct des objets présents dans la scène. Les deux personnages sont suivis d'un bout à l'autre de la séquence et aux moments des changements de scène la nouvelle danseuse est bien segmentée comme nouvelle région alors que la précédente a disparu.

Quand on regarde la segmentation grossière (voir figures 10.19 et 10.20) on observe une forte simplification dès la deuxième image grâce à la simplicité du mouvement de la séquence. Certaines régions sont segmentées sur l'écran du fond. En effet, elles correspondent aux régions qui suivent un mouvement indépendant: les danseurs. Les présentateurs de temps en temps bougent aussi suffisamment pour être resegmentés. Un passage intéressant est celui de l'image 144 à l'image 150 qui correspond à un des changements de scène. Sur l'image 144 pratiquement toute l'image, exceptée la danseuse, est réduite à une seule région. Le changement de scène dans l'écran du fond provoque une mauvaise compensation de mouvement. Cela fait que la plupart des régions de la segmentation fine sont rajoutées à la segmentation grossière. Malgré cela, la fusion par mouvement qui suit le rajout de régions fusionne presque la totalité des régions appartenant à la présentatrice.

Nous pouvons conclure que l'algorithme détecte bien les zones de mouvement.

L'image codée à partir de cette segmentation grossière est présentée sur les figures 10.21 et 10.22. La qualité est de 29.88 dB et le coût de codage est de 28.14 kbits/s. Le détail de ce coût est le suivant:

Information des fusions et des resegmentations	0.32 kbits/s
Information de mouvement	4.01 kbits/s
Information de contour	6.27 kbits/s
Information de texture	17.54 kbits/s
TOTAL	28.14 kbits/s

Table 10.2: Détail du coût de codage de la séquence news

Le taux de compression est ainsi de $\frac{2703.36}{28.14} = 96$. Ce taux est calculé en tenant compte qu'on ne transmet qu'une image sur 6 de la séquence et qu'en réception on répète la même image 6 fois pour reproduire la séquence à la vitesse correcte. Si on utilise un algorithme d'interpolation performant comme celui de [] ce taux peut être considéré comme étant 6 fois plus grand, c'est-à-dire 576.

10.4.3 Séquence "Carphone"

La séquence est au format QCIF (176 × 144 pixels) à une fréquence de 30 images par seconde. Nous réalisons un sous-échantillonnage temporel et nous ne traitons qu'une image sur six, c'est-à-dire, cinq images par seconde. La quantité d'information initiale par seconde est donc:

$$5\ \frac{images}{seconde} \times (176 \times 144)\ \frac{pixels}{image} \times 24\ \frac{bits}{pixel} =\ 3041.28\ \frac{Kbits}{seconde}$$

Cette séquence est de complexité intermédiaire par rapport aux deux précédentes. Il s'agit d'un personnage qui discute de manière animée dans une voiture qui roule. A travers la vitre de la voiture on voit défiler un paysage de campagne.

L'image originale sous-échantillonnée temporellement est sur les figures 10.23 et 10.24 et la segmentation fine correspondante dans les figures 10.25 et 10.26.

Encore une fois, nous sommes face à une segmentation qui tout au long de centaines d'images suit correctement les objets segmentés au début de la séquence.

La segmentation grossière est sur les figures 10.27 et 10.28. Elle contient les objets de mouvement indépendant. Notamment le personnage (ainsi que les régions aux alentours par mouvement apparent). Dans la deuxième partie de la séquence nous détectons aussi un mouvement indépendant sur la vitre de la voiture. En effet, ce mouvement correspond au changement de paysage pendant que la voiture roule. Nous pouvons signaler la simplicité de la segmentation entre les image 144 et 174, qui correspond à un moment de calme dans la séquence.

L'image codée à partir de la segmentation grossière des figures 10.27 et 10.28 est sur les figures 10.29 et 10.30. La qualité est de 30.52 dB et le coût de codage est de 40.85 kbits/s. Le détail de ce coût est le suivant:

Information des fusions et des resegmentations	0.56 kbits/s
Information de mouvement	5.68 kbits/s
Information de contour	9.54 kbits/s
Information de texture	25.07 kbits/s
TOTAL	40.85 kbits/s

Table 10.3: Détail du coût de codage de la séquence carphone

Le taux de compression est ainsi de $\frac{3041.28}{40.85} = 74.44$. Ce taux est calculé en tenant compte qu'on ne transmet qu'une image sur 6 de la séquence et qu'en réception on répète la même image 6 fois pour reproduire la séquence à la vitesse correcte. Si on utilise un algorithme d'interpolation performant comme celui de [] ce taux peut être considéré comme étant 6 fois plus grand, c'est-à-dire 446.

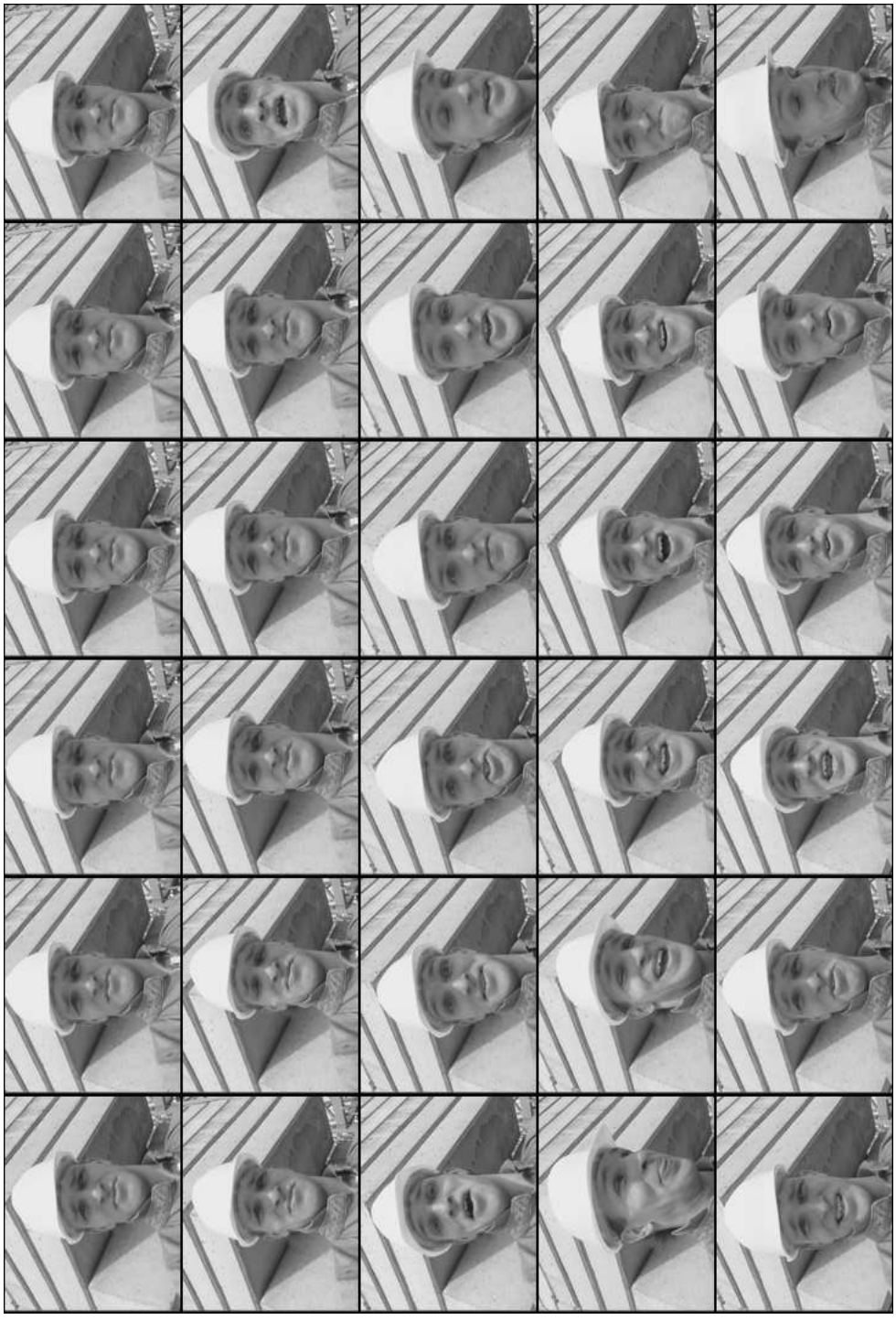


Figure 10.7: Séquence originale sous-échantillonnée dans le temps (une image sur 6). Trames: 0 à 174.

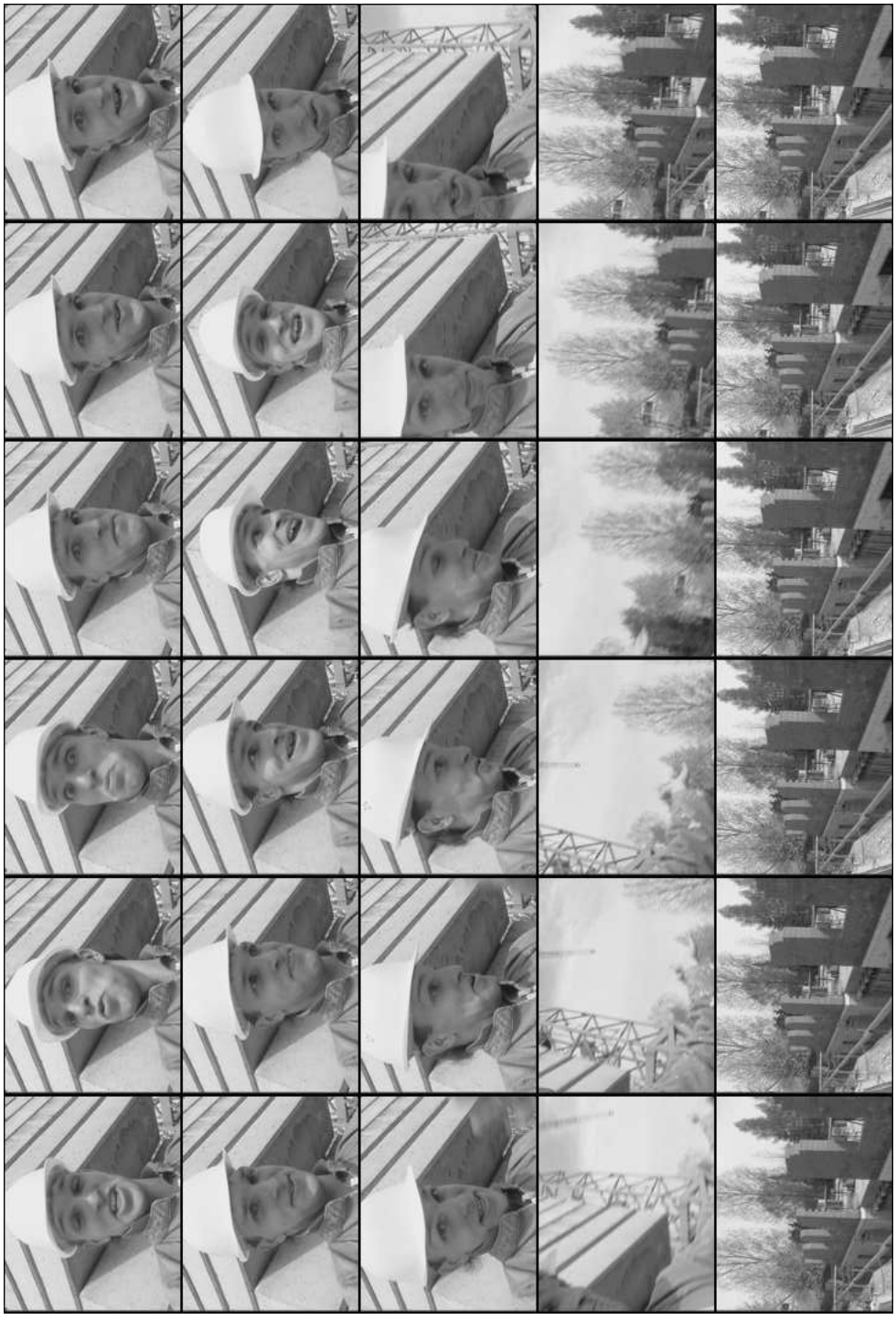


Figure 10.8: Séquence originale sous-échantillonnée dans le temps (une image sur 6). Trames: 180 à 354.

Figure 10.9: Segmentation fine correspondant à la figure10.7. Trames: 0 à 174.

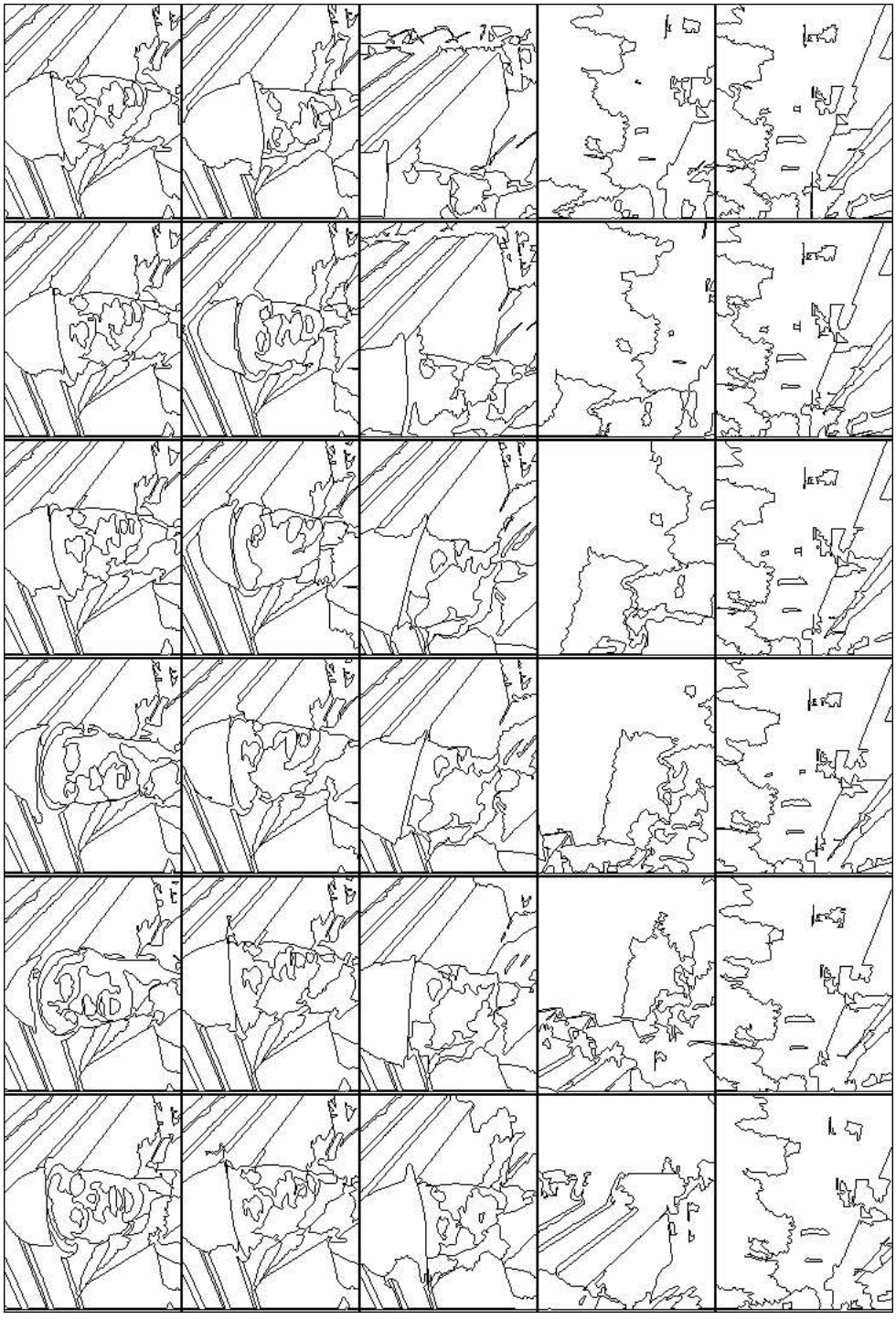


Figure 10.10: Segmentation fine correspondant à la figure 10.8. Trames: 180 à 354.

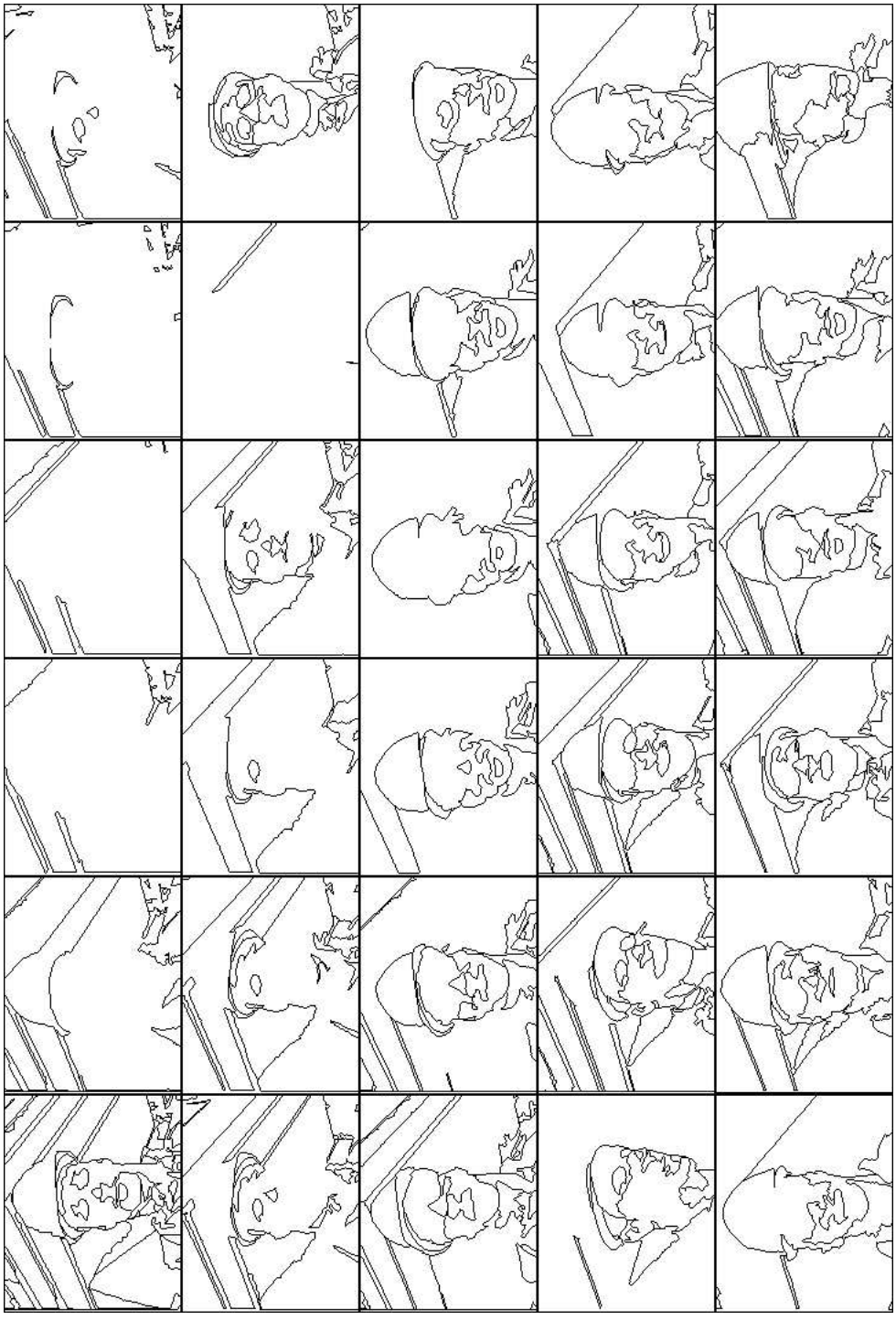


Figure 10.11: Segmentation grossière correspondant à la figure10.7. Trames: 0 à 174.

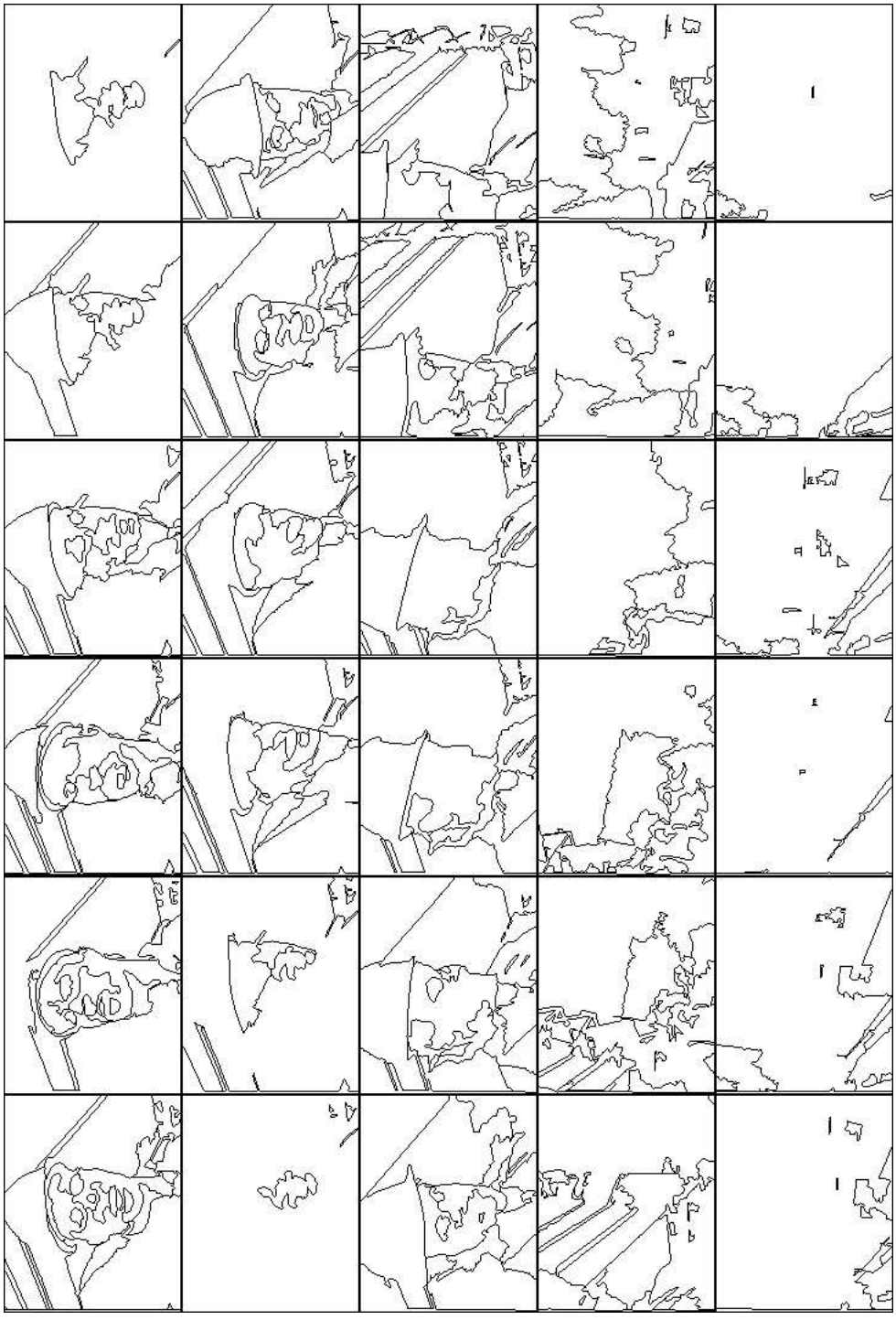


Figure 10.12: Segmentation grossière correspondant à la figure 10.8. Trames: 180 à 354.

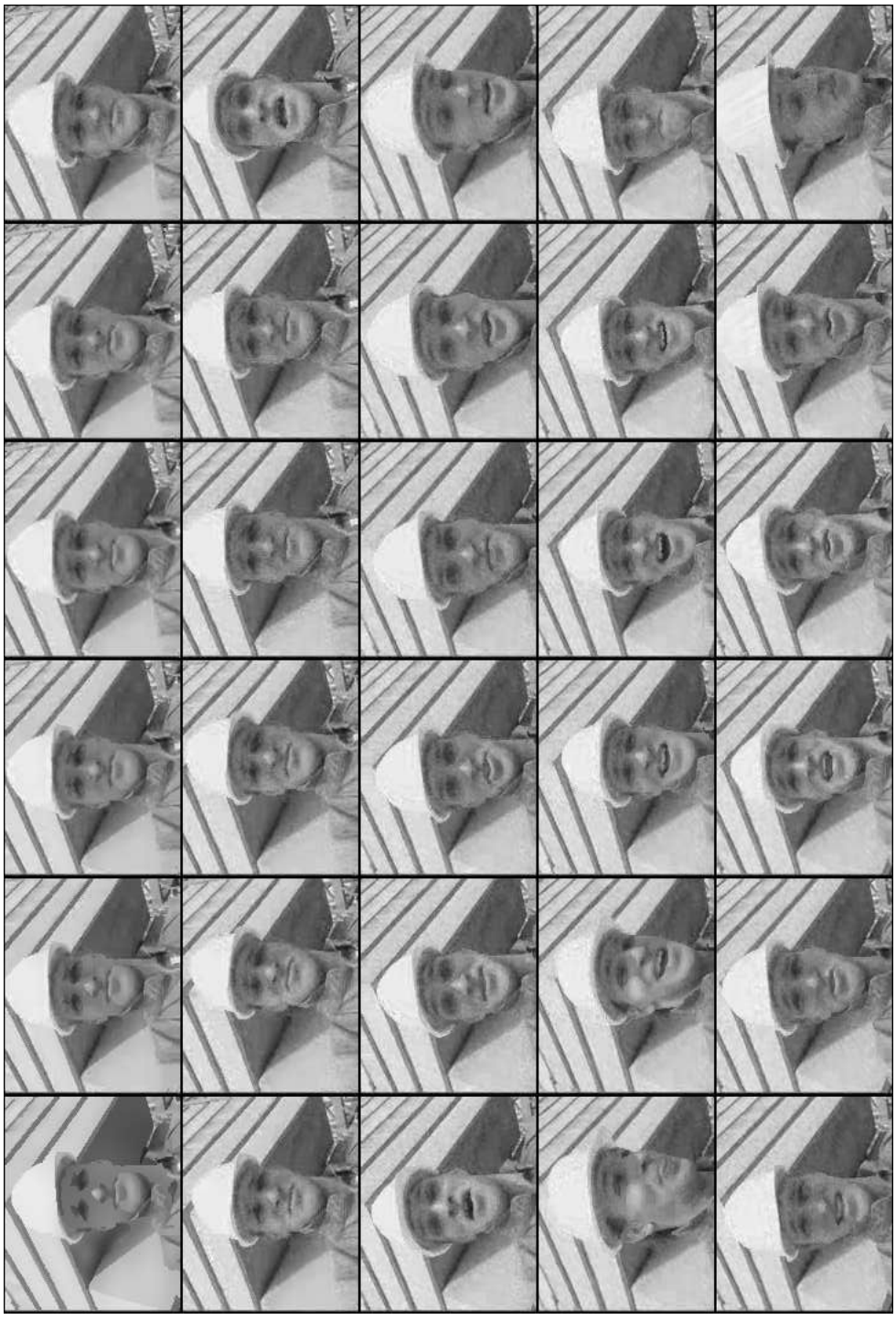


Figure 10.13: Séquence codée. Trames: 0 à 174.

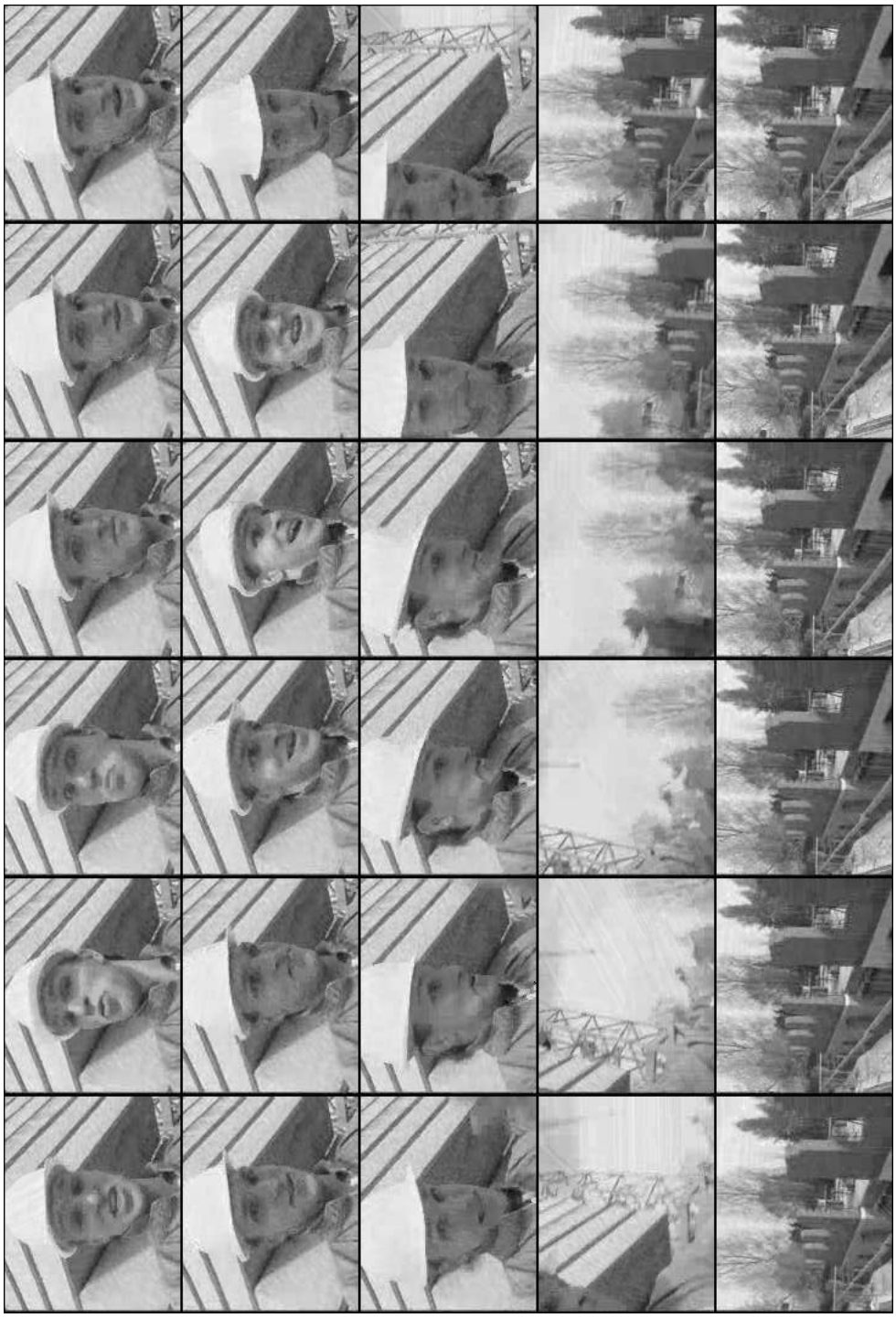


Figure 10.14: Séquence codée. Trames: 180 à 354.

Figure 10.15: Séquence originale sous-échantillonnée dans le temps (une image sur 6). Trames: 0 à 144.

Figure 10.16: Séquence originale sous-échantillonnée dans le temps (une image sur 6). Trames: 150 à 294.

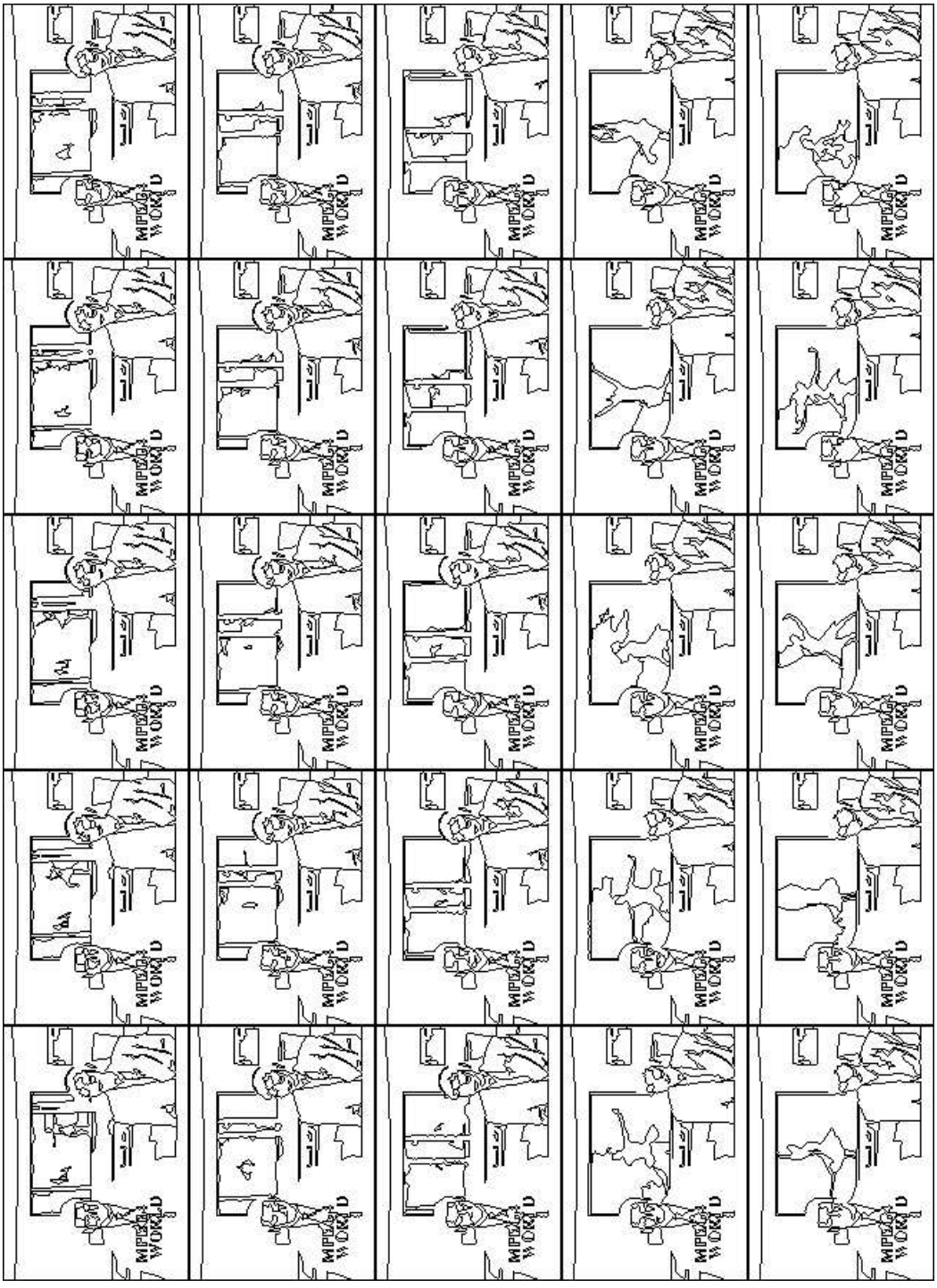


Figure 10.17: Segmentation fine correspondant à la figure10.15. Trames: 0 à 144.

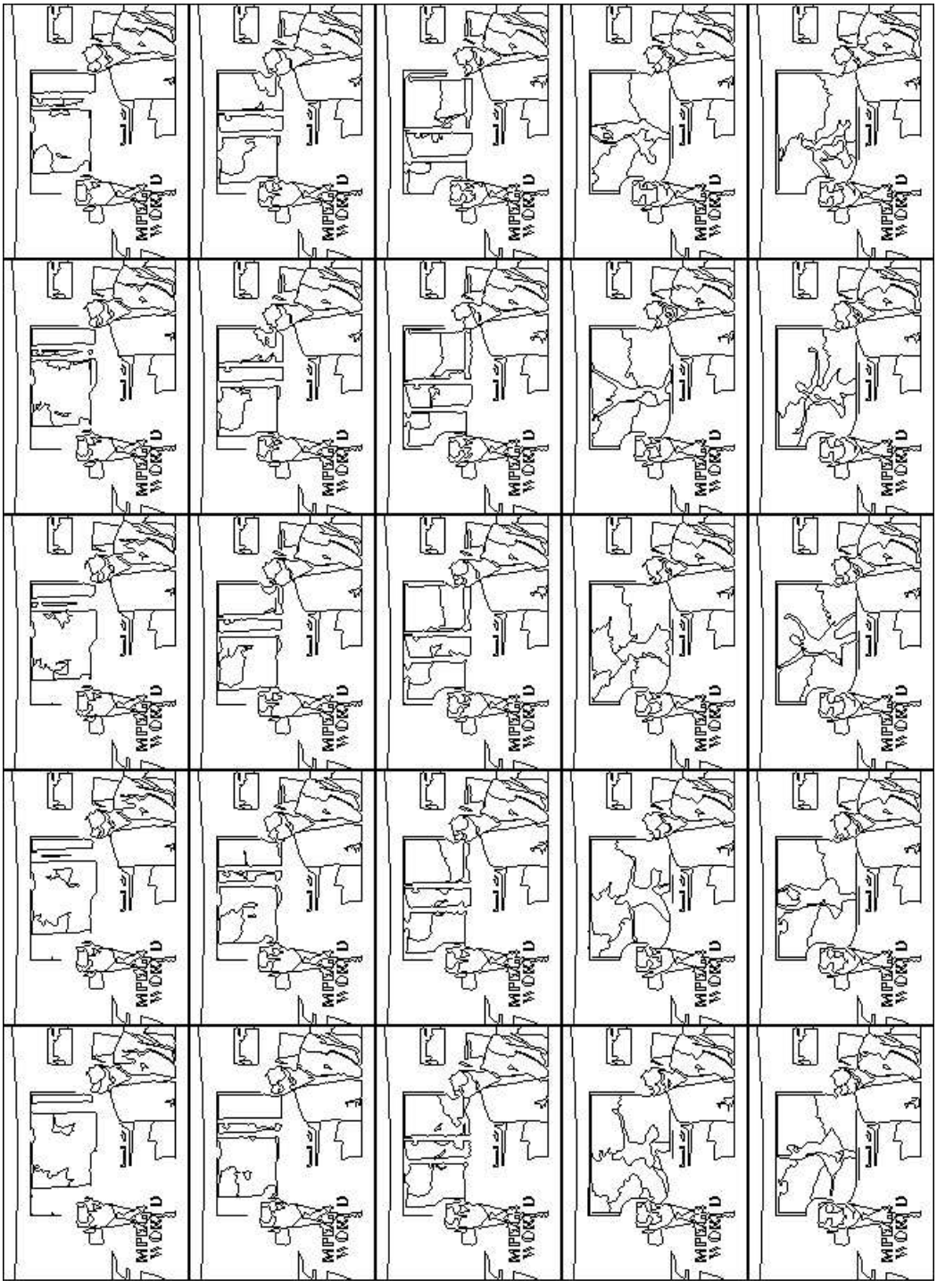


Figure 10.18: Segmentation fine correspondant à la figure 10.16. Trames: 150 à 294.

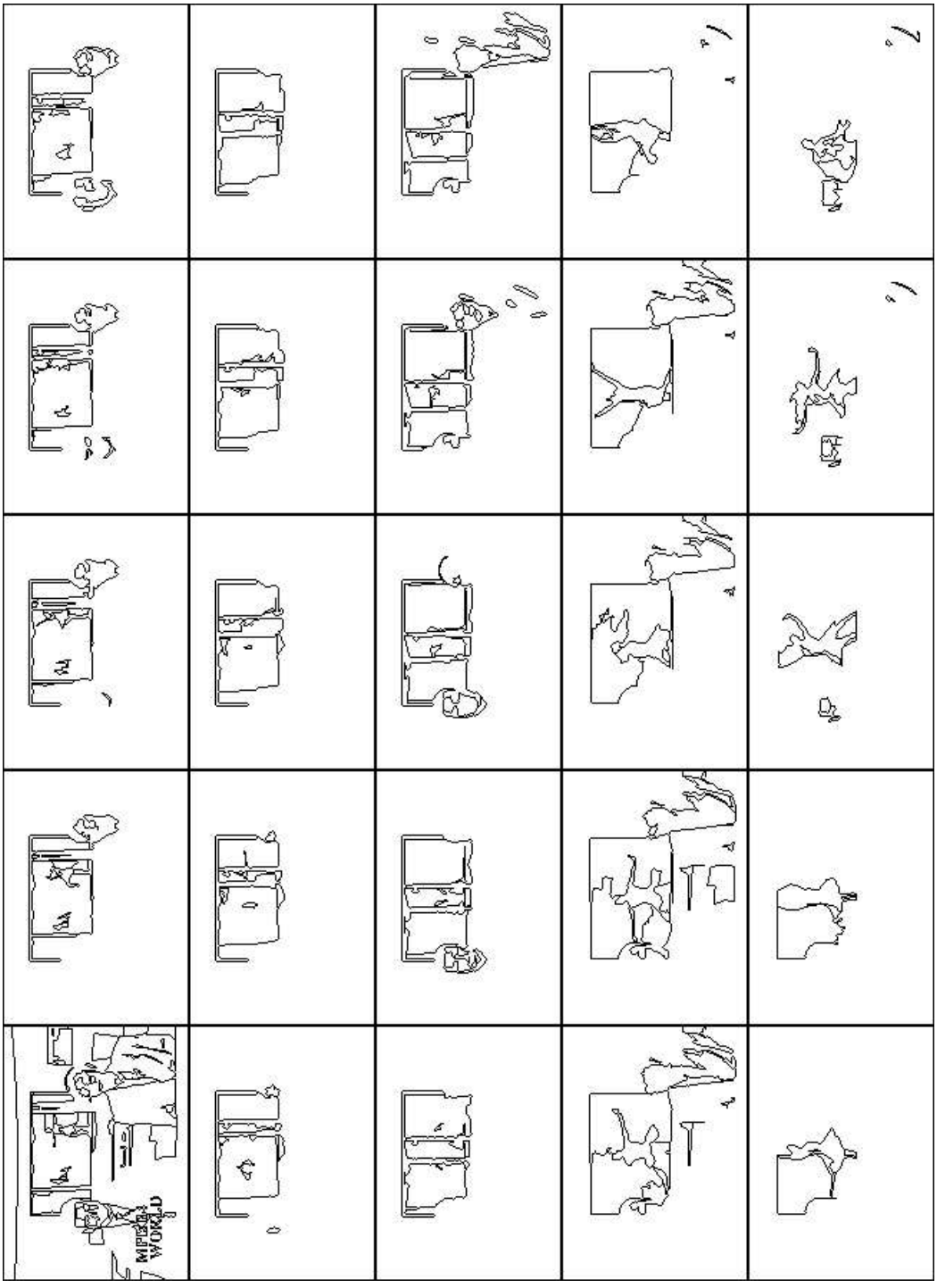


Figure 10.19: Segmentation grossière correspondant à la figure10.15. Trames: 0 à 144.

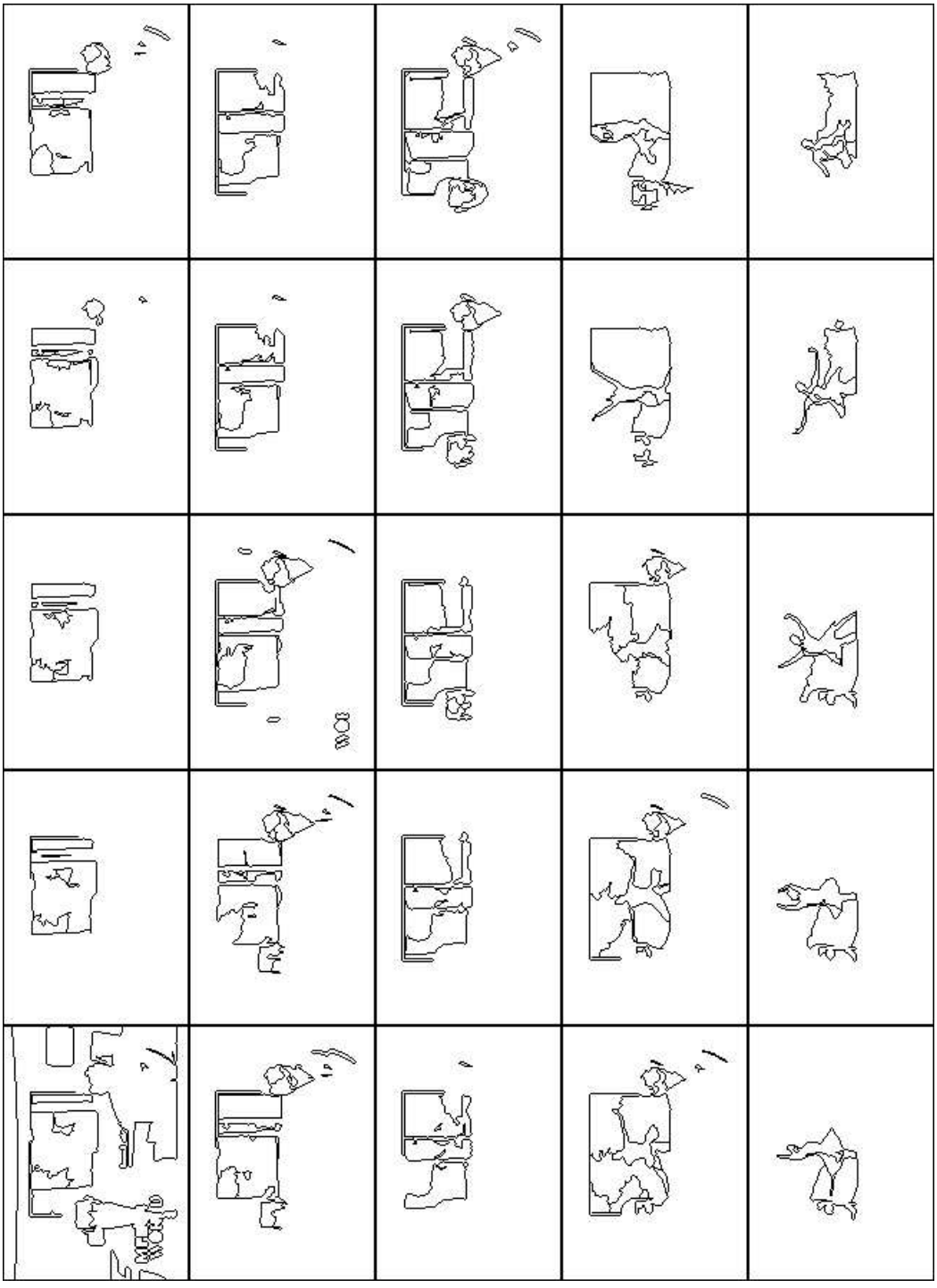


Figure 10.20: Segmentation grossière correspondant à la figure 10.16. Trames: 150 à 294.

Figure 10.21: Séquence codée. Trames: 0 à la 144.

Figure 10.22: Séquence codée. Trames: 150 à 294.

Figure 10.23: Séquence originale sous-échantillonnée dans le temps (une image sur 6). Trames: 0 à la 174.

Figure 10.24: Séquence originale sous-échantillonnée dans le temps (une image sur 6). Trames: 180 à 354.

Figure 10.25: Segmentation fine correspondant à la figure10.23. Trames: 0 à 174.

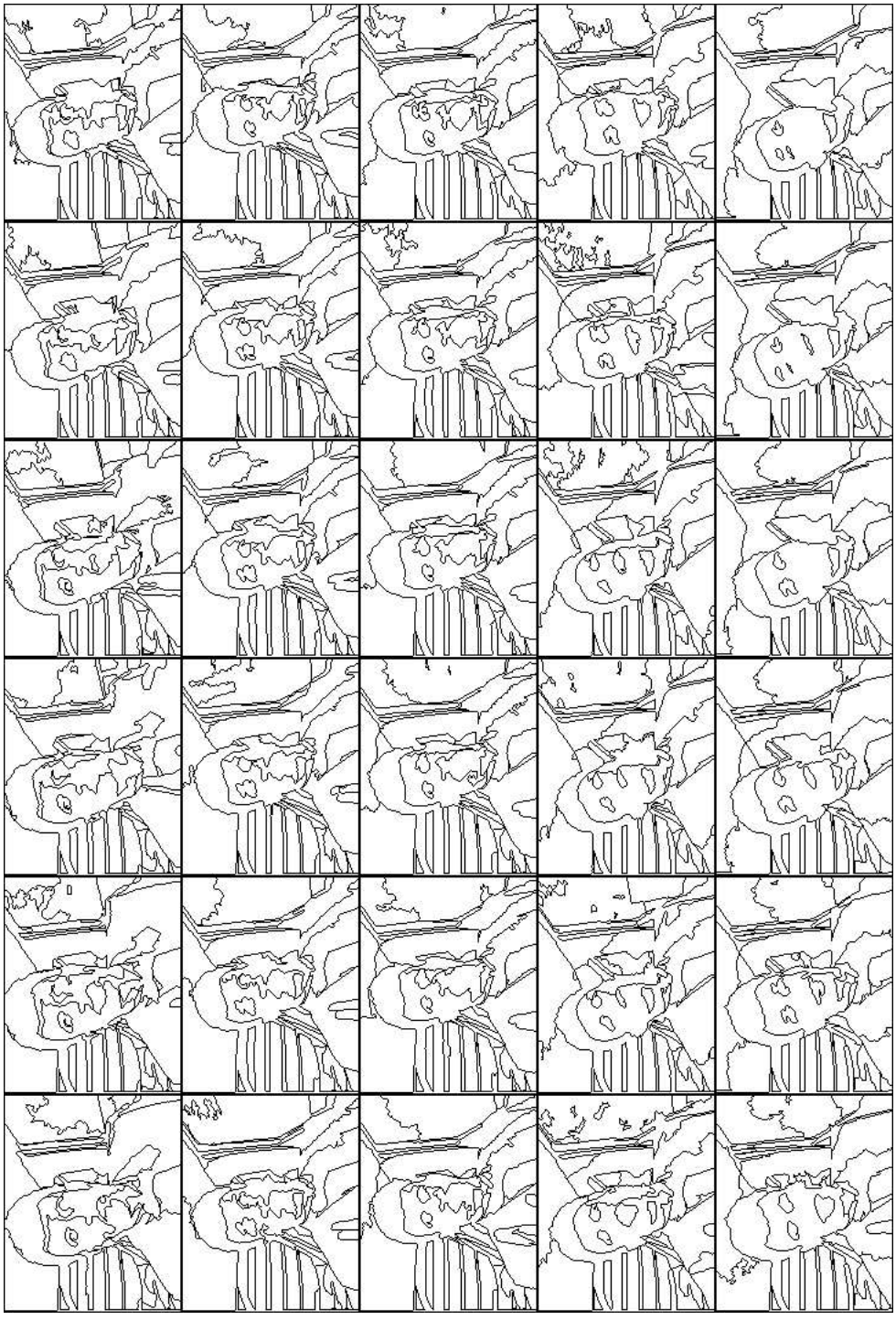


Figure 10.26: Segmentation fine correspondant à la figure 10.24. Trames: 180 à 354.

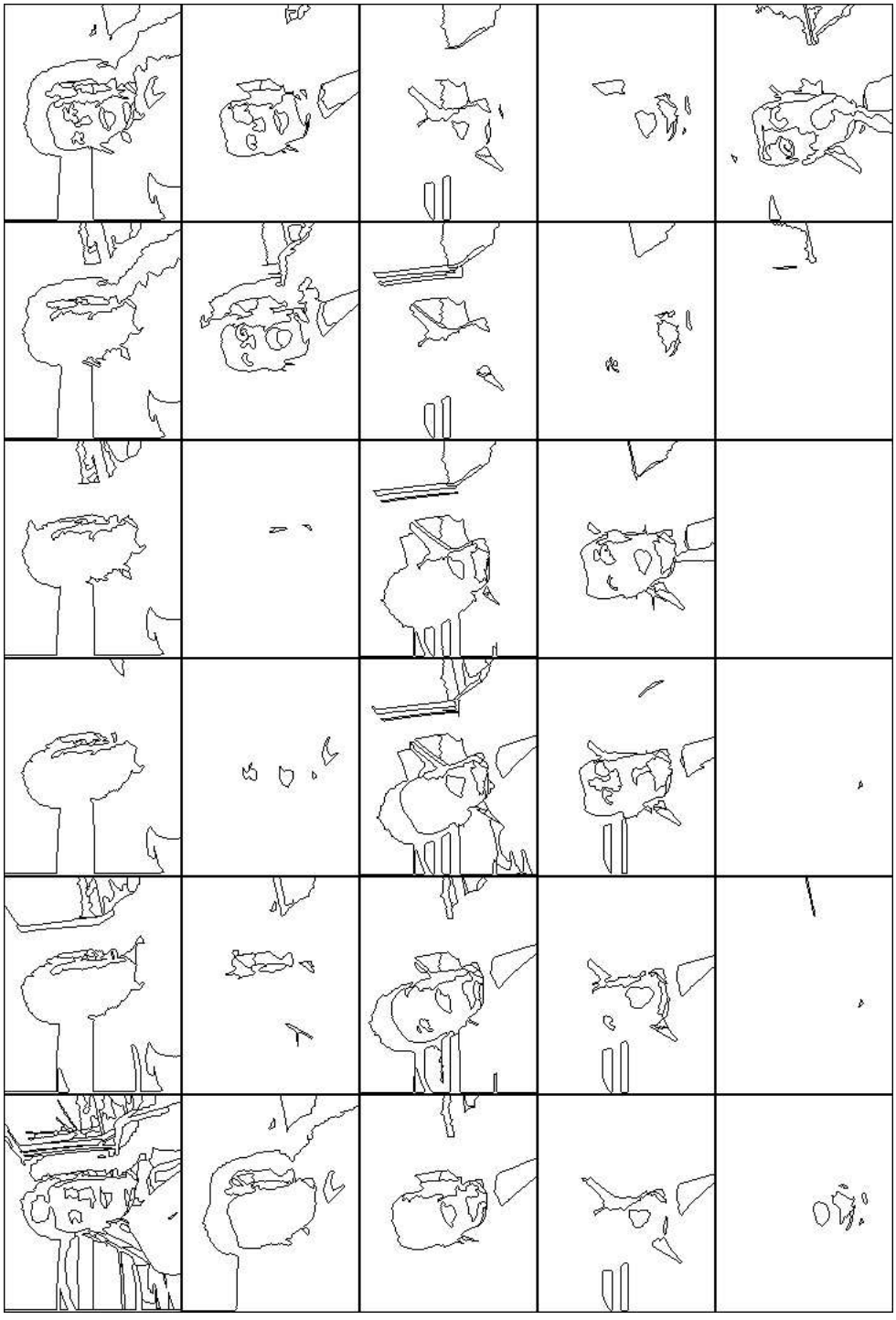


Figure 10.27: Segmentation grossière correspondant à la figure10.23. Trames: 0 à 174.

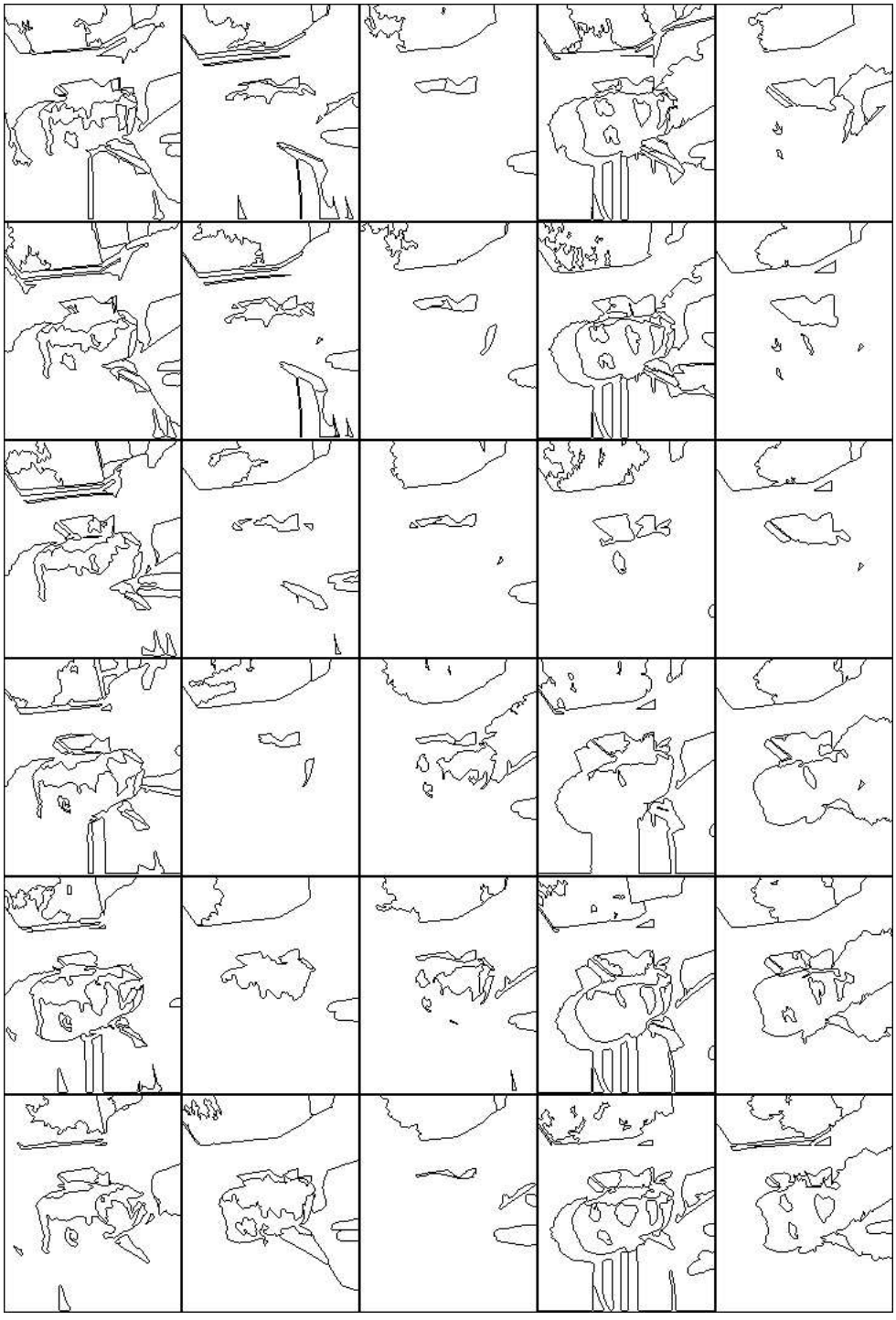


Figure 10.28: Segmentation grossière correspondant à la figure 10.24. Trames: 180 à 354.

Figure 10.29: Séquence codée. Trames: 0 à 174.

Figure 10.30: Séquence codée. Trames: 180 à 354.

10.5 Conclusion

Dans ce chapitre nous avons intégré tous les éléments que nous avons développés dans les chapitres précédents et nous avons obtenu un algorithme complet de segmentation de séquences d'images. Les premiers tests réalisés sur des séquences entières ont révélé des incompatibilités entre les différentes étapes du système de codage. En effet, les fusions par mouvement perturbent le fonctionnement des autres étapes. Les raisons sont les suivantes:

- Elles génèrent des régions homogènes au sens mouvement. Leur caractérisation par niveau de gris ou par texture n'est pas représentative, ce qui pose des problèmes aux étapes initiales de la segmentation.
- Elles produisent des changements importants de forme qui empêchent à l'algorithme de codage de contour d'exploiter la redondance temporelle.
- Les changements importants de forme posent des problèmes aussi pour la compensation de mouvement en mode "restreint".

Malgré ces inconvénients, les fusions par mouvement sont intéressantes du point de vue de l'économie de codage. C'est pourquoi, nous nous sommes intéressés à adapter les différentes étapes affectées pour qu'elles gèrent de manière efficace les régions homogènes au sens mouvement.

Une fois les adaptations réalisées, nous avons finalement abouti à notre algorithme de segmentation de séquences. Ensuite nous l'avons testé sur des séquences entières. Cette fois-ci, puisque toutes les étapes sont cohérentes, nous avons obtenu des résultats satisfaisants.

Chapter 11

Fonctionnalité

11.1 Introduction

De nos jours trois domaines différents de l'industrie, les télécommunications, les ordinateurs multimedia et la télévision échangent des éléments qui historiquement appartenaient à seulement l'un d'entre eux: aujourd'hui les ordinateurs sont munis de vidéo et de son; l'interactivité est introduite dans la télévision; la vidéo et l'interactivité sont entrées dans le monde des télécommunications. Des besoins similaires apparaissent dans les trois domaines et chacun d'eux apporte sa propre solution, souvent incompatible d'un point de vue technologique avec les autres.

MPEG-4 est un standard de codage qui émerge de nos jours, ayant comme but la convergence des applications communes aux trois domaines. Le moyen pour le faire est de définir un cadre de travail flexible, muni d'un ensemble d'outils pour développer diverses fonctionnalités. MPEG-4 prévoit de fournir un standard de codage comprenant des applications basées sur le contenu pour la communication, l'accès et la manipulation de données audio-visuelles digitales. Le standard devra notamment permettre l'accès universel, de hauts taux de compression et l'interactivité.

L'accès universel signifie la disponibilité des données audio-visuelles pour un large éventail de moyens de stockage et de transmission.

Les hauts taux compressions sont nécessaires afin de mieux exploiter les moyens de transmission et de stockage, qui sont physiquement limités.

L'interactivité basée sur le contenu entraîne la capacité d'interagir avec les objets d'une scène. Généralement, cette interaction est limitée à des objets de synthèse. Etendre l'interactivité aux objets naturels est un point important pour les fonctionnalités MPEG-4. Dans ce but, la structure d'un codeur orienté objet est particulièrement attractive. La notion d'objet est intrinsèque au codeur, ce qui rend naturelle l'implémentation de ces fonctionnalités.

Dans ce chapitre nous allons développer une fonctionnalité MPEG-4 qui est le *suivi d'objets* [26].

11.2 Suivi d'objets

Une application de suivi d'objets consiste à repérer un objet dans une scène et à suivre son évolution au cours de la séquence. Un objet est pour nous un masque binaire qui représente la zone d'intérêt à suivre.

11.2.1 Génération du masque

Le masque peut être extrait d'une base de données, pourvu qu'il corresponde à une entité dans la scène. Ensuite la segmentation sera contrainte à ce masque, c'est-à-dire, les régions de la segmentation seront soit complètement à l'intérieur soit complètement à l'extérieur du masque.

Néanmoins le codeur orienté objet offre une possibilité plus intéressante du point de vue de l'interactivité. Il permet à l'utilisateur de choisir de manière interactive le masque d'intérêt. Une fois la première image d'une séquence décodée, l'utilisateur peut dessiner, de manière grossière, à l'aide d'une souris ou d'un crayon électronique, un trait sur la zone qui l'intéresse. Il n'est pas envisageable que l'utilisateur dessine avec précision les contours des objets. Pour cela nous allons nous appuyer sur la segmentation. Nous allons faire le rapport entre le trait dessiné et un ensemble de régions de la segmentation. Nous procédons de la manière suivante:

- nous détectons les régions qui coupent le trait dessiné. Elles appartiennent à la zone d'intérêt.
- les régions qui restent à l'intérieur du masque après l'opération précédente, sont rajoutées au masque (la zone d'intérêt est supposée ne pas comporter de trous).

Voyons un exemple. Prenons l'image de la figure 11.1(a) sur laquelle l'utilisateur a dessiné un trait (en blanc) signalant la zone d'intérêt. La segmentation correspondante est sur la figure 11.1(b). La zone d'intérêt obtenue par union de régions de la segmentation est sur la figure 11.1(c).

Nous voyons que nous sommes capables d'obtenir le contour d'un objet à partir d'un trait grossier dessiné à l'intérieur. Ceci est un point fort vis à vis des fonctionnalités MPEG-4 interactives.

Remarquons l'importance de la segmentation utilisée comme référence pour obtenir le masque à partir du trait. Si la segmentation contient un grand nombre de régions le trait dessiné devra passer près du bord pour obtenir le masque complet de l'objet. Si par contre la segmentation est très grossière, il est possible que l'objet d'intérêt ne soit pas segmenté correctement. C'est pourquoi, il serait intéressant de laisser à l'utilisateur la possibilité de moduler le nombre de régions de la segmentation de référence. De toute façon, la construction d'un masque peut s'effectuer sur plusieurs images: l'utilisateur dessine un premier trait qui définit la première approximation du masque et ensuite, il peut rajouter d'autres traits dans les images qui suivent pour agrandir le masque ou pour le réduire en éliminant certaines régions.

Une autre approche pour générer le masque d'intérêt pourrait être basé sur la notion de marqueur utilisée dans les chapitres précédents. Le problème de cette approche est que l'utilisateur doit non seulement dessiner un trait qui désigne l'objet d'intérêt, mais il doit aussi donner un marqueur "du fond". L'algorithme

de segmentation trouverait ensuite le contour précis entre les deux marqueurs, c'est-à-dire, le contour de l'objet d'intérêt. L'avantage de cette approche est l'indépendance par rapport à une segmentation de référence. Par contre elle a besoin de marquer la zone de non-intérêt, ce qui peut être une notion difficile à expliquer à l'utilisateur.

(a) Trait grossier pour marquer la zone d'intérêt (b) Segmentation (c) Zone d'intérêt

Figure 11.1: Génération de la zone d'intérêt

11.2.2 Suivi du masque

Le masque d'intérêt correspond à l'union de plusieurs régions de la segmentation. Le suivi du masque consiste à suivre chacune des régions qui le constituent.

Un point sur lequel nous avons beaucoup insisté au long de cette thèse est la stabilité temporelle des régions de la segmentation. Elle nous a permis d'obtenir de meilleurs taux de compression. Grâce à cette continuité temporelle, nous sommes maintenant capables de développer une application de suivi d'objets. Comme nous avons vu dans la section 9.1, les régions de la segmentation précédente jouent le rôle de marqueurs du temps courant dans une procédure récursive, ce qui nous permet d'obtenir les objets précédents à leur emplacement courant. Une fois que les régions précédentes ont été projetées au temps courant, nous réalisons les opérations suivantes pour obtenir le masque courant:

- d'abord nous effectuons l'union des régions projetées qui appartenaient au masque au temps précédent
- ensuite nous rajoutons au masque les éventuelles nouvelles régions qui sont apparues à l'intérieur de celui-ci.

11.2.3 Fusions par mouvement et fonctionnalité

Dans les chapitres précédents nous avons adapté la segmentation à un système de codage dans le but d'améliorer l'efficacité du codeur. Pour cela, nous fusionnons les régions qui bougent ensemble. Maintenant, nous envisageons de suivre un masque d'intérêt constitué d'un ensemble de régions. Dans cette situation les régions appartenant au masque ne doivent pas fusionner avec celles qui sont en dehors. Autrement on court le risque de rajouter au masque tous les objets qui bougent comme celui-ci. Imaginons par exemple, une zone d'intérêt qui est momentanément arrêtée par rapport au fond. Si nous permettons les fusions "mixtes" (zones d'intérêt avec des zones de non intérêt) la zone d'intérêt deviendrait l'image entière. C'est pourquoi les fusions par mouvement, dans le

cadre de la fonctionnalité, sont réalisées seulement entre régions de la même classe: soit de la classe zone d'intérêt, soit de la zone de non intérêt.

11.3 Résultats

Cette procédure est appliquée au masque obtenu dans la section 11.2.1 (fig. 11.1(c)). Le résultat est sur la figure 11.2. La figure 11.2(a) montre l'évolution du masque d'intérêt. Nous voyons sur la figure 11.2(b) la bonne correspondance avec le contenu de la scène. Le contour du masque est superposé à l'image originale.

Remarquons que cette idée de zone d'intérêt peut être étendue au suivi de plusieurs zones d'intérêt en même temps. Il suffit de définir un masque qui n'est pas binaire mais qui a un label différent pour chaque zone d'intérêt.

La figure 11.3 montre un autre exemple de suivi d'objets. La zone d'intérêt comprend cette fois ci trois zones distinctes: les deux visages des présentateurs et la danseuse. Nous voyons l'évolution de ces trois zones sur la figure 11.3(a) et la superposition du contour du masque sur l'image originale sur la figure 11.3(b).

11.4 Conclusion

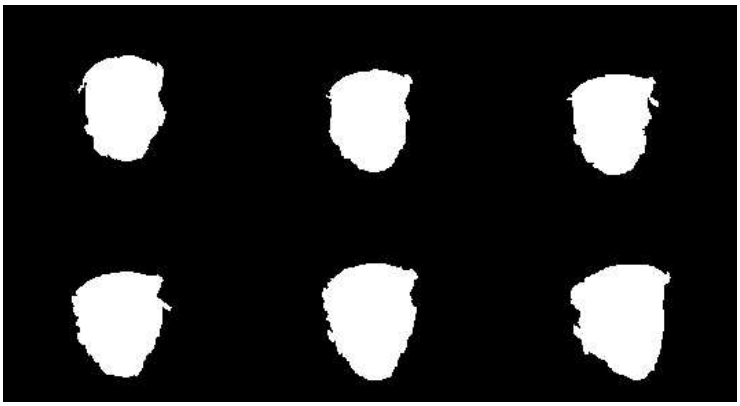
De nos jours, l'audio-visuel gagne de plus en plus de terrain dans différents domaines industriels. L'effort pour développer une application précise est dispersé dans différents domaines et plusieurs solutions, incompatibles entre elles du point de vue technologique, apparaissent pour résoudre le même problème.

Le standard MPEG-4 a pour but d'unifier les efforts et de faire converger les applications communes à différents domaines. MPEG-4 prévoit de fournir un cadre de travail de codage standard et flexible, qui comprenne des applications interactives pour la communication, l'accès et la manipulation de données audio-visuelles.

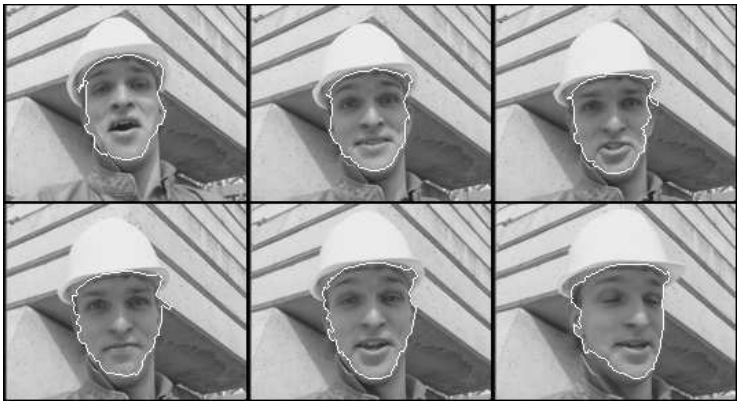
Nous avons montré dans ce chapitre le potentiel d'un système de codage orienté objet pour développer des applications interactives basées sur le contenu. Notamment nous avons montré la possibilité de suivre une zone d'intérêt à travers le temps. Nous avons montré la capacité qu'offre la segmentation pour générer les vrais contours d'une zone d'intérêt à partir d'un simple trait grossier dessiné à la main par l'utilisateur. Ensuite la zone d'intérêt ainsi obtenue peut être suivie tout au long de la séquence, grâce à la procédure de segmentation récursive, qui projette les régions précédentes (permettant aussi l'introduction de nouvelles régions).

Une fois le suivi d'objets introduit dans les algorithmes de codage, d'autres fonctionnalités deviennent aussi possibles. Par exemple, le codage différentiel, qui consiste à coder avec une meilleure qualité la zone d'intérêt que le reste de l'image.

Un codeur orienté objet est particulièrement attractif pour développer des applications "content-based" parce que la notion d'objet est intrinsèque au codeur.

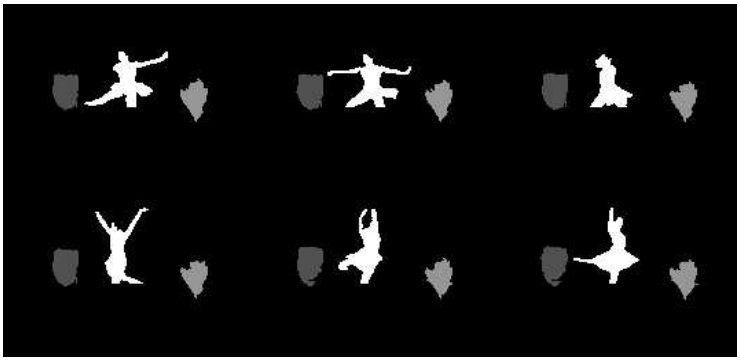


(a) Suivi du masque d'intérêt

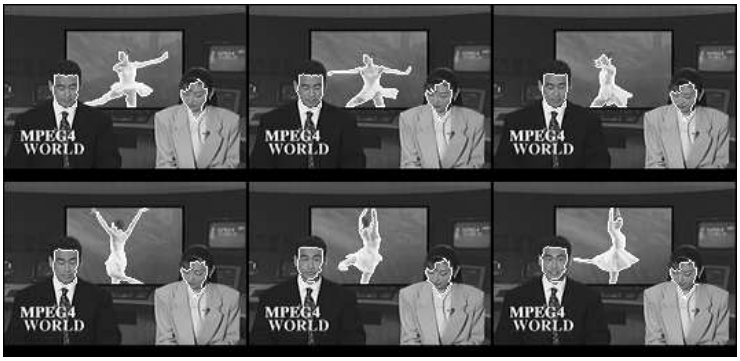


(b) Image originale avec le contour du masque superposé

Figure 11.2: Suivi de foreman par projection.



(a) Suivi du masque d'intérêt



(b) Image originale avec le contour du masque superposé

Figure 11.3: Suivi d'une ballerine par projection.

Chapter 12

conclusion

Les systèmes de codage orientés objet, connus aussi comme systèmes de codage de deuxième génération, analysent le contenu d'une image et la codent en termes d'objets présents dans la scène. Ainsi ces systèmes opèrent en deux étapes: (1) segmentation et (2) codage. La segmentation consiste à diviser l'image en régions qui correspondent autant que possible aux objets présents dans la scène et qui s'emboîtent les unes dans les autres comme les pièces d'un puzzle. Ensuite les contours de la partition obtenue et le contenu des régions sont codés séparément.

Généralement ces deux étapes opèrent de manière indépendante, ce qui peut mener à une situation inefficace, comme par exemple les deux suivantes:

- la partition contient plusieurs régions qui auraient pu être codées comme une seule région, avec un coût de codage plus bas.
- la partition est trop pauvre et le codage de texture ne représente pas correctement le contenu des régions.

Dans cette thèse nous avons abordé le problème de la segmentation en vue du codage. Le travail original a consisté à développer un système de codage cohérent, qui évite les situations inefficaces que nous venons de citer.

La première étape a consisté à appliquer à notre problème l'algorithme classique de segmentation de la Morphologie Mathématique: "la ligne de partage des eaux". Cet algorithme a apporté une solution à de nombreuses applications. Parmi celles-ci, certaines ont abouti à une implémentation hardware en temps réel. Néanmoins son application au codage mène à des difficultés importantes. D'une part, nous pouvons signaler la perte de résolution due à l'utilisation du gradient. Elle nous mène à des segmentations dépourvues de détails fins, dont le codage génère des images de mauvaise qualité visuelle. D'autre part, la caractérisation d'une frontière par la valeur minimale que prend sur elle le gradient nous pose des problèmes au moment de choisir les bons marqueurs ainsi qu'au moment de trouver les contours entre ces marqueurs.

Une fois les difficultés de cette approche analysées, nous avons étudié d'autres méthodes de segmentation qui évitent les problèmes rencontrés. Parmi celles-ci, l'algorithme de fusion de régions s'est révélé le mieux adapté à notre application pour les raisons suivantes:

- il s'intéresse à toute zone plate, indépendamment de sa taille, éliminant ainsi le problème de résolution inhérent au gradient.
- il permet une caractérisation globale des frontières qui est plus pertinente et robuste qu'une caractérisation à partir d'une valeur minimale sur leur longueur.
- il utilise le principe des opérateurs connexes: les zones plates d'une image sont des entités indivisibles. Ainsi, la fusion de zones plates élimine des contours mais n'en génère jamais de nouveaux, ce qui permet de préserver la forme des objets.
- le fait de travailler avec les zones plates d'une image et non pas avec les pixels permet une analyse à plus haut niveau. Une fois les premières fusions réalisées, avec un critère simple, les nouvelles régions obtenues peuvent être munies de caractéristiques plus complexes, telles qu'une représentation de texture ou un vecteur de mouvement.
- il fusionne itérativement les deux régions qui se ressemblent le plus, sans introduire artificiellement un jeu de marqueurs.

Nous avons ainsi implémenté un algorithme de fusion de régions et nous avons développé différents critères de fusion, orientés vers une application de codage.

L'objectif est de fusionner les régions qui peuvent être codées comme une seule région, évitant de fusionner celles qui provoquent une "catastrophe", c'est-à-dire, celles qui génèrent une région dont le codage est de très mauvaise qualité. Pour cela nous avons développé des critères de fusion basés sur la qualité des régions après fusion. On attribue à chaque frontière une évaluation qui correspond à la qualité des régions qu'elle sépare, dans l'hypothèse où la frontière était éliminée. Ensuite les fusions prioritaires sont celles qui, a posteriori, préservent une bonne qualité de représentation. Ainsi, nous améliorons le rapport "qualité / coût" du système de codage: les frontières qui n'apportent pas une amélioration significative à la qualité de l'image sont éliminées, avec la réduction du coût de codage que cela implique. En effet, après une fusion le coût de codage est réduit parce que:

- un seul jeu de paramètres de texture ou un seul vecteur de mouvement est utilisé pour décrire la région résultante de la fusion.
- dans le processus de fusion "disparaissent" des pixels de contour qui autrement auraient dû être codés.

Nous avons créé un cadre flexible pour développer des critères de fusion complexes. L'algorithme, implémenté sur une structure de graphe, est très efficace en lui même. La complexité algorithmique vient des critères de fusion utilisés, qui ont une charge de calcul importante. Néanmoins des stratégies sous-optimales ont été implémentées et validées pour réduire le temps de calcul.

Nous sommes arrivés à un algorithme de segmentation de séquences qui interagit avec les algorithmes de codage. Le rapport "qualité / coût" du système de codage global a ainsi été amélioré.

Par ailleurs nous avons beaucoup insisté sur la stabilité temporelle de la segmentation. Une segmentation stable dans le temps permet de tirer profit de la redondance temporelle et de coder l'évolution de la séquence à travers les vecteurs de mouvement des objets présents dans la scène. Pour obtenir la stabilité temporelle nécessaire, nous avons introduit la notion de marqueur dans l'algorithme de fusion de régions et nous avons considéré les régions de la segmentation précédente comme marqueurs. Cette approche combine les avantages de l'utilisation de marqueurs avec ceux de l'algorithme de fusion de régions: d'une part, nous retrouvons les régions du temps précédent à leur emplacement du temps courant (grâce à l'imposition des régions précédentes comme marqueurs); d'autre part nous introduisons de nouvelles régions quand une zone de la trame en cours ne ressemble à aucune région du passé (l'algorithme de fusion de régions permet la cristallisation de régions qui n'ont pas été marquées).

La stabilité temporelle obtenue nous a permis non seulement de réduire le coût du codage, mais aussi de développer une application de suivi d'objets, qui rentre dans le cadre des nouvelles fonctionnalités de la norme MPEG 4. En effet, un codeur orienté objet a un grand potentiel pour développer des applications "content-based" parce que la notion d'objet est intrinsèque au codeur.

12.1 Perspectives

Nous avons donné un premier pas vers un système de codage cohérent qui segmente en fonction des capacités des algorithmes de codage appliqués par la suite. Nous l'avons testé sur des séquences réelles et nous avons obtenu des résultats encourageants. Néanmoins la porte reste ouverte à de nombreuses améliorations.

L'algorithme de fusion de régions que nous avons développé peut-être interprété en termes d'un arbre de fusions. En effet, nous partons d'une sursegmentation que nous voulons simplifier: les régions de la partition finale seront l'union d'une ou plusieurs régions initiales. Autrement dit, nous pouvons imaginer les régions de la partition initiale comme les feuilles d'un arbre; plusieurs feuilles sont réunies dans une seule branche, plusieurs branches se rattachent à une autre branche plus grande et finalement toutes les branches rencontrent le tronc de l'arbre. La simplification de la segmentation consiste à considérer comme régions non pas les feuilles mais certaines branches de l'arbre (réunion de plusieurs régions initiales). Plus la segmentation est simplifiée, plus les branches se rapprochent du tronc.

L'arbre que nous construisons optimise la qualité de l'image codée. Cela donne déjà de bons résultats. Mais ce n'est pas encore une stratégie optimale. Il est possible qu'entre deux fusions qui vérifient les conditions de qualité demandées nous choisissions celle qui est moins intéressante du point de vue du taux de codage. Imaginons par exemple que les fusions $R_1 - R_2$ et $R_3 - R_4$ vérifient les conditions de qualité. Entre les deux, nous avons donné priorité à celle qui

préserve une meilleure qualité, mais il est possible que ce soit l'autre qui réduise davantage le coût de codage. Il serait intéressant de développer des critères qui tiennent compte non seulement de la qualité de représentation mais aussi du coût de codage.

Le système de codage SESAME [7] va dans cette direction. Ce système génère plusieurs partitions candidates emboîtées les unes dans les autres, ce qui définit aussi un arbre de partitions. Ensuite il choisit parmi les régions des partitions proposées et grâce à une relaxation du Lagrangien *"Distorsion* + λ *coût"* [49, 42] les régions qui:

- constituent une partition
- minimisent la distorsion pour un coût donné

L'arbre que nous produisons dans notre système est beaucoup plus riche que celui de SESAME, mais il est construit en optimisant seulement la qualité de l'image codée. Nous pourrions appliquer à notre arbre une optimisation analogue à celle proposée dans SESAME pour faire intervenir le critère de coût.

Par ailleurs, une solution plus simple qui pourrait être explorée, serait d'utiliser nos algorithmes avec un critère qui combine la distorsion et le coût du codage comme, par exemple, celui de [37]. Cet article considère que le meilleur modèle de codage pour une région donnée est celui qui minimise l'expression:

$$taille\_région \times log\frac{(Error^2)}{taille} + coûtdecodage$$

Une autre amélioration possible concerne le critère de mouvement. Pour éviter les temps d'attente trop longs, nous n'avons pas calculé le vecteur de mouvement associé à une région fusionnée, mais nous avons choisi parmi les vecteurs des régions avant fusion, celui qui modélise au mieux le mouvement global. Cela pourrait être amélioré sans beaucoup augmenter le temps de calcul en utilisant les résultats de [6]. L'algorithme en question consiste à concatener les paramètres de mouvement calculés pour chacune de deux régions. Une factorisation de la matrice ainsi obtenue permet de repérer les coefficients qui ont une influence directe sur l'erreur de prédiction et d'obtenir un nouveau vecteur de mouvement pour la région fusionnée.

Par ailleurs, le choix des fusions par mouvement est fait uniquement à partir de l'information de deux trames: si la compensation de mouvement de plusieurs régions est de bonne qualité à un moment donné, les régions sont fusionnées. Néanmoins il est possible que ces régions se séparent à nouveau dans la trame suivante. Dans ce cas, il aurait peut être été plus économe de ne pas les avoir fusionnées. Dans une application où le délai n'est pas une condition restrictive, comme par exemple le stockage, la décision de fusion peut être prise sur plusieurs images.

Bibliography

[1] S. Beucher. Road segmentation by watershed algorithms. In *PROMETHEUS Workshop, Sophia Antipolis, France*, April 1990.

[2] S. Beucher. *Segmentation d'Images et Morphologie Mathématique*. PhD thesis, E.N.S. des Mines de Paris, 1990.

[3] S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *Proc. Int. Workshop Image Processing, Real-Time Edge and Motion Detection/Estimation*, 1979.

[4] Josep R. Casas and Luis Torres. Coding of details in very low bit-rate video systems. *IEEE Trans. on Cirscuits and Systems for Video Technology*, 4(3):317–327, June 1994.

[5] Josep R. Casas and Luis Torres. Coding of significant features in very low bit-rate video systems. *SPIE Visual Communications and Image Processing*, 2308:73–85, September 1994.

[6] Nokia Research Center. Forward motion compensation. In *SIM(95) 20 COST 211*, Louvain, May 1995.

[7] I. Corset, L. Bouchard, S. Jeannin, P. Salembier, F. Marques, M. Pardàs, R. Morros, F. Meyer, and B. Marcotegui. Technical description of sesame (segmentation-based coding system allowing manipulation of objects). Technical report, ISO/IEC JTC1/SC29/WG11. MPEG 95 / 408, November 1995.

[8] J. Crespo. *Morphological Connected Filters and Intra-Region Smoothing for Image Segmentation*. PhD thesis, School of Electrical Engineering, Georgia Institute of Technology, 1993.

[9] J. Crespo and J. Serra. Morphological pyramids for image coding. In *Proceedings of SPIE, Cambridge*, 1993.

[10] M. Van Droogenbroeck. *Traitement d'Images Numériques au Moyen d'Algorithmes Utilisant la Morphologie Mathématique et la Notion d'Objet: Application au Codage*. PhD thesis, Université Catholique de Louvain and E.N.S. des Mines de Paris, 1994.

[11] M. Eden and M. Kocher. On the performance of contour coding algorithm in the context of image coding. *Part 1: Contour segment coding. EURASIP, Signal Processing, 8:381-386*, 1985.

[12] H. Freeman. On the encoding of arbitrary geometric configurations. *IRE Trans. Electron. Comput.*, EC-10:260–268, June 1985.

[13] Francis Friedlander. *Le Traitement Morphologique d'Images de Cardiologie Nucléaire.* PhD thesis, E.N.S. des Mines de Paris, 1989.

[14] C. Gratin. *De la Représentation des Images au Traitement Morphologique d'Images Tridimensionelles.* PhD thesis, E.N.S. des Mines de Paris, 1993.

[15] M. Grimaud. *La Géodésie Numérique en Morphologie Mathématique. Application a la Détection Automatique de Microcalcifications en Mammographie Numérique.* PhD thesis, E.N.S. des Mines de Paris, 1991.

[16] M. Grimaud. New measure of contrast : dynamics. *Image Algebra and Morphological Processing III, San Diego CA, Proc. SPIE*, 1992.

[17] M. Guedj. Segmentation d'images de milieux poreux en microscopie optique à réflexion. Technical Report N-932, Ecole des Mines de Paris, Centre de Morphologie Mathématique, November 1984.

[18] J.R. Jain and A.K. Jain. Displacement and measurement and its application in interframe coding. *IEEE Trans, COM-29,(12),*, pages 1799–1808, 1981.

[19] T. Jochems. *Morphologie Mathématique appliquée au contrôle industriel de pièces coulées.* PhD thesis, E.N.S. des Mines de Paris, 1994.

[20] J. Klein. *Conception et Réalisation d'une unité logique pour l'analyse quantitative d'images.* PhD thesis, University of Nancy, 1976.

[21] J.C. Klein, F. Lemonnier, and C. Fernández. Morphological codec simulation and demonstrator presentation. Technical report, CEC Deliverable number R2053/CMM/DS/R/012/b1, 1994.

[22] B.M. Kurdy. Algorithme de détection des anneaux dans un composite. Technical Report N-45/87/MM, Ecole des Mines de Paris, Centre de Morphologie Mathématique, June 1987.

[23] C. Lantuéjoul. *La squelettisation et son application aux mesures topologiques de mosaÏques polycristallines.* PhD thesis, École nationale supérieure des mines de Paris, 1978.

[24] S. Laroche. Classification des populations bactériennes dans le yaourt pendant la fermentation. Technical Report N-10/88/MM, Ecole des Mines de Paris, Centre de Morphologie Mathématique, May 1988.

[25] LEP. First results of motion prediction based on a differential estimation method. In *SIM(94) 35 COST 211*, Tampere, June 1994.

[26] F. Marqués, B. Marcotegui, and F. Meyer. Tracking areas of interest for content-based functionalities in segmentation-based video coding. In *International Conference on Acoustics, Speech and Signal Processing, ICASSP'96*, Atlanta, USA, May 1996.

[27] F. Marqués, J. Sauleda, and T.Gassul. Shape and location coding for contour images. In *Picture Coding Symposium, 24-28*, March 1993.

[28] F. Meyer. Quantitative analysis of the chromatin of lymphocytes. an essay on comparative structuralism,. In *Leitz Symposium on Quantitative Morphometry and Image Analysis, Wetzlar, RFA, 24-28*, September 1979.

[29] F. Meyer. Algorithmes séquentiels. In *11ème Colloque GRETSI, Nice*, 1987.

[30] F. Meyer. Algorithmes à base de files d'attente hiérarchique. Technical Report NT-46/90/MM, Ecole des Mines de Paris, Centre de Morphologie Mathématique, September 1990.

[31] F. Meyer. Un algorithme optimal de ligne de partage des eaux. In *Proceedings* $8^{\underline{ème}}$ *Congrès AFCET, Lyon-Villeurbanne*, pages 847–857, 1991.

[32] F. Meyer. Color image segmentation. In *IEEE Fourth International Conference on Image Processing and its Applications*, pages 303–306, April 1992.

[33] F. Meyer. Morphological image segmentation for coding. Technical Report N-01/93/MM, Ecole des Mines de Paris, Centre de Morphologie Mathématique, January 1993.

[34] F. Meyer. Morphological image segmentation for coding. In *Workshop on Mathematical Morphology, Barcelona*, pages 46–51, May 1993.

[35] F. Meyer and S. Beucher. Morphological segmentation. *J. Visual Commun. Image Repres.*, 1(1):21–45, 1990.

[36] O. Monga. *Segmentation d'Images par Croissance Hiérarchique de Régions.* PhD thesis, Université Paris Sud. Centre d'Orsay, 1988.

[37] H. Nicolas and C. Labit. Motion and illumination variation estimation using a hierarchy of models: Application to image sequence coding. *Journal of Visual Communications and Image Representation*, 6(4):303–316, 1995.

[38] M. Pardàs and P. Salembier. Joint region and motion estimation with morphological tools. In *Workshop on Mathematical Morphology and its Applications to Image Processing, Fontainebleau*, September 1994.

[39] M. Pardàs and P. Salembier. Time-recursive segmentation of image sequences. In *EUSIPCO-94*, Edinburgh, September 1994.

[40] M. Pardàs. *Segmentación Morfológica de Secuencias de Imágenes: Aplicación a la Codificación.* PhD thesis, Universidad Politécnica de Cataluña, Barcelona, January 1995.

[41] F. Preteux. Segmentation automatique du corps vertébral à partir d'images scannographiques. Technical Report N-931, Ecole des Mines de Paris, Centre de Morphologie Mathématique, 1984.

[42] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distorsion sense. *IEEE Trans. on Image Processing*, 2(2):160–175, April 1993.

[43] P. Salembier. Morphological multiscale segmentation for image coding. *EURASIP Signal Processing, 359-386*, 38(3):359–386, August 1994.

[44] P. Salembier and J. Serra. Morphological multiscale image segmentation. In *Proceedings of SPIE, Boston*, pages 620–631, 1992.

[45] H. Sanson. Joint estimation and segmentation of motion for video coding at low bit rates. In *COST 211ter European workshop on new techniques for coding of video signals at very low bit rates*, Hanover, December 1993.

[46] J. Serra. *Mathematical Morphology. Volume I.* London: Academic Press, 1982.

[47] J. Serra, editor. *Mathematical Morphology. Volume II: theoretical advances.* London: Academic Press, 1988.

[48] J. Serra and L. Vincent. An overview of morphological filtering. *Circuits Systems Signal Processing*, 11(1):47–108, 1992.

[49] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Trans. Acoust. Speech Signal Processing*, 36:1445–1453, September 1988.

[50] U.Franke. Selective deconvolution: A new approach to extrapolation and spectral analysis of discrete signals. *Int. Conf. on Acousticds, Speech and Signal Processing, IEEE*, pages 30.3.1–30.3.4, May 1987.

[51] C. Vachier. *Extraction de Caractéristiques, Segmentation d'Image et Morphologie Mathématique.* PhD thesis, E.N.S. des Mines de Paris, 1995.

[52] C. Vachier and F. Meyer. Extraction des structures fibreuses en mammographie. application à la détection automatique des opacités du sein. Technical Report NT-42/91/MM, Ecole des Mines de Paris, Centre de Morphologie Mathématique-General Electric C.G.R, 1991.

[53] C. Vachier and F. Meyer. Extinction values: A new measurement of persistence. *IEEE Workshop on Non Linear Signal/Image Processing*, pages 254–257, June 1995.

[54] L. Vincent. *Algorithmes Morphologiques à Base de Files d'Attente et de Lacets. Extension aux Graphes.* PhD thesis, E.N.S. des Mines de Paris, 1990.

[55] L. Vincent. Morphological area openings and closings for grayscale images. In *Proc.NATO Shape in Picture Workshop, Driebergen,*. Springer Verlag, 1992.

[56] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Machine Intell.*, 13, June 1991.

[57] John Wang and Edward Adelson. Representing moving images with layers. *IEEE Trans. on Image Processing*, 3, September 1994.

[58] Xuan Yu. *Vision Dynamique et Morphologie Mathématique. Application à l'Analyse de Scènes Routières.* PhD thesis, E.N.S. des Mines de Paris, 1993.