Mise en correspondance de partitions en vue du suivi d'objets

Cristina Gomila

Mise en Correspondance de Partitions en vue du Suivi d'Objets

---

THÈSE

pour obtenir le grade de

Docteur de l'Ecole des Mines de Paris

Spécialité « Morphologie Mathématique »

présentée et soutenue publiquement
par

Cristina GOMILA

le 12 Septembre 2001

Directeur de thèse : Fernand MEYER

Jury

Jean SERRA	Président
Antonio ALBIOL	Rapporteur
Ferran MARQUÉS	Rapporteur
Edouard FRANÇOIS	Examinateur
Fernand MEYER	Examinateur
Michael WOLLBORN	Examinateur

A mes parents
A mon frère
A Joan

Fuis l'étude dont l'opération meurt avec son opérateur
Léonardo da Vinci

Remerciements

Je voudrias profiter de ces moments de joie (oui, j'ai fini ! !) pour remercier tous ceux qui parmi vous avez fait de ces quatre années de thèse une période chère et inoubliable. Ainsi, je tiens à dire merci à

Fernand Meyer, qui a été pour moi beaucoup plus qu'un directeur de thèse. Il a su m'apprendre à exploiter le meilleur de moi-même : en m'écoutant, en me conseillant, en me montrant comment trouver le bon côté des choses.

Ferran Marqués, ami très cher, à qui je demande pardon pour ma trahison (je vois difficilement la possibilité de faire une deuxième thèse ! !).

Jean Serra, pour sa sympathie vers l'Espagne, pour m'avoir accueillie dans le centre et pour m'avoir permis de l'accompagner lors de ses voyages à Barcelone.

Jean-Claude Klein, pour tous les bons moments passés ensemble lors de notre collaboration dans le cadre du projet M4M.

Sven Bauer, Jens Vollmer, Claudio Marchisio et Edouard François, pour leur amitié, ainsi que tous nos autres partenaires dans le projet M4M, avec qui j'ai passé de très bons moments.

Philippe Salembier, pour son amitié et son soutien inconditionnel (avant, pendant et après la thèse).

Xisca, pour sa présence tout au long de ces années. Nous avons partagé des expériences inoubliables ainsi que beaucoup de bavardages.

Etienne et Bea, pour leur vitalité (les parties de pétanque, les soirées chez eux, les promenades en forêt et ceci ne sont que quelques exemples).

Michel, pour sa disponibilité et sa débrouillardise informatique, qui m'ont sauvée à de nombreuses reprises et empêchée de tout recommencer.

Jesús, avec qui j'ai eu le plaisir de partager notre petit coin du grenier et beaucoup de pensées.

Catherine, pour sa joie, pour sa sympathie, pour sa disponibilité, et pour une liste interminable de bonnes qualités.

Marc, pour sa convivialité et sa bonne humeur, même dans les jours les plus froids et pluvieux.

Tous les amis, que ce soient des thésards, des chercheurs ou des collègues de travail, que j'ai eu la chance de rencontrer pendant ces années. Faisant partie de cette longue liste, je voudrais mentionner ici Lluis Garrido, Liliane Pipault, Lothar Bergen et Kiki Wignton, Claire-Helène Demarty, Valéry Risson, Christophe Gratin, Allan Hanbury, Thomas Walter, Christophe Bernard, Eva Dejnozkova et Petr Dokladal, dans la chronologie de ma mémoire.

Finalement j'ai laissé en dernier Joan, car mes remerciements à lui seul peuvent remplir des pages entières. Pour tous et chacun des instants que nous avons passé ensemble :

MERCI ! ! ! !

Résumé

Dans le domaine des applications multimédia, les futurs standards vont permettre de créer de nouvelles voies de communication, d'accès et de manipulation de l'information audiovisuelle qui vont bien au-delà de la simple compression à laquelle se limitaient les standards de codage précédents. Parmi les nouvelles fonctionnalités, il est espéré que l'utilisateur pourra avoir accès au contenu des images par édition et manipulation des objets présents. Néanmoins, la standardisation ne couvre que la représentation et le codage de ces objets, en laissant ouvert un large champ de développement pour ce qui concerne la problématique liée à leur extraction et à leur suivi lorsqu'ils évoluent au long d'une séquence vidéo. C'est précisément sur ce point que porte cette thèse.

Dans un premier temps, nous avons procédé à l'étude et à la mise au point d'algorithmes de filtrage et de segmentation à caractère générique, car ces outils sont à la base de tout système d'analyse du contenu d'une image ou d'une séquence. Plus concrètement, nous avons étudié en détail une nouvelle classe de filtres morphologiques connus sous le nom de nivellements ainsi qu'une variation des algorithmes de segmentation basée sur l'inondation contrainte d'une image gradient. Les techniques de segmentation ont pour but de produire une partition de l'image aussi proche que possible de celle faite par l'œil humain, en vue de la reconnaissance postérieure des objets. Néanmoins, dans la plupart des cas, cette dernière tâche ne peut être faite que par interaction humaine et, pourtant, lorsqu'on veut retrouver un objet dans une large collection d'images, ou suivre son évolution au long d'une séquence, la surveillance de chacune des partitions devient impossible. S'impose alors le développement d'algorithmes de mise en correspondance capables de propager l'information dans une série d'images, en limitant l'interaction humaine à une seule étape d'initialisation.

En faisant le passage des images fixes aux séquences, la partie centrale de cette thèse est consacrée à l'étude du problème de la mise en correspondance de partitions. La méthode que nous avons développée, nommée technique de Segmentation et Appariement Conjoint (SAC), peut être définie comme étant de nature hybride. Elle combine des algorithmes classiques de mise en correspondance de graphes avec de nouvelles techniques d'édition, basées sur les hiérarchies de partitions fournies par la segmentation morphologique. Cette combinaison a donné lieu à un algorithme très robuste, malgré l'instabilité typiquement associée aux processus de segmentation. La segmentation de deux images peut différer fortement si on la considère du seul point de vue d'une partition unique ; néanmoins nous avons montré qu'elle est beaucoup plus stable si on considère des hiérarchies de partitions emboîtées, dans lesquelles tous

les contours présents apparaissent, chacun avec une valuation indiquant sa force. Les résultats obtenus par la technique SAC ont fait d'elle une approche très prometteuse. Souple et puissante, elle est capable de reconnaître un objet lorsqu'il réapparaît après occultation grâce à la gestion d'un graphe de mémoire. Bien que nous nous soyons intéressés tout particulièrement à la problématique du suivi, les algorithmes mis au point ont un champ d'application beaucoup plus vaste dans le domaine de l'indexation, en particulier pour la recherche d'objets dans une base de données d'images ou de séquences.

Finalement, dans le cadre du projet européen M4M (MPEG f(o)ur mobiles) nous avons abordé la mise en œuvre d'un démonstrateur de segmentation en temps réel capable de détecter, segmenter et suivre un personnage dans des séquences de vidéophonie. Dans le cadre de cette application, la contrainte du temps réel est devenue le grand défi à surmonter, en nous obligeant à simplifier et à optimiser nos algorithmes. L'intérêt principal en termes des nouveaux services est double : d'un côté le détourage automatique du locuteur permettrait d'adapter le codage à l'objet, économisant du débit sans perte de qualité sur les régions d'intérêt ; d'un autre côté il permettrait de faire l'édition personnalisée des séquences en changeant la composition de la scène, par exemple en introduisant un nouveau fond, ou en disposant plusieurs locuteurs dans une salle de conférence virtuelle.

Abstract

In the field of multimedia applications, the incoming standards promote the creation of new ways of communication, access and manipulation of audiovisual information that go far beyond the plain compression obtained by the preceding coding norms. Among the new functionalities, it is expected that the user will be allowed to access the image content by editing and manipulating the objects of interest. Nevertheless, standards are restricted to object representation and coding, leaving opened a large field of development concerning the problem of object extraction and tracking when they move along a video sequence.

In a first step, we have proceeded to the study and fine tuning of widespread applicated algorithms for image filtering and segmentation, being these tools at the basis of all content-based image and video analysis systems. More particularly, we have focused on a novel class of morphological filters known as levelings, as well as on a variant of the segmentation algorithms based on the constrained flooding of a gradient image. Segmentation techniques aim at yielding a partition image as close as possible to the one produced by the human eye, with a view to the later object recognition. Nevertheless, in most cases this last task needs human interaction. However, when we would like to retrieve an object from large collection of images, or when we would like to track an object through a long sequence, the surveillance of each image becomes infeasible. To face these situations, the development of matching algorithms able to propagate the information through a series of images become essential, human interaction being limited to a initialization step.

Going from still images to sequences, the core of this thesis is devoted to the study of the partition matching problem. The method we have developed, named Joint Segmentation and Matching technique (JSM), can be defined as being of hybrid nature. It combines classical algorithms of graph matching with new editing techniques based on the hierarchy of partitions resulting from morphological segmentation. This mix provides a very robust algorithm, in spite of the instability classically associated to the segmentation processes. The result of segmenting two images can strongly differ if the segmentation process produces a single partition image, however we have shown that results are much more stable when producing a hierarchy of nested partitions, in which all contours are present and ranked through a weighted value. The JSM technique is considered a very promising approach according to the obtained results. Being flexible and powerful, it allows the recognition of an object when it reappears after occlusion thanks to the management of a memory graph. Although we have particularly focused our interest on the tracking problem, the developed algorithms can be extended to

a large field of applications, being specially suited to perform object retrieval from image or video sequences databases.

Finally, in the framework of the European project M4M (MPEG f(o)ur mobiles), we have focused on the development and implementation of a real-time demonstrator for detecting, segmenting and tracking the speaker in videophone sequences. In the view of this application, the real-time constraint has become the greatest challenge to overcome, forcing us to simplify and optimize our algorithms. The main interest in terms of new services is twofold : on one hand the automatic segmentation of the speaker permits the object-based coding, reducing the bitrate without loss of quality on the regions of interest ; on the other hand, it allows the user to edit the sequences by changing the scene composition, for example by introducing a new background, or grouping several speakers in a virtual meeting room.

Table des matières

Chapitre 1

Introduction et Motivation

Dans ce premier chapitre nous allons présenter le cadre de travail qui a entouré nos recherches : nous parlons de la norme MPEG4 et plus concrètement, du projet européen M4M (MPEG fo(u)r Mobiles) auquel nous avons participé. Nous voudrions également à ce stade donner un aperçu de la structure du document en suivant le parcours de nos recherches.

1.1 La norme MPEG4

Dans le domaine des applications multimédia, le nouveau standard MPEG4 [77] va permettre de créer de nouvelles voies de communication, d'accès et de manipulation de l'information audiovisuelle qui vont bien au-delà de la simple compression obtenue par les standards précédents MPEG1 et MPEG2 [68]. Principalement les nouvelles fonctionnalités de MPEG4 se basent sur trois atouts :

Flexibilité : la rapide évolution des nouvelles technologies a mis en évidence le besoin de créer un standard capable de suivre les derniers développements, aussi bien au niveau hardware qu'au niveau méthodologique. Ainsi, MPEG4 a été conçu sur la base d'une syntaxe (MSDL, MPEG4 Syntactic Description Language) permettant de configurer de manière flexible émetteur et récepteur, et permettant l'interactivité de l'utilisateur pour gérer la composition de la scène.

Intégration : la dernière révolution des télécommunications a eu lieu principalement grâce à la disponibilité d'une plus large bande passante. Ainsi, dans un domaine en forte croissance qui échange de plus en plus de donnés, MPEG4 apparaît comme une plate-forme capable d'offrir une solution à la nouvelle demande de services intégrés. Cela nous mène vers la convergence des systèmes de télécommunications, internet, jeux vidéo, TV, services de bases de données, ...

Manipulation du contenu : MPEG4 autorise plus que la simple compression des images, en permettant au récepteur d'avoir accès au contenu de la scène. Pour cela l'image a dû être analysée et décomposée en VOPs (Video Object Planes). Chacun des VOPs extrait un objet significatif dans l'ensemble de l'image et suit son évolution tout au long de la séquence. Cela

permet la manipulation d'un objet de façon indépendante des autres VOPs de l'image. Par exemple, une scène de la vidéophonie peut être composée de deux VOPs, le premier contenant le fond de la scène, le deuxième contenant le personnage. Il est ainsi possible de manipuler le contenu de la scène, en changeant par exemple de fond.

Nous avons illustré dans la Figure 1.1 quelques-unes des nouvelles fonctionnalités MPEG4. Cela est un exemple de la façon dont l'utilisateur peut éditer et manipuler le contenu de l'image à partir des VOPs : intégrant des objets synthétiques et naturels, du texte, de graphiques, du son, ...

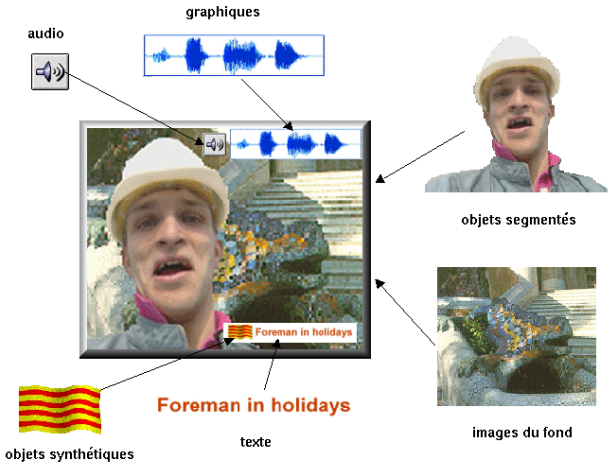


FIG. 1.1 – Exemple des fonctionnalités MPEG4.

Néanmoins, la norme MPEG4 ne standardise que la représentation et le codage des VOPs. Ainsi, les techniques de segmentation et de suivi d'objets utilisées en périphérie du codeur MPEG4 ne font pas partie de la norme. En cela le standard fait preuve de flexibilité car il n'existe aucune méthode universelle de segmentation et d'analyse de scènes permettant de construire des VOPs en toute circonstance. Il laisse ainsi ouvert un large champ de développement : l'analyse et la segmentation de scènes, la richesse des interactions possibles avec le contenu de l'image, ou même la capacité du système de suivre les objets d'intérêt au long d'une séquence. C'est précisément sur ce dernier point que portera notre étude.

1.2 Le projet européen M4M (MPEG fo(u)r Mobiles)

Après les succès commerciaux des standards MPEG1 et MPEG2, avec des millions de chips décodeurs vendus, il est espéré que dans un futur déjà proche, MPEG4 devienne progressivement plus important dans tous les aspects concernant les systèmes de diffusion et de communication multimédia. Face à cette nouvelle demande, le projet européen M4M (MEDEA A116) [58] a été conçu pour créer des circuits intégrées dans le domaine des communications mobiles en temps réel. Pour cela un vaste domaine de travail a dû être couvert. Dans un premier temps on a focalisé tous les efforts sur le développement de nouveaux algorithmes,

capables de faire face à la contrainte du temps réel. Ensuite on a ouvert une étape d'évaluation de ces techniques afin de mieux connaître les performances acquises. Finalement les études théoriques ont dû transférer leur savoir-faire aux développeurs de systèmes, qui doivent à leur tour optimiser les logiciels et développer les architectures électroniques dédiées à faible consommation. Cette interaction a donné lieu à des solutions conjointes, d'esprit pratique ; elle a suscité des discussions sur la viabilité des systèmes, et surtout a permis d'ouvrir les portes aux nouveaux systèmes commerciaux.

Dans ce cadre, nous avons centré nos recherches en matière de filtrage et de segmentation. Cela contribuera de manière tout à fait significative au projet en donnant des solutions pour la partie non standardisée de MPEG4 et plus particulièrement sur les problèmes relatifs à l'étude et à la mis au point d'un encodeur vidéo temps réel pour des applications de vidéophonie.

1.3 Plan de la thèse

Le propos de cette thèse est de développer des outils de segmentation permettant de suivre des objets dans des séquences vidéo. Dans cette ligne, nous avons procédé en trois étapes : en premier nous avons étudié le processus de segmentation pour des images fixes sous un angle généraliste, afin de mettre au point des outils souples et puissants ; ensuite, nous avons considéré le problème de la mise en correspondance de partitions, faisant le passage des images aux séquences ; et finalement, nous avons abordé la mise en œuvre d'un démonstrateur de segmentation en temps réel pour des applications de vidéophonie, valorisant nos algorithmes dans un cadre industriel.

1.3.1 Partie I : Le Paradigme de la Segmentation

La segmentation d'une scène peut être vue comme la recherche d'une partition de l'espace la plus proche possible de la reconnaissance visuelle des objets. Cependant, dans le domaine de l'imagerie numérique l'expérience nous montre qu'il n'existe pas une méthode de segmentation valable en toute circonstance. Ainsi, le chemin qui nous mène du pixel à l'objet reste dans la plupart de cas un défi. C'est ce que nous avons appelé le paradigme de la segmentation.

Dans ce vaste chantier nous avons laissé l'intuition guider l'ordre de nos recherches. Pour décrire l'approche dans le grandes lignes nous verrons comme les filtres, en simplifiant l'image, font le passage du *pixel* aux *zones plates*. Puis, nous quitterons le plus rapidement possible l'échelle du pixel pour aborder l'échelle de la *région*. Ainsi, les algorithmes de segmentation proprement dits ont en charge la localisation précise des contours des objets. Notre but final étant d'obtenir une *représentation hiérarchique du contenu* de l'image à différents niveaux de détail.

1.3.2 Partie II : La Mise en Correspondance

Cette deuxième partie du document est consacrée au problème de la *mise en correspondance de partitions* en vue du suivi d'objets, sujet principal de cette thèse. Le problème de

la mise en correspondance, comme celui de la segmentation, est l'un des problèmes les plus difficiles auxquelles l'analyse d'images a à résoudre, et bien sûr il est loin d'être résolu.

En l'égard à ces considérations, nous avons abordé le sujet ayant à l'esprit le développement d'une technique aussi souple que possible, sur laquelle on puisse bâtir ensuite des applications variées. Nous avons adopté les *graphes* comme éléments de support de nos algorithmes, ceux-ci étant d'excellents outils de synthèse. Ainsi, dans le cadre de notre approche, la mise en correspondance de partitions revient à un problème de *mise en correspondance de graphes*.

La méthode que nous avons développée peut être définie comme étant de nature hybride, car elle combine des algorithmes classiques de mise en correspondance de graphes avec de nouvelles techniques d'édition, basées sur les hiérarchies de partitions fournies par nos outils de segmentation.

Dans une étude comparative nous étudierons la performance de cette nouvelle approche par rapport à d'autres techniques de suivi d'objets plus classiques. Du point de vue de la robustesse, nos algorithmes vont surmonter quelques uns des inconvénients les plus contraignants concernant l'appariement de partitions, comme l'apparition ou disparition d'objets. Par ailleurs, leur extension au domaine de la recherche d'objets dans des bases de données serait aussi envisageable, et reste l'une des pistes ouvertes pour une poursuite éventuelle de ce travail.

1.3.3 Partie III : Mise en œuvre d'une Application

Dans la dernière partie de cet ouvrage, nous avons abordé un problème bien différent, car il concerne la mise en œuvre d'une application en temps réel dans le cadre du projet M4M. Ici nous devions être capable de détecter, segmenter et suivre un personnage dans des séquences de vidéophonie. L'intérêt principal en termes de nouveaux services est de pouvoir transmettre de manière plus fidèle une région d'intérêt que le reste de l'image ; ou bien encore d'appliquer à l'intérieur de cette zone un autre algorithme de compression mieux adapté.

Nos premières démarches se sont focalisées sur la problématique liée à l'initialisation du système, à la conception d'un boucle de suivi, et au contrôle des erreurs. L'objectif étant dans tous les cas de bâtir des algorithmes robustes avec un maximum de simplicité. Dans le cadre de cette application, la contrainte du temps réel est devenue le grand défi à surmonter, en nous obligeant à simplifier et optimiser nos algorithmes. Ce travail a été fait en collaboration étroite avec le groupe d'intégration dont le but final été de produire un démonstrateur temps réel. Ainsi, en développant des algorithmes simples et efficaces et en les mettant en œuvre nous avons valorisé nos approches théoriques et montré leur viabilité dans une finalité plus industrielle.

Première partie

Le Paradigme de la Segmentation : du pixel à l'objet

Chapitre 2

Introduction

L'objectif de ce chapitre est de présenter de façon intuitive le chemin qui nous amène du pixel, comme élément de base des images numériques, à l'objet conceptuel que nous apercevons : c'est le paradigme de la segmentation.

Nous commencerons ce parcours en introduisant le processus d'acquisition des images numériques, et plus en détail, nous allons nous intéresser à la représentation de la couleur : comment pouvons-nous créer un signal simulant la perception de la couleur telle qu'elle est faite par l'œil humain ? La réponse laisse entrevoir tout un éventail de possibilités, ce sont les espaces de couleur. Et parmi ceux-ci, nous plongerons dans les caractéristiques de l'espace YUV, standard de vidéo digital et par conséquent, entrée de nos algorithmes.

La segmentation de ces images devient donc la recherche d'une partition de l'espace la plus proche possible de la reconnaissance visuelle des objets. Ainsi, dans une première étape, le passage du pixel à la zone plate divise l'image par rapport à des zones homogènes.

Cependant la présence de bruit et de textures va nous rendre les résultats insatisfaisants. C'est là où la simplification de l'image devient de toute évidence nécessaire. Nous obtiendrons à l'aide des outils de filtrage, une image de plus en plus homogène dont la segmentation des objets doit se faire plus facilement. Néanmoins, nous verrons que la reconnaissance des objets par un sens sémantique reste encore un défi.

2.1 Les images numériques

De la scène réelle à l'image numérique que nous voyons à l'écran, plusieurs étapes se sont succédées. Nous allons par la suite les rappeler brièvement :

L'acquisition des images numériques. Deux éléments sont nécessaires lorsqu'il s'agit d'enregistrer une image numérique. Le premier est un dispositif physique sensible à la lumière, et capable de produire à sa sortie un signal électrique proportionnel au niveau d'énergie

captée. Le nombre de senseurs utilisés dans l'échantillonnage va déterminer la résolution spatiale de l'image enregistrée. A ce stade, une image monochrome peut être décrite comme une fonction f(x,y) représentant l'intensité lumineuse aperçue sur chacun des points (x,y) appartenaient à l'image. L'exemple le plus connu est celui des caméras de TV et les scanners. Deuxièmement, nous avons besoin d'un élément capable de convertir le signal analogique en un signal numérique. Ce processus comporte une perte d'information tandis que le signal continu est quantifié et discrétisé. La plupart des systèmes représentent la luminosité de l'image à une échelle globale de 256 niveaux de gris (8 bits par pixel), étant donné que l'œil humain n'est capable que de différencier 100 niveaux de gris de façon simultanée. Après la discrétisation, l'image peut être traitée comme une matrice de dimensions M×N dont M équivaut au nombre de lignes et N au nombre de colonnes. Chacun des points de cette matrice va être connu comme *pixel*.

L'enregistrement, le codage et la transmission. Toute étape d'acquisition suivie d'une étape de stockage aura besoin d'une grande quantité de mémoire. Pour donner un exemple, notons que la télévision numérique doit transmettre 25 images par seconde, dont chacune a une taille de 720×576 pixels, ce qui demande 10 Mbytes par seconde. Nous verrons dans la section suivante que la présence de couleurs va tripler cette quantité. Il devient alors nécessaire de comprimer afin de réduire cette lourde quantité de données. Finalement, si l'image doit être transmise, un processus de codage va transformer ces données en un signal adapté au canal de transmission.

La Réception et la visualisation à l'écran. Le décodage est le processus qui nous permet d'interpréter en réception le signal qui a été transmis. Certaines étapes de filtrage peuvent être incluses afin de réduire le niveau de bruit incorporé au signal pendant la transmission. Une fois que le signal a été décodé, la dernière étape consiste à fournir un outil de visualisation capable de transformer le signal numérique d'entrée en un ensemble de stimulus visuels qui vont être perçus par l'œil humain. Ces dispositifs nous les connaissons normalement comme des écrans.

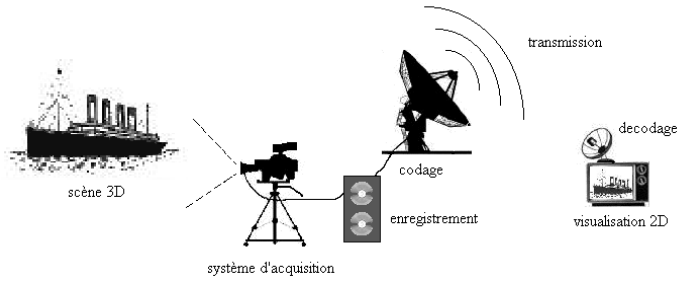


FIG. 2.1 – Enregistrement, transmission et visualisation des images numériques.

Jusqu'à présent nous nous sommes limités aux images monochromes. La modélisation de la couleur telle qu'elle est faite par le système visuel humain est du domaine de la *colorimétrie*. Par la suite, nous allons introduire les notions élémentaires sur les processus de perception et

de représentation de la couleur, en analysant plus en détail les caractéristiques de l'espace de couleur YUV, car ces images seront l'entrée de nos algorithmes.

2.1.1 Perception et représentation de la couleur

La colorimétrie est la théorie qui traite des aspects psychovisuels de la couleur, et plus spécialement de sa mesure et de sa spécification. Par la suite, nous allons faire un bref rappel des éléments de base, le lecteur pourra trouver plus de détails dans [31].

La sensation de couleur est causée par la présence ou absence de lumière. Différentes longueurs d'ondes (λ) sont à la base de couleurs différentes. Cependant l'œil humain ne peut apercevoir que l'ensemble des fréquences comprises entre 380nm (violet) et 780nm (rouge), que nous appellerons le *spectre visible* :

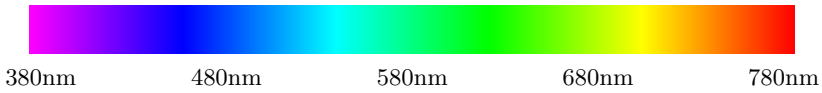


FIG. 2.2 – Spectre des couleurs visibles.

Néanmoins, les faisceaux lumineux naturels sont rarement purs. Cela équivaut à dire qu'une couleur est créée par un mélange additif de certaines proportions de longueurs d'ondes. Ainsi, en termes physiques, une couleur C va s'exprimer sous la forme d'une distribution spectrale,

$$C(\lambda) = \sum_{k=1}^{n} \beta_k P_k(\lambda) \tag{2.1}$$

dont $\Omega = \{P_k(\lambda)\}$, avec $k = 1, ..., n$ est la base de *l'espace couleur* de l'émetteur. Chaque couleur P_k est nommée *couleur primaire*. Le vecteur $(\beta_1, \beta_2, ..., \beta_n)$ définit les composantes de la couleur C dans la base Ω.

Les expériences de Young-Helmholtz ont montré qu'il suffisait de trois couleurs de base (rouge, vert et bleu) pour synthétiser presque tous les stimuli lumineux colorés perçus par l'œil humain. Certainement, une telle réduction ne permet pas la reconstitution parfaite de toutes les couleurs existantes dans la nature, mais si le dernier récepteur est l'œil, il suffit de s'adapter à sa réponse spectrale. Ainsi la Commission Internationale de l'Eclairage (CIE) a adopté en 1931 le système RGB comme standard de base.

Cependant, cette représentation ne s'adapte pas aux attributs perceptibles de l'intensité et à la tonalité des couleurs. Ainsi, les espaces de couleur ont évolué vers de nouvelles décompositions sur la base d'une composante de *luminance*, qui reflète les changements d'intensité lumineuse, et deux composantes de *chrominance* relatives à la teinte et la pureté des couleurs. Celles-ci définissent un plan connu comme *diagramme chromatique*. Dans tous les espaces de couleur, la ligne qui connecte l'origine (couleur noire) au point de la couleur blanche est dite *ligne achromatique*.

De nombreux modèles mathématiques ont été définis afin de créer un langage commun de description de la couleur. Certains se sont adaptés à la perception que nous avons de la couleur pour interagir facilement avec un utilisateur. Ainsi dans des espaces comme le HSV ou le HLS, chaque composante est associée à des attributs visuels comme la luminosité, la teinte ou la saturation. Par ailleurs, d'autres systèmes ont été optimisés pour simplifier les calculs. Mais nous allons nous intéresser à ce qui concerne les espaces de couleur utilisés par les systèmes de codage et de transmission des signaux vidéo.

L'espace de couleur du signal de télévision a été choisi afin d'assurer la compatibilité avec les systèmes de télévision en noir et blanc existant auparavant [98]. Ces systèmes transmettent le signal de luminance Y comme pondération expérimentale des composantes standard RGB enregistrées par les caméras.

$$Y = 0.299\,R + 0.587\,G + 0.114\,B \tag{2.2}$$

L'incorporation de la couleur au signal de TV noir et blanc a été faite en rajoutant deux composantes de chrominance, par différence de Y avec deux des couleurs primaires $(Y, R-Y, B-Y)$. Cet espace de couleur peut être obtenu par transformation linéaire du cube RGB comme le montre la Figure 2.4 (a)-(b). L'origine du système de coordonnées est toujours associé à la couleur noire, le point de luminance maximal au blanc, et les autres sommets du parallélépipède correspondent aux couleurs primaires et secondaires. Ici la ligne achromatique se trouve sur l'axe de luminance. Tous les systèmes basés sur la décomposition $(Y,\, R-Y,\, B-Y)$ sont appelés *systèmes de composantes vidéo* étant donné qu'ils s'appliquent aux transmission et codage des signaux vidéo. Plus particulièrement, nous nous intéressons à l'espace des couleurs YUV, reconnu comme standard de vidéo digitale par la norme CCIR-601. Ici, les composantes chromatiques ont été pondérées par des contraintes liées à la transmission :

$$\begin{aligned} U &= 0.56\,(B-Y) \\ V &= 0.71\,(R-Y) \end{aligned} \tag{2.3}$$

Cependant, on peut obtenir facilement les composantes YUV à partir des composantes RGB, en appliquant une matrice de transformation linéaire,

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.148 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.4}$$

En faisant cette transformation, l'information de luminance va être fortement décorrélée de l'information de chrominance. On peut remarquer cela sur la Figure 2.3, dont les composantes RGB et YUV d'une image couleur sont comparées. Parallèlement la Figure 2.4 illustre la transformation du point de vue géométrique. Notons que les composantes (U,V) définissent un plan chromatique de forme hexagonale (Figure 2.4 (c)). Dans les vues inférieures est supérieures du parallélépipède on peut observer clairement comment la teinte est associée

aux variations angulaires, tandis que la saturation (inverse de la quantité de blanc contenue dans le signal) dépend de la valeur radiale.

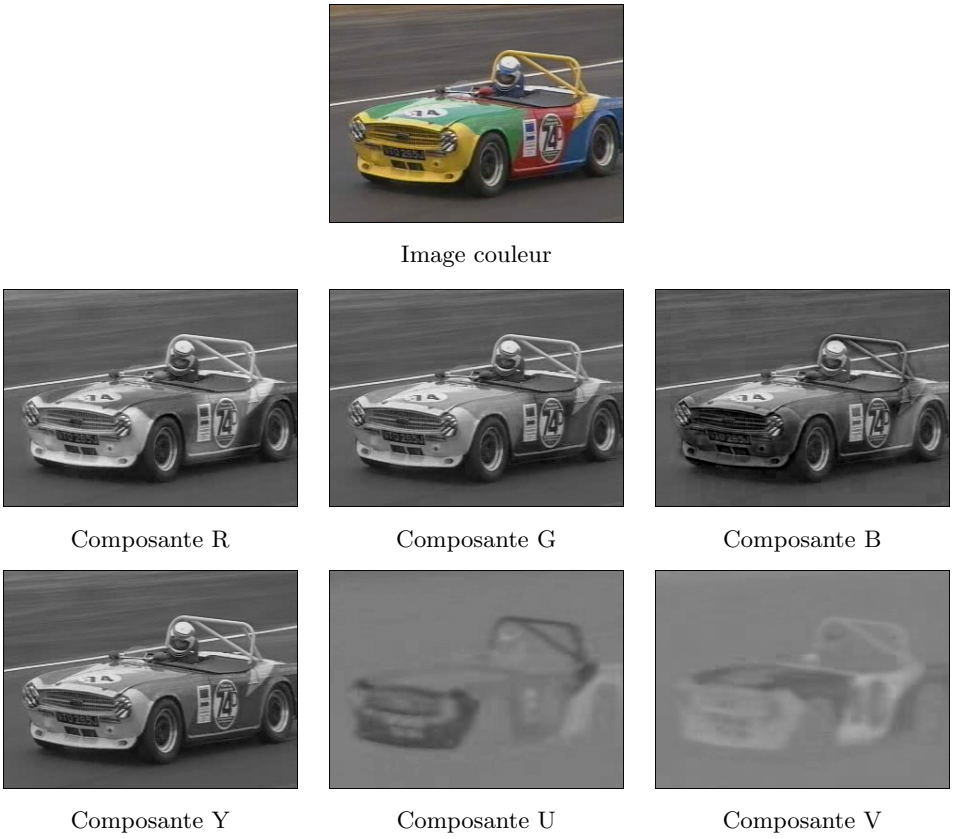
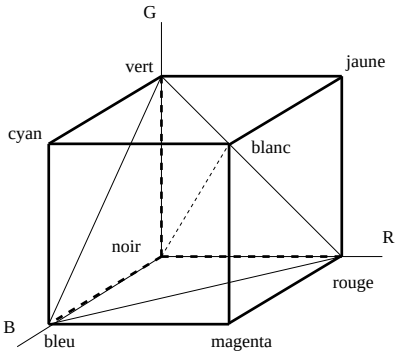


FIG. 2.3 – Décomposition d'une image couleur dans les espaces RGB et YUV.

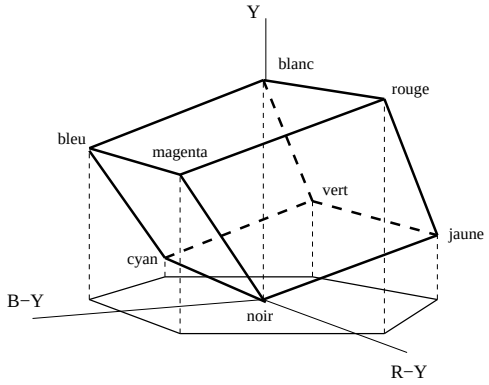
Il ne nous reste qu'à faire une remarque sur le codage des composantes de chrominance. On sait que l'œil n'est capable d'apprécier la couleur que lorsqu'il s'agit de surfaces relativement grandes. Sur les petits détails l'œil ne perçoit pas la teinte ni la saturation et garde seulement l'information de luminance. Ainsi les systèmes de codage vont transmettre une image monochrome plus détaillée, en faisant un sous-échantillonnage des composantes de chrominance. Bien que la dégradation ne soit pas visible, pour l'analyse d'image cette perte d'information devient une contrainte de plus.

2.2 La partition de l'espace

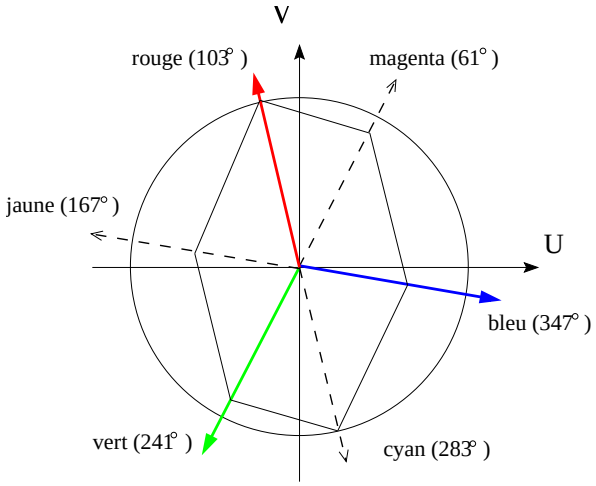
Nous avons commencé ce chapitre en introduisant de façon succincte les différentes étapes qui s'écoulent entre l'enregistrement d'une scène et la visualisation lointaine de l'image associée. Par la suite nous allons nous intéresser au contenu de ces images. Notre approche comporte deux étapes : la première nous mènera du pixel aux zones visuellement homogènes, puis nous obtiendrons des régions correspondant aux objets présents dans la scène.



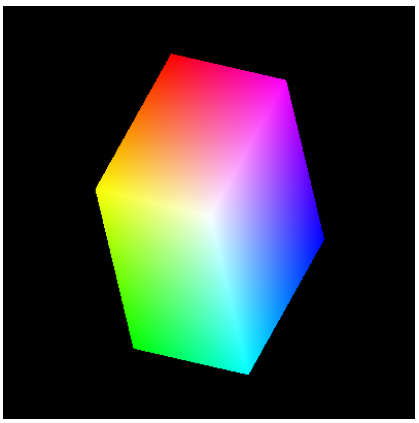
(a) Espace des couleurs RGB



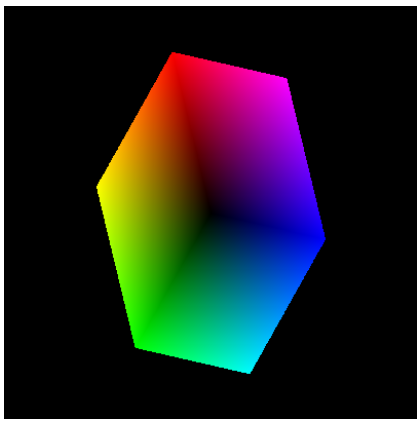
(b) Espace des couleurs $(Y, R-Y, B-Y)$



(c) Plan chromatique $(U, V) = (0.56(B-Y), 0.71(R-Y))$



(d) Vue supérieure



(e) Vue inférieure

FIG. 2.4 – Espace des couleurs YUV.

2.2.1 Des pixels aux zones plates

Nous avons défini auparavant le pixel comme l'élément indivisible à la base de toute grille numérique. Cependant, lorsque nous regardons une image nous ne somme pas capables de l'apercevoir à cause des limitations de notre acuité visuelle. On doit élargir fortement la taille de l'image, comme nous l'avons fait dans l'exemple de la Figure 2.5, pour arriver à le distinguer.

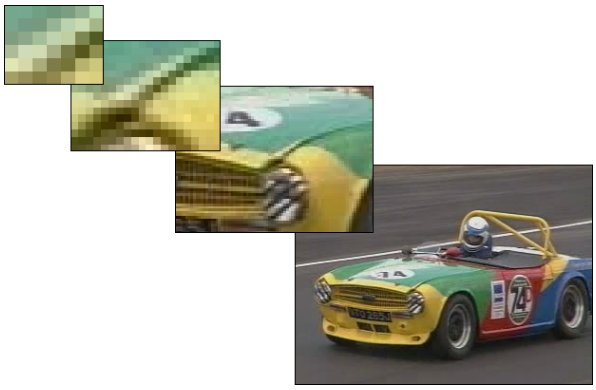


FIG. 2.5 – Le pixel comme élément à la base des images numériques.

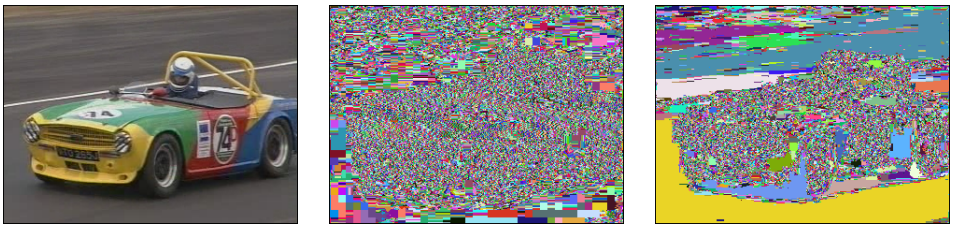
Néanmoins cette décomposition n'est pas très intéressante du point de vue de l'analyse de la scène car elle est indépendante du contenu. Nous pouvons aller plus loin et établir un lien parmi les pixels et les zones d'intérêt visuel. Nous trouvons ainsi deux classes différentes :

- les **zones plates** formées par un ensemble de pixels voisins qui ont tous une couleur semblable.

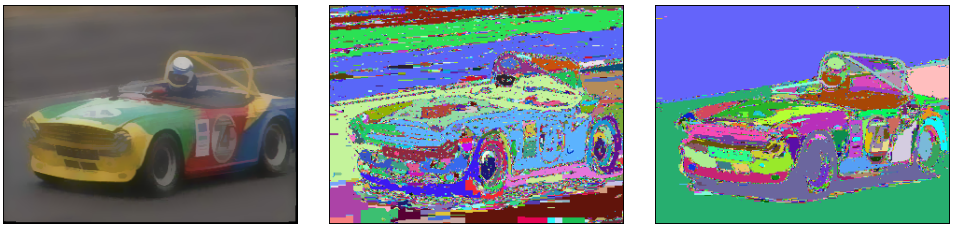
- les **zones de transition** comme la frontière qui sépare les pixels de couleurs très différentes. Ces frontières correspondent aux contours des objets.

Plus loin nous verrons que la définition de zone plate n'est pas unique : elle peut être formulée dans un sens strict (la couleur de tous les pixels d'une même zone plate doit être identique), ou en acceptant une certaine variation d'un pixel à l'autre comme fait l'œil humain. Nous avons illustré dans la Figure 2.6 (a) un exemple des différences que cette variation de critère peut entraîner. Ici chacune des zones plates de l'image *voiture* a été représentée avec une couleur uniforme. On peut apprécier clairement que plus le critère est strict, moins l'objet dans la scène est reconnaissable.

Bien qu'on obtienne des zones de plus en plus larges en assouplissant le critère d'homogénéité, on se rend compte que le bruit et les textures sont à l'origine de petites régions non désirées. Nous verrons dans le Chapitre 3 comment les algorithmes de filtrage peuvent élargir les zones plates en simplifiant l'image. A ce stade nous avons seulement fourni la comparaison dans (b) pour montrer la puissance de telles techniques :



(a) image originale critère strict critère souple



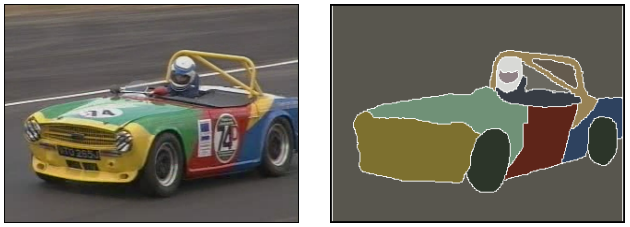
(b) image simplifiée critère strict critère souple

FIG. 2.6 – Détection des zones plates.

2.2.2 Des zones plates aux régions

La compréhension des images chez les personnes se base sur la reconnaissance des objets. Dans le domaine de l'imagerie numérique ce comportement est simulé en groupant des pixels dans des régions. Ce processus est connu comme *segmentation* et l'image résultante comme *partition*. Ainsi, dans un sens strict, il est correct de dire que l'ensemble des pixels forment une partition de l'image, dont chacun correspond à une région. Néanmoins cette partition est indépendante du contenu et n'apporte pas d'information permettant d'identifier les objets.

Imaginons à présent que quelqu'un nous demande de délimiter avec un crayon les contours présents dans l'image (a) de la Figure 2.7. La variété des réponses serait sûrement large, mais dans la plupart des cas elles ressembleront à la partition proposée dans (b), dont on a extrait de larges régions de couleurs différentes.

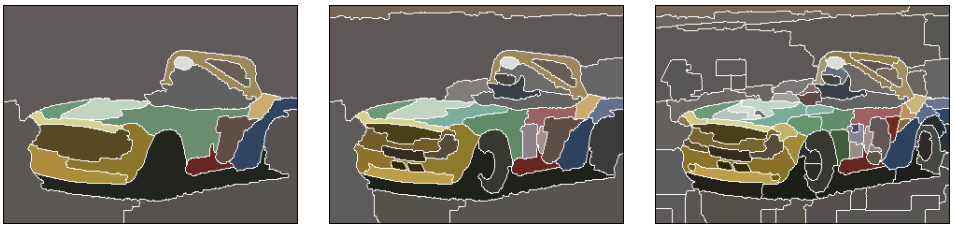


(a) Image originale (b) Segmentation visuelle

FIG. 2.7 – Des régions aux objets présents dans une scène.

Cependant on pourrait questionner la validité de ce résultat, car nous avons utilisé pour ce faire de l'information qui n'appartient pas à l'image. Ainsi, nous avons segmenté les roues alors qu'elles fusionnent complètement avec l'ombre de la voiture. Autrement dit, nous avons suppléé le manque de contraste avec une connaissance à priori de la forme de l'objet à extraire. Si ce même processus avait été fait automatiquement par une machine, nous aurions eu des

segmentations du type de celles montrées dans la Figure 2.8. Dans ce cas, on se rend compte de la vraie difficulté du problème car le système n'est capable de segmenter que les contours les plus contrastés, ce qui produit des résultats peu satisfaisants du fait du bruit et des textures.



10 régions 20 régions 100 régions

FIG. 2.8 – Segmentation automatique de la scène.

Arrivé à ce stade, il reste encore à achever la tâche la plus difficile, celle qui permet le passage des régions aux objets (Figure 2.9). Ainsi, si dans notre image on reconnaît visuellement une voiture, même son pilote, le système ne détecte que des régions car il ne dispose pas d'un sens sémantique.

FIG. 2.9 – Reconnaissance de l'objet.

L'expérience nous montre que dans la plupart de cas, la reconnaissance automatique d'objets reste une tâche extrêmement compliquée, l'interaction de l'utilisateur étant requise.

2.3 Conclusions

L'objectif que nous nous sommes fixé dans ce chapitre était d'introduire le lecteur dans la problématique de l'analyse des images numériques, et plus particulièrement, de lui faire sentir la difficulté liée à la segmentation et reconnaissance des objets présents dans une scène.

Le cadre de la segmentation étant posé, nous procéderons dans les chapitres qui suivent à l'analyse détaillée des algorithmes qui en font partie.

Chapitre 3

Simplification de l'Image : du pixel à la zone plate

En suivant le chemin intuitif que nous avons introduit au chapitre précédent, la simplification des images constitue le point de départ de tout processus de segmentation. L'objet de ce chapitre sera donc de fournir des outils puissants de filtrage permettant le passage du pixel à la zone plate.

Dans ce domaine, nous porterons une attention spéciale aux filtres morphologiques, et plus particulièrement à une nouvelle classe de filtres connexes connus comme les nivellements. Les nivellements ont prouvé qu'ils étaient des filtres très performants, capables d'atténuer fortement le bruit et les textures tout en préservant les fortes transitions du signal qui marquent les contours des objets.

Dans tous les cas, nous analyserons comment les définitions faites pour de fonctions scalaires peuvent s'étendre aux fonctions multi-spectrales qui caractérisent les images couleur, véritable entrée des nouvelles applications de l'imagerie numérique.

Finalement, nous allons conclure ce chapitre en analysant les performances des outils de filtrage lorsqu'ils sont appliqués au pré-traitement des images, soit en vue de la segmentation, soit en vue du codage.

3.1 Les zones plates

Tout d'abord nous allons introduire quelques définitions et notions essentielles à la caractérisation des zones plates. Cela nous permettra d'abandonner le pixel comme élément de base d'une image numérique, pour des groupements de pixels qui cherchent à mieux s'approcher du sens visuel de l'homogénéité des régions.

Pour une étude théorique plus approfondie sur la notion de connexion pour des images numériques, le lecteur pourra s'adresser aux travaux de Matheron et Serra dans [86]. Plus récemment, ces définitions ont été élargies par Serra donnant lieu aux connexions de seconde génération [89].

3.1.1 Définitions de base

Dans tout ce qui suit, nous allons considérer les images numériques sous la forme d'un *graphe* $G = \{V_G, A_G\}$. Les coordonnées de chacun des pixels déterminent les points de la grille $V_G : \mathcal{Z} \times \mathcal{Z}$, tandis que les relations de voisinage définissent l'ensemble des arêtes $A_G \in \mathcal{Z}^2$.

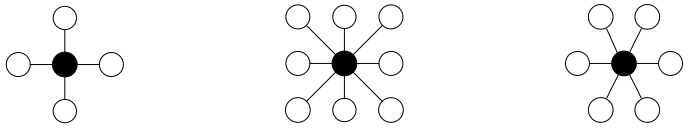
Ainsi, nous pouvons introduire la notion de voisinage d'un point p dans la grille V_G comme l'ensemble de pixels qui sont directement connectés à lui,

$$\forall\, p,\, q \in V_G, \quad \text{p et q sont voisins} \Leftrightarrow (p, q) \in A_G$$

où le paire ordonné (p, q) est l'arête qui relie les points p et q. Plus formellement nous dirons que le voisinage du pixel p, $N_G(p)$, définit un sous ensemble de V_G de taille quelconque, tel que

$$\forall\, p \in V_G, \quad N_G(p) = \{\, q \in V_G, (p, q) \in A_G\}$$

Nous allons supposer toujours $N_G(p)$ étant invariant par translation et symétrique par rapport au point p. Ainsi, définir un voisinage revient à définir la connexité des pixels sur la grille numérique. En pratique, les relations de connexité les plus utilisées dans $\mathcal{Z}^2$ sont du type hexagonal (6 voisins par nœud), ou carré (4 ou 8 voisins par nœud). Notons tout simplement que la trame hexagonale est préférée du point de vue algorithmique pour ses bonnes propriétés de symétrie, tandis que la trame carrée à 4 ou 8 connexités est largement plus appropriée car elle correspond à la position des capteurs des caméras.



Connexité 4 Connexité 8 Connexité 6

FIG. 3.1 – Différents types de connexité pour des images numériques.

Sur cette base nous allons définir les termes suivants :

Définition 1 (Chemin) *Soient (x,y) deux pixels dans la trame V_G. Nous dirons qu'il existe un chemin reliant (x,y) si, et seulement si, il existe un n-uplet de pixels $(p_0, p_1, \ldots, p_n)$ tel que $p_0 = x$ et $p_n = y$ et*

$$p_i \in N_G(p_{i-1}) \qquad \forall i = 1, \ldots, n$$

A partir de la définition du chemin, nous allons considérer des groupements de pixels sous la notion de composante connexe,

Définition 2 (Composante connexe) *Soit A un ensemble de pixels inclus dans V_G, et soit p un pixel de A. La composante connexe de A qui contient p, $C_p(A)$, est l'union de tous les chemins d'origine p inclus dans A.*

Il serait alors possible de définir une zone plate comme la plus large composante connexe dont la fonction est constante, ou bien comme nous l'avons fait, en l'exprimant directement par la notion de chemin,

Définition 3 (Zone plate) *Deux pixels (x,y) appartiennent à la même zone plate de la fonction f si, et seulement si, il existe un n-uplet de pixels $(p_0, p_1, \ldots, p_n)$ tel que $p_0 = x$ et $p_n = y$, et pour tous les i, (p_i, p_{i+1}) sont voisins et ont la même valeur : $f_{p_i} = f_{p_{i+1}}$.*

Remarquons seulement qu'une zone plate peut être réduite à un seul pixel de l'image. Cependant, le critère qui caractérise les zones plates dans un sens strict peut être assoupli afin de créer des partitions de plus en plus grossières. Nous parlerons donc des zones quasi-plates, définies comme suit,

Définition 4 (Zone quasi-plate) *Deux pixels (x,y) appartiennent à la même zone quasi-plate de la fonction f si, et seulement si, il existe un n-uplet de pixels $(p_0, p_1, \ldots, p_n)$ tel que $p_0 = x$ et $p_n = y$, et pour tous les i, (p_i, p_{i+1}) sont voisins et leurs valeurs vérifient une relation symétrique tel que :*

$$\forall i \quad R(p_i, p_{i+1}) = R(p_{i+1}, p_i)$$

Parmi tout l'éventail de possibilités qu'une telle définition permet d'envisager, nous nous sommes intéressés aux relations symétriques du type :

$$\|f_{p_i} - f_{p_{i+1}}\| \leq \lambda$$

Ainsi nous dirons que deux pixels appartiennent à la même zone quasi-plate s'il est possible d'établir un chemin parmi eux dont la pente maximale soit inférieure ou égale à λ. Sous ce critère les zones strictement plates sont obtenues en prenant $\lambda = 0$. Pour des valeurs de $\lambda > 0$ il est important de remarquer que dans une même zone quasi-plate il peut y avoir des chemins de pente supérieure à λ. Ceci s'illustre parfaitement dans la Figure 3.2 sur un exemple synthétique. Si on prend $\lambda = 0$ il existe autant de zones plates que de pixels. Si on prend $\lambda = 1$ l'image est divisée en deux zones quasi-plates. Notons que la pente entre deux pixels de la même zone peut être plus forte que celle existante entre deux pixels de zones différentes.

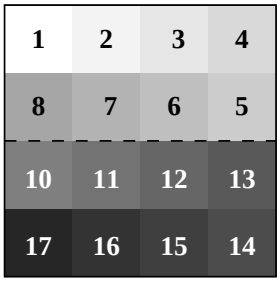


FIG. 3.2 – Exemple de zones quasi-plates.

La Figure 3.3 montre un exemple de la différence entre le nombre de zones plates obtenues en appliquant la définition stricte ou en permettant une légère pente ($\lambda = 1$). On observe dans ce dernier cas que les zones finement texturées se sont regroupées dans de plus larges régions, réduisant jusqu'à 50% le nombre de régions. Mais, par ailleurs l'expérience nous montre qu'une relaxation excessive de ce critère peut entraîner une perte d'information significative au niveau des contours.

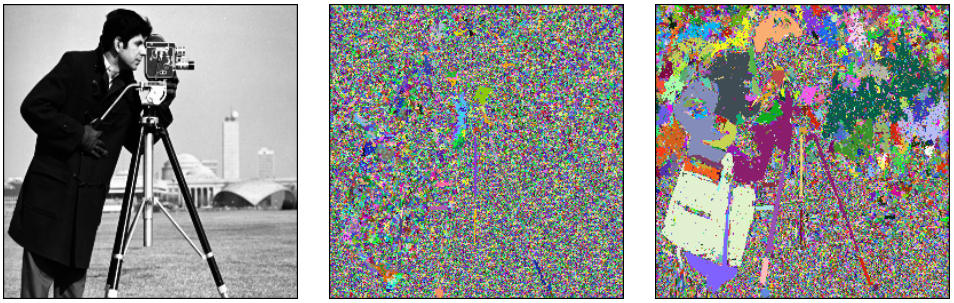


Image originale $\|f_{p_i} - f_{p_{i+1}}\| = 0$ $\|f_{p_i} - f_{p_{i+1}}\| = 1$

FIG. 3.3 – Caractérisation des zones quasi-plates.

Jusqu'ici nous avons défini les zones plates sur les images monochromes comme étant des zones homogènes en niveaux de gris. A présent, si nous voulons étendre ce concept aux images couleur de façon cohérente, nous dirons qu'une *zone de couleur plate* est la composante connexe la plus grande ayant tous les vecteurs de couleur identiques. Par contre, lorsqu'il s'agit d'établir une équivalence avec les zones quasi-plates, le choix est beaucoup plus large car les inégalités prennent un sens vectoriel.

Etant donné l'absence d'ordre total dans un espace vectoriel, nous avons classé les différentes approches selon qu'elles traitent chacune des composantes séparément (approche marginale) ; de façon conjointe (approche vectorielle) ; ou de façon mixte (par exemple, le traitement séparé de la luminance par rapport à la chrominance). Plus formellement nous avons défini les zones de couleur quasi-plate comme,

Définition 5 (Zone de couleur quasi-plate) *Soit $\vec{c}_p = \{c_p^0, c_p^1, c_p^2\}$ le vecteur de couleur associé au pixel p. Nous dirons que deux pixels (x, y) appartiennent à la même zone de couleur quasi-plate, si et seulement si, il existe un ensemble de pixels $(p_0, p_2, \ldots, p_n)$ tel que $p_0 = x$ et $p_n = y$, et pour tous les i, (p_i, p_{i+1}) sont voisins et leur vecteurs de couleur, $\vec{c}_i$ et $\vec{c}_{i+1}$, vérifient une relation symétrique du type :*

- *marginale :* $\forall i \quad R(c_i^n, c_{i+1}^n) = R(c_{i+1}^n, c_i^n) \quad n = 0, 1, 2$
- *vectorielle :* $\forall i \quad R(\vec{c}_i, \vec{c}_{i+1}) = R(\vec{c}_{i+1}, \vec{c}_i)$

ou n'importe quelle approche mixte.

L'approche marginale est l'extension la plus directe des critères utilisés en une dimension, car les régions sont générées par intersection des zones homogènes de chacune des composantes. Remarquons seulement que pour les images couleur, le critère peut changer d'une composante à l'autre pour tenir compte des caractéristiques particulières de l'espace des couleurs considéré.

Les approches purement vectorielles se basent exclusivement sur la définition d'une fonction distance, établissant un ordre réduit parmi les vecteurs de l'espace.

Finalement, à cheval sur ces deux catégories se trouvent les méthodes mixtes, qui traitent d'un coté une des composantes, et de l'autre coté le groupement des autres dans un plan vectoriel. En guise d'exemple, les lambda zones plates définies auparavant sur une dimension, peuvent s'étendre à l'espace des couleurs YUV sous une des approches suivantes,

- marginale : $L_1(y_i, y_{i+1}) \leq \lambda_y$ et $L_1(u_i, u_{i+1}) \leq \lambda_u$ et $L_1(v_i, v_{i+1}) \leq \lambda_v$

- vectorielle : $L_2(\vec{c}_i, \vec{c}_{i+1}) \leq \lambda_c$

dont L_1 et L_2 équivalent aux distances de Minkowski d'ordre 1 et 2 respectivement.

Dans le premier cas il est envisageable de fixer des valeurs de λ différents pour chacune des composantes. Couramment nous prendrons $\lambda_y \geq \lambda_u = \lambda_v$ car, comme nous avons vu dans le chapitre précédent, l'espace YUV n'est pas uniforme (les variations de luminance sont beaucoup plus fortes que celles de la chrominance).

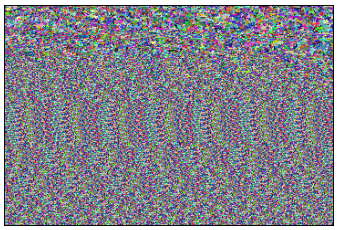
La Figure 3.4 illustre un exemple de la détection de zones de couleur quasi-plate par extension marginale du critère défini en une dimension. Dans (a) l'application du critère strict de zone plate limite le groupement des pixels, tandis que dans (b), en permettant une certaine pente, les régions très homogènes, comme le ciel ou la mer deviennent facilement reconnaissables. Par contre, si on prend des seuils encore plus larges pour homogénéiser les régions plus texturées, on risque de fusionner des objets indépendants. Ainsi, dans (c), en permettant de fortes variations de luminance afin d'unifier les ombres parmi les herbes, nous avons également fusionné les régions représentant le ciel et de la mer.

En concordance avec la définition que nous avons faite auparavant, nous dirons que deux pixels appartiennent à la même zone s'il est possible d'établir un chemin parmi eux de pente inférieure à λ. Pour des valeurs de λ équivalents, l'approche vectorielle permet des variations plus fortes. Par conséquent, il est alors logique d'espérer que les zones plates ainsi générées seront encore plus larges que les marginales. Mais il est intéressant de remarquer que,

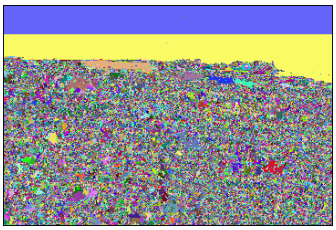
- l'approche marginale permet d'établir des chemins différents pour chacune des composantes à l'intérieur d'une même zone plate. Par contre,

- les approches vectorielles ne peuvent créer une zone plate que s'il existe au moins un chemin commun à toutes les composantes de l'espace considéré.

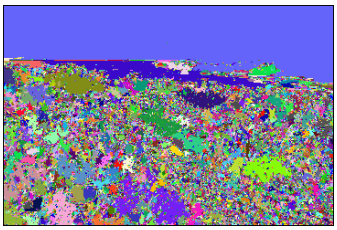
Image couleur



(a) $\lambda_y = \lambda_u = \lambda_v = 0$



(b) $\lambda_y = 4, \lambda_u = \lambda_v = 2$



(c) $\lambda_y = 6, \lambda_u = \lambda_v = 3$

FIG. 3.4 – Caractérisation des zones de couleur quasi-plate.

Nous avons vu jusqu'ici que la présence de bruit ainsi que de textures ne permet pas de détecter les zones plates comme le fait l'œil humain. Bien que beaucoup d'autres approches soient possibles dans le but d'élargir les zones plates, l'évaluation des critères proposés met déjà en évidence une dichotomie : plus on tolère de dérives pour surmonter les textures, plus on risque de fusionner des objets proches en couleur. Par contre, nous verrons par la suite qu'il existe des opérateurs capables de simuler cette réponse visuelle en homogénéisant les petites variations du signal.

3.2 Les filtres morphologiques connexes

L'utilisation des filtres a été largement répandue dans le traitement des images numériques dans le but de simplifier l'image. Et parmi les filtres classiques nous nous intéresserons plus particulièrement aux filtres morphologiques.

Un filtre morphologique est une transformation croissante et idempotente. La première des conditions est la plus importante, étant donné qu'elle assure la préservation de l'ordre dans le treillis après filtrage. La deuxième nous informe qu'il s'agit d'un processus irréversible qui entraîne une perte d'information.

Néanmoins, avant de rentrer dans la théorie inhérente à une analyse plus détaillée, nous allons faire une première approche au niveau des effets du filtrage sur l'image. Le lecteur, s'il le désire, pourra trouver plus d'informations dans les ouvrages de référence en ce domaine [86, 87].

3.2.1 Effets du filtrage sur les contours de l'image

Lorsqu'on utilise des filtres pour simplifier une image et élargir les zones plates, on risque fortement de dégrader les contours des objets. Afin d'illustrer ce problème, nous avons étudié les résultats obtenus par deux des filtres les plus utilisés : le Filtre Gaussien, et le Filtre Alterné Séquentiel, qu'on trouve aisément dans la littérature de base de l'analyse d'image [39, 79].

A gauche de la Figure 3.5 on a placé l'image originale, ainsi que le relief de la coupure horizontale marquée sur l'image. On observe comment la présence de bruit et des textures provoque des pics dans le relief du signal tandis que les contrastes dûs aux contours des objets marquent les transitions les plus fortes. Pour le filtrage gaussien, on remarque clairement que les pics causés par le bruit ont été supprimés dans leur totalité. Cependant, l'atténuation des hautes fréquences comporte aussi un adoucissement des transitions liées aux vrais contours. La conséquence la plus directe est la perception floue de l'image. Afin de fournir une comparaison, nous avons filtré aussi cette image par un Filtre Alterné Séquentiel. La simplification achevée est encore plus forte que celle obtenue précédemment, même si les transitions sont plus brusques. Néanmoins il apparaît un autre effet non désiré : la déformation des contours, car on observe que ceux-ci ont pris la forme de l'élément structurant utilisé à la base du filtrage.

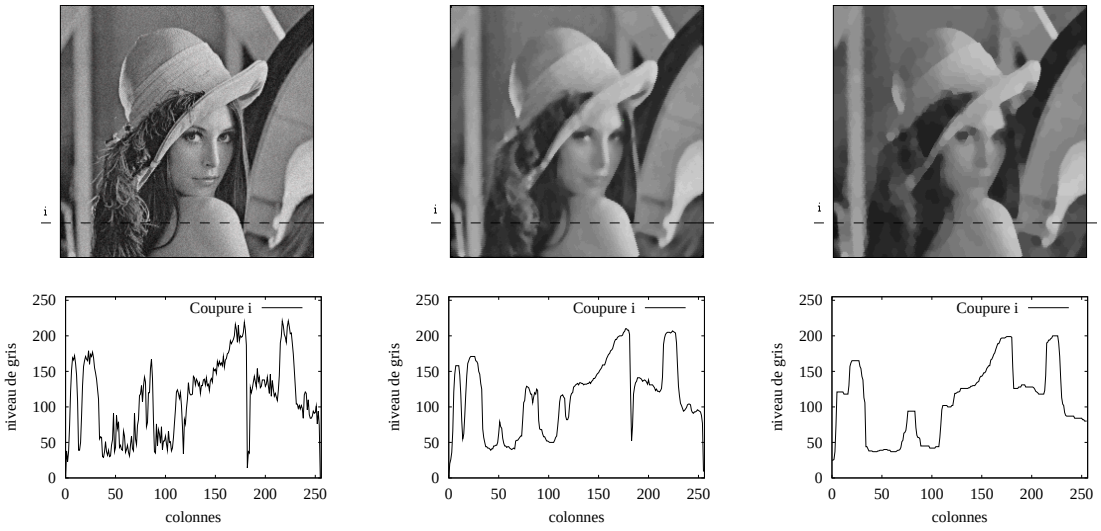


Image bruitée F. Gaussien F. Alterné Séquentiel

FIG. 3.5 – Effets du filtrage sur l'image.

Dans la section qui suit nous verrons qu'il existe des filtres morphologiques capables de simplifier les petites rugosités du signal en reproduisant en détail les fortes transitions.

3.2.2 Définitions et caractérisation

Depuis leur apparition, les filtres morphologiques par reconstruction sont de plus en plus utilisés car ils ont une rare propriété : ils simplifient l'image tout en préservant la qualité des

contours. Cette caractéristique leur a valu une place d'honneur comme outils de filtrage pour des applications très variées, allant de la suppression de bruit jusqu'à la segmentation.

Et pourtant ça n'a été que quelques années plus tard que ces propriétés ont été expliquées par Serra et Salembier dans [90, 84]. Ainsi, on a appris que les filtres morphologiques par reconstruction faisaient partie d'une classe beaucoup plus large nommée les *opérateurs connexes*. Ceux-ci se définissent comme,

Définition 6 (Opérateur connexe) *Soit T un treillis complet de fonctions. Un opérateur $\Psi : T \to T$ est dit connexe si pour toute fonction $f \in T$, se vérifie :*

$$\forall (p, q)\ voisins, \quad f_p = f_q \Rightarrow \Psi(f_p) = \Psi(f_q)$$

Les opérateurs connexes agissent sur l'image au niveau des zones plates, que nous avons définies au début de ce chapitre comme étant les composantes connexes les plus larges dont l'image est homogène en niveau de gris. C'est-à-dire que les opérateurs connexes élargissent les zones plates d'une fonction. Ainsi, on a appelé *aplanissement* au résultat obtenu par application d'un opérateur connexe à une fonction. Plus formellement,

Définition 7 (Aplanissement) *Soient f et g deux fonctions d'un treillis T. Nous dirons que g est un aplanissement de f si et seulement si,*

$$\forall (p, q)\ voisins, \quad f_p = f_q \Rightarrow g_p = g_q$$

Par ailleurs, des opérateurs connexes comme classe générale aux filtres par reconstruction, l'échelle est bien trop large. Meyer a proposé une classification plus fine dans [62, 63, 64] en introduisant des sous classes et en établissant un ordre d'inclusion parmi elles, que nous avons illustré dans la Figure 3.6. On retrouve à droite les aplanissements comme étant la classe la plus générale qui rassemble tous les opérateurs connexes. Par opposition, à gauche se placent les ouvertures et fermetures par reconstruction, s'agissant des algorithmes les plus restreints. Et parmi ces deux extrêmes on a défini les aplanissements monotones, les aplatissements, et les nivellements, qui ont de loin les propriétés les plus attractives.

Finalement nous présentons les arasements et les inondations comme un cas particulier de nivellements desquels dérivent les ouvertures et fermetures par reconstruction. Par la suite nous allons introduire l'ensemble de ces opérateurs.

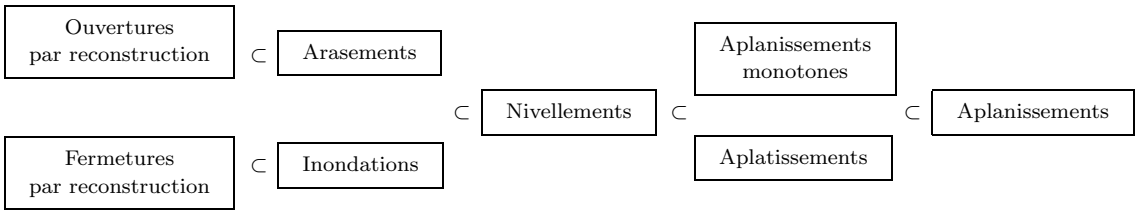


FIG. 3.6 – Propriétés d'inclusion des opérateurs connexes.

Nous avons vu auparavant que le principal atout des aplanissements était leur façon de traiter l'image au niveau des zones plates : si $f_p = f_q$ alors $g_p = g_q$, et de façon équivalente, si $g_p \neq g_q$ alors $f_p \neq f_q$. Cela signifie que, pour chaque transition parmi deux pixels dans g, il doit y avoir aussi une transition dans f, mais pas forcément suivant le même sens de variation. Par conséquent, les aplanissements peuvent créer de nouveaux maxima et minima comme le montre l'exemple de la Figure 3.7.

Pour empêcher de tels changements, on a défini les aplanissements monotones comme,

Définition 8 (Aplanissement monotone) *Soient f et g deux fonctions d'un treillis T. Nous dirons que g est un aplanissement monotone de f si et seulement si,*

$$\forall (p,q)\ voisins,\ \ g_p > g_q \Rightarrow f_p > f_q$$

La contrainte que nous avons imposée force la concordance parmi le sens des variations de l'image originale et ceux de l'image traitée. On peut affirmer que les aplanissements monotones ne créent jamais des extrema régionaux. Cependant, la fonction aplatie peut pendre des valeurs qui se trouvent en dessous et au dessus de la fonction originale. Ils peuvent donc aplatir certaines zones mais aussi étirer le contraste parmi d'autres (voir un exemple dans la Figure 3.7).

Pour assurer l'aplatissement global de la fonction à partir des aplanissements, on a dû définir une nouvelle classe d'opérateurs en rajoutant une contrainte sur la dynamique des transitions.

Définition 9 (Aplatissement) *Soient f et g deux fonctions d'un treillis T. Nous dirons que g est un aplatissement de f si et seulement si,*

$$\forall (p,q)\ voisins,\ \ g_p > g_q \ \ \Rightarrow \begin{cases} f_p \geq g_p\ et\ g_q \geq f_q \\ ou \\ f_q \geq g_p\ et\ g_q \geq f_p \end{cases}$$

Cela signifie que chacune des transitions de la fonction g est emboîtée dans une transition plus forte de la fonction f de départ. Par conséquent, elles peuvent tourner un maximum dans un minimum, et à l'inverse, comme c'est le cas de l'exemple de la Figure 3.7.

Les aplatissements étant des opérateurs connexes, ne créent jamais de nouveaux contours, et on peut démontrer que dans toutes les zones où $f \neq g$, g est plate.

Finalement, la combinaison des deux opérateurs précédents a donné lieu à une nouvelle classe, les nivellements, qui héritent des bonnes propriétés de chacun d'eux.

Définition 10 (Nivellement) *Soient f et g deux fonctions d'un treillis T. Nous dirons que g est un nivellement de la fonction f, si et seulement si elle vérifie la relation,*

$$\forall (p,q)\ voisins,\ \ g_p > g_q \ \Rightarrow\ f_p \geq g_p\ et\ g_q \geq f_q$$

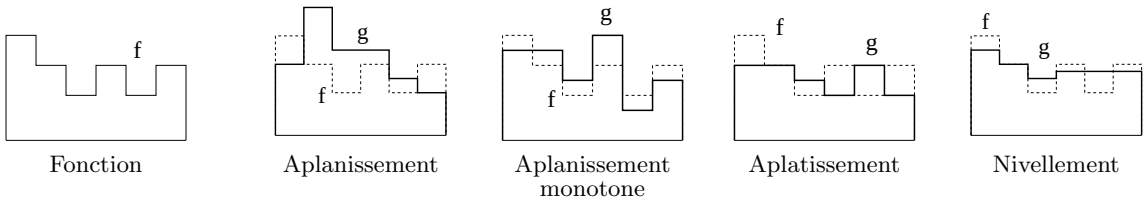


FIG. 3.7 – Différences parmi les opérateurs connexes.

Les nivellements réduisent la dynamique des pics du signal tout en préservant le sens des variations. Ainsi, on peut démontrer qu'à l'intérieur de chaque minimum (resp. maximum) régional de la fonction g correspond un minimum (resp. maximum) régional dans la fonction f. Cela n'empêche que certains des extrema régionaux de la fonction f peuvent être aplatis dans g. On peut prouver aussi que, sauf pour les fonctions constantes, les nivellements sont des opérateurs antisymétriques : si f est un nivellement de g et, de façon simultanée, g est un nivellement de f, nous pouvons conclure que $f = g$. Le lecteur qui le désire pourra trouver une analyse approfondie de toutes ces propriétés dans [63, 64].

Par la suite nous allons nous intéresser au processus de construction des nivellements, car le critère que nous avons vu auparavant nous permet seulement d'identifier une fonction g comme étant un nivellement de la fonction f. Ainsi, dans la Figure 3.8 nous dirons que g est un nivellement de f si pour toute paire de pixels voisins, (i, j), se vérifie la relation,

$$g_i > g_j \;\Rightarrow\; f_i \geq g_i \;\; et \;\; g_j \geq f_j$$

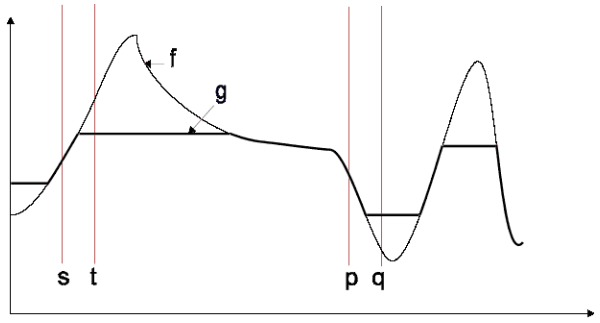


FIG. 3.8 – Définition des nivellements.

En outre, la condition de nivellement sur une paire de pixels voisins (p, q) peut s'exprimer avec une seule inégalité en prenant le supremum ($\vee$) et l'infimum ($\wedge$) de f_p et g_q, et par extension, de f et g dans tout le voisinage de p :

$$\forall (p,q)\, voisins, \qquad f_p \wedge g_q \;\leq g_p \leq\; f_p \vee g_q \tag{3.1}$$

$$\forall p, \qquad f_p \wedge \Big(\sup_{\forall q \in N_p} (g_q) \Big) \;\leq g_p \leq\; f_p \vee \Big(\inf_{\forall q \in N_p} (g_q) \Big) \tag{3.2}$$

Par ailleurs cette dernière expression continue à être vraie si nous considérons aussi la valeur du point central dans le voisinage,

$$\forall p, \qquad f_p \wedge (g_p \vee \sup_{\forall q \in N_p} (g_q)) \leq g_p \leq f_p \vee (g_p \wedge \inf_{\forall q \in N_p} (g_q)) \tag{3.3}$$

A ce stade, nous pouvons exprimer les relations au niveau des pixels par des opérateurs morphologiques. Notons que, pendre le maximum dans un voisinage équivaut à faire une dilatation morphologique par un élément structurant plat contenant le point central ainsi que tous ces voisins (δg) ; et par dualité, pendre le minimum dans un voisinage équivaut à faire une érosion morphologique (εg). En introduisant ces opérateurs, nous aboutions à une deuxième formulation des nivellements, établie par Matheron dans [57] :

Critère 1 (Nivellement) *Soient f et g deux fonctions d'un treillis T. Nous dirons que g est un nivellement de la fonction f, si et seulement si, se vérifie la relation*

$$\forall p, \qquad f_p \wedge \delta g_p \leq g_p \leq f_p \vee \varepsilon g_p$$

Ce nouveau critère définissant les nivellements nous montre le chemin à suivre pour obtenir un nivellement de f par transformation d'une fonction g quelconque. Dans tout ce qui suit, g' désignera la *fonction nivellement* de f obtenue à partir de g, dont f sera nommée la *fonction de référence*, et g la *fonction marqueur*. En pratique g' sera le plus fort nivellement de f obtenu par édition de g en itérant le critère d'assignation suivant :

- Dans $\{g < f\}$ on doit remplacer g par δg jusqu'à ce que la création d'une zone plate, ou jusqu'à ce que la fonction g' soit égale à la fonction f.
- Dans $\{g > f\}$ on doit remplacer g par εg jusqu'à ce que la création d'une zone plate, ou jusqu'à ce que la fonction g' soit égale à la fonction f.

La Figure 3.9 illustre un exemple de la procédure décrite sur deux fonctions continues, f et g, en une dimension. On peut remarquer que sur le nivellement g', toutes les composantes connexes où $g' \neq f$ sont plates, ce qui réduit la dynamique des transitions.

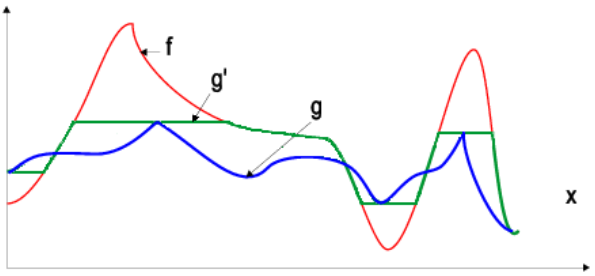


FIG. 3.9 – Construction d'un nivellement à partir d'une fonction marqueur.

On observe également que le nivellement reproduit en détail les zones de forte transition dans la fonction de référence f : à chaque transition entre deux pixels de la fonction nivelée g' correspond une transition, même plus forte, dans la fonction f. Cette propriété fait des nivellements d'excellents outils de filtrage. Pour cette raison, une description détaillée de leur mise en œuvre, ainsi qu'une étude approfondie de leurs applications, fera le sujet de la Section 3.3.

A titre d'exemple, nous allons niveler dans la Figure 3.10 l'image *Lena* utilisée auparavant pour illustrer les effets du filtrage sur les contours. Afin de mieux visualiser la façon dont le nivellement agit sur les valeurs des pixels, nous avons suivi les changements du profil de l'image le long d'une ligne. Sur le signal associé à l'image originale, on observe clairement des pics resserrés, conséquence du bruit présent dans l'image de référence.

L'objectif de nos algorithmes sera donc de filtrer ces pics sans pourtant dégrader les transitions qui correspondent aux contours des objets. Pour atteindre un tel but, nous avons pris comme marqueur une image très aplatie qui correspond à une simplification grossière de l'image de référence.

Nous pouvons avancer déjà que le degré de simplification de l'image marqueur va déterminer le niveau de simplification de l'image nivelée. Cela signifie que si le marqueur est complètement en dessous ou en dessus d'un certain pic du signal de référence, cette transition ne sera pas reconstruite par le nivellement. Néanmoins, sur les profils montrés dans l'exemple, le lecteur doit se rendre compte que la préservation de certaines zones (par exemple le reflet du chapeau dans le miroir) est le résultat de la propagation bidimensionnelle des valeurs du marqueur.

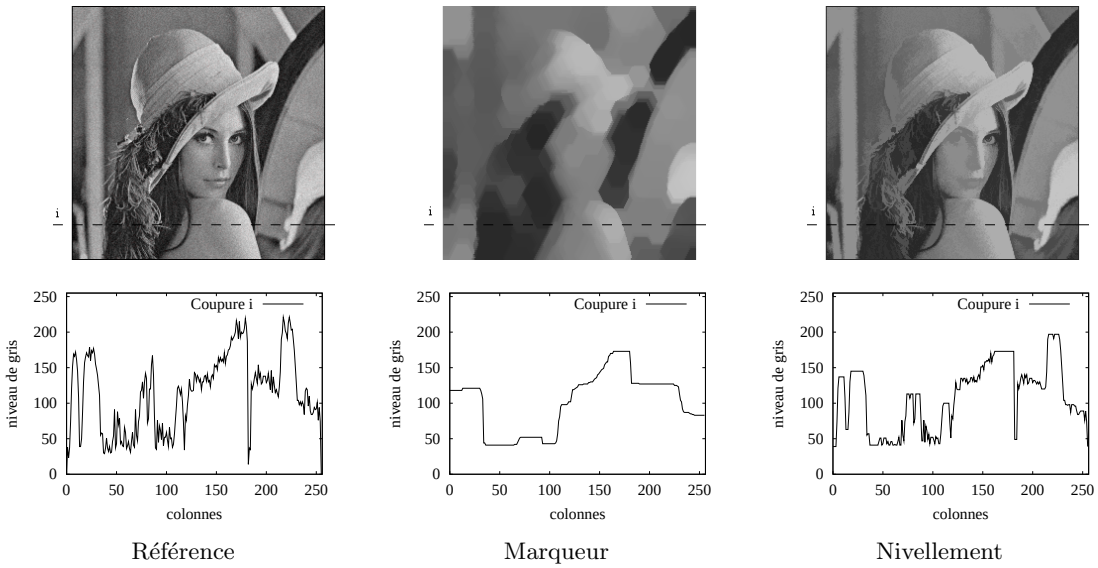


FIG. 3.10 – Application des nivellements au filtrage d'une image bruitée.

Sur l'exemple précédent il est aisé d'observer que les nivellements, étant des opérateurs connexes, élargissent les zones plates. Par conséquent, le résultat d'un nivellement va dépendre de la définition de zone plate sous-jacente. Implicitement, jusqu'ici, nous avons montré des

exemples en utilisant des zones strictement plates, mais nous pourrions également considérer des zones quasi-plates afin de générer des simplifications encore plus fortes. Ainsi, dans une formulation plus générale nous dirons que la fonction g est un nivellement de f si et seulement si,

$$\forall (p,q)\, voisins,\ \ g_p \succ g_q \ \Rightarrow\ f_p \succeq g_p \ et\ g_q \succeq f_q \tag{3.4}$$

A chaque définition de $\succ$ correspondra une classe étendue de nivellement. Plus concrètement, si nous reprenons la définition des zones quasi-plates dont $\forall i\ \|f_{p_i} - f_{p_{i+1}}\| \leq \lambda$, nous aboutirons à des inégalités,

$$\begin{aligned} g_p \succ g_q &\ \Leftrightarrow\ g_p > g_q + \lambda \\ g_p \preceq g_q &\ \Leftrightarrow\ g_p \leq g_q + \lambda \end{aligned} \tag{3.5}$$

en prenant $\lambda \geq 0$. Ainsi, l'expression 3.1 caractérisant les nivellements basiques devient,

$$\forall (p,q)\, voisins,\ \ f_p \wedge (g_q - \lambda)\ \leq g_p \leq\ f_p \vee (g_q + \lambda) \tag{3.6}$$

Ensuite, l'extension au voisinage de p peut se formuler avec l'aide de dilatations et d'érosions morphologiques,

$$\forall p \quad f_p \wedge \delta_\lambda(g_p)\ \leq g_p \leq\ f_p \vee \varepsilon_\lambda(g_p) \tag{3.7}$$

dont,

$$\varepsilon_\lambda(g) = g \wedge (\varepsilon g + \lambda) \tag{3.8}$$

$$\delta_\lambda(g) = g \vee (\delta g - \lambda) \tag{3.9}$$

Les nivellements ainsi créés seront appelés *lambda nivellements*. Les deux derniers opérateurs, nommés *lambda érosion* (ε_λ) et *lambda dilatation* (δ_λ), correspondent strictement à des érosions et dilatations morphologiques avec des éléments structurants non plats. Néanmoins, l'équivalence établie montre que, du point de vue pratique, il est également possible de les calculer sur la base des éléments plats.

Cette dernière formulation des nivellements est une extension de celle faite par Matheron dans le Critère 1 pour les nivellements plats. Notons toutefois que dans le cas où $\lambda = 0$ nous retrouvons les nivellements plats, tandis que pour des valeurs de $\lambda > 0$ les zones crées auront une certaine pente. Ainsi, nous pouvons généraliser le critère de construction des nivellements en introduisant les opérateurs non plats,

- Dans $\{g < f\}$ on doit remplacer g par $\delta_\lambda g$ jusqu'à la création d'une zone quasi-plate, ou jusqu'à ce que la fonction g' soit égale à la fonction f.

- Dans $\{g > f\}$ on doit remplacer g par $\varepsilon_\lambda g$ jusqu'à la création d'une zone quasi-plate, ou jusqu'à ce que la fonction g' soit égale à la fonction f.

Ainsi, la fonction marqueur g restera inchangée si, dans un des intervalles où $\{g < f\}$ ou $\{g > f\}$, la pente entre des pixels voisins ne dépasse pas la valeur de λ.

La Figure 3.11 permet de comparer le lambda nivellement par rapport à l'approche plate montrée auparavant dans la Figure 3.9.

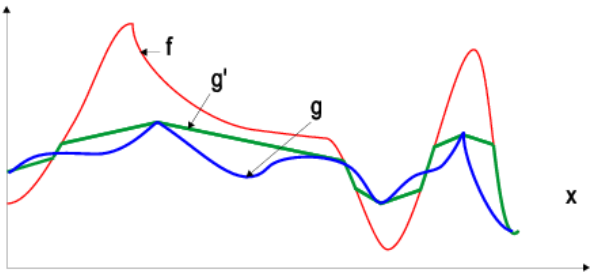


FIG. 3.11 – Construction du lambda nivellement.

Illustrant un exemple sur une image à niveau de gris, la Figure 3.12 compare le nivellement obtenu lorsque les zones plates strictes sont considérées, ou lorsque les zones quasi-plates sont acceptées. Pour rehausser les traits distinctifs parmi ces deux approches, nous avons montré une coupure horizontale sur l'image de départ et chacune des images nivelées. Ainsi on observe que la plus grande souplesse des lambda nivellements mène à créer de plus larges zones homogènes parmi les régions faiblement texturées.

Sur ce même exemple, nous avons comparé la taille des zones plates de l'image de référence par rapport à celles des images nivelées avec $\lambda = 0$ et $\lambda = 1$. Remarquons que, étant donné l'image de référence bruitée, les zones strictement plates se réduisent, presque dans toute leur totalité, à des pixels isolés. Par contre, les nivellements, en supprimant les pics les plus abrupts permettent aux zones plates de s'étendre en suivant des chemins de faible pente. Nous verrons dans le chapitre suivant que cette caractéristique fait des lambda nivellements des filtres très performants pour des applications orientées vers la segmentation.

Après avoir introduit la possibilité d'élargir la classe des nivellements en définissant de nouvelles zones plates, nous verrons par la suite qu'il est aussi possible d'en dériver d'autres opérateurs connexes. Ainsi, nous définirons les inondations étant de nivellements extensifs, et les arasements, par dualité, étant anti-extensifs.

Définition 11 (Inondation) *Soient f et g deux fonctions d'un treillis T. Une fonction g est une inondation de la fonction f si et seulement si, $g \geq f$ et*

$$\forall (p,q)\ voisins,\ \ g_p > g_q \Rightarrow g_p = f_p$$

Les inondations sont des nivellements extensifs qui élargissent les zones plates d'une fonction par fermeture morphologique. Cette définition peut être facilement expliquée par analogie avec le processus d'inondation d'un relief topographique comme celui de la Figure 3.13 (a).

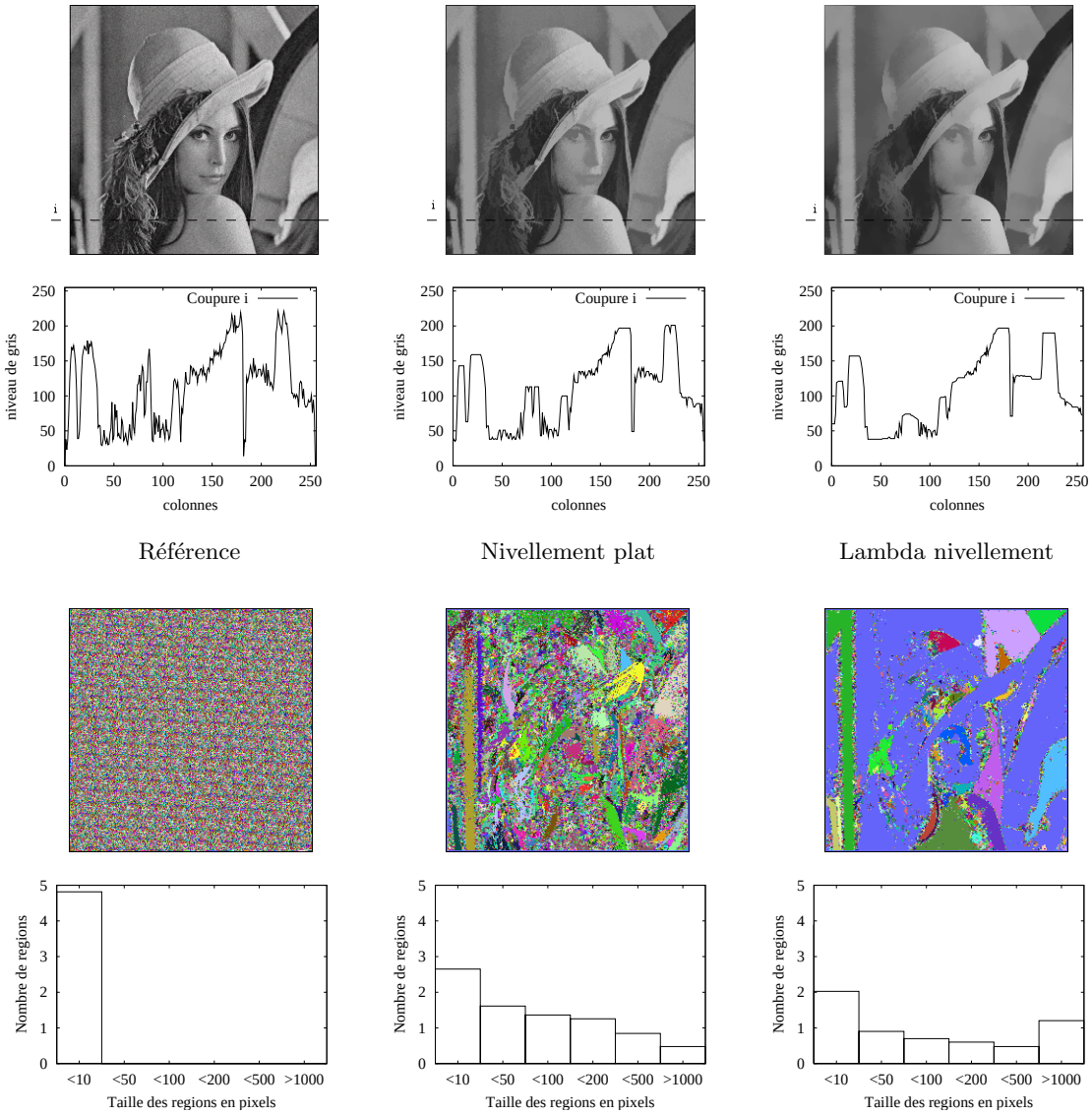


FIG. 3.12 – Extension des nivellements aux zones quasi-plates.

En concordance avec la terminologie cartographique, nous appellerons *lac* chacune des composantes connexes de g obtenues par inondation d'un *bassin versant* de f. Dans ce cas, il est simple de voir qu'un lac est une zone plate contenant au moins un pixel p dont $g_p > f_p$. Par ailleurs nous distinguerons deux types de lacs :

– les lacs confinés au fond d'une vallée, dont tous les pixels voisins sont à une plus grande hauteur. Dans ce cas, l'inondation élargit la taille d'un minimum régional de g,

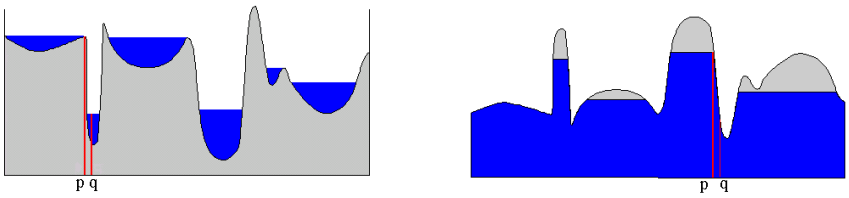
– les lacs pleins qui ont complètement inondé leur vallée, dont l'eau est montée jusqu'au col voisin le plus bas. Dans ce cas, l'inondation supprime un minimum de la fonction g.

Sur des images monochromes, la hauteur du relief équivaut au niveau de gris de la fonction ; les bassins versants sont associés aux zones foncées, tandis que les cols représentent les objets les plus clairs. Par conséquent, l'inondation des bassins versants revient à couvrir les zones foncés avec des niveaux de gris plus clairs.

Il est également possible de définir les opérateurs duaux comme,

Définition 12 (Arasement) *Soient f et g deux fonctions d'un treillis T. Une fonction g est un arasement de la fonction f si et seulement si, $g \leq f$ et*

$$\forall (p,q)\ voisins,\ \ g_p > g_q \Rightarrow g_q = f_q$$



(a) Inondation (b) Arasement

FIG. 3.13 – Cas particuliers des nivellements.

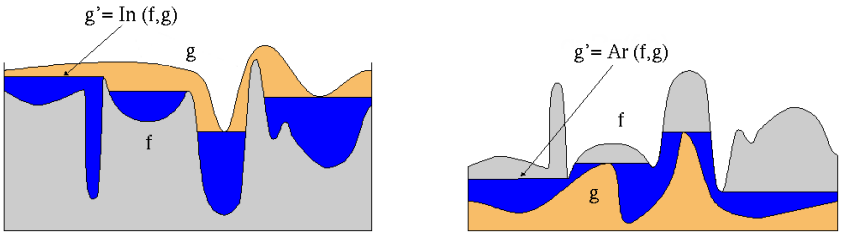
Lorsque ces opérateurs sont contraints par une deuxième fonction, nommée *fonction marqueur*, ils donnent lieu à des fermetures et des ouvertures par reconstruction :

- L'**inondation** contrainte par une fonction marqueur étant croissante, extensive et idempotente, est une **fermeture par reconstruction**.

- L'**arasement** contraint par une fonction marqueur étant croissant, anti-extensif et idempotent, est une **ouverture par reconstruction**.

La fonction marqueur limite l'inondation (arasement) de la fonction originale. En d'autre termes, il s'agit de calculer la plus forte inondation (arasement) qui reste en dessous (au dessus) de la fonction marqueur. Dans la Figure 3.14 nous avons illustré un exemple où l'inondation (arasement) g' de f est contrainte par g.

Ces opérateurs ont été largement utilisés comme des outils de filtrage car, grâce à la présence d'une fonction marqueur, ils peuvent supprimer certains pics du signal tout en laissant d'autres inchangés.



Fermeture par reconstruction Ouverture par reconstruction

FIG. 3.14 – Inondations et arasements contraints à une fonction marqueur.

3.3 Les nivellements

Dans la section précédente, nous avons défini les filtres connexes, et parmi ceux-ci, nous avons introduit les nivellements. Sans vouloir entrer dans une analyse détaillée de toutes leurs propriétés, notre approche visait seulement à fonder les notions théoriques sous-jacentes à cette nouvelle classe d'opérateurs.

Après avoir entrevu leur bonnes qualités, nous sommes à présent en situation d'avancer que les nivellements, étant de puissants outils de filtrage, constitueront une partie essentielle de notre chaîne de segmentation.

Pour cette raison, nous allons dédier entièrement cette section à leur étude du point de vue pratique : depuis la mise en œuvre des algorithmes jusqu'à l'analyse des applications qui en découlent.

3.3.1 Mise en œuvre

Comme point de départ théorique nous allons reprendre ici la formulation des nivellements établie dans le Critère 1 (page 27). Rappelons-la brièvement : nous dirons que g est un nivellement de la fonction f si pour tout pixel de l'image se vérifie la relation,

$$f \wedge \delta g \leq g \leq f \vee \varepsilon g$$

A partir de cette formulation il est aisé de dériver un algorithme pour construire le plus fort nivellement de f contraint par une fonction marqueur g. Il s'agirait de modifier progressivement la valeur de g sur chacun des pixels qui ne vérifient pas le critère établi. En pseudo-code ce processus pourrait s'exprimer de la façon qui suit,

faire :
- si $f \wedge \delta g > g$ prendre $g = f \wedge \delta g$
- si $g < f \vee \varepsilon g$ prendre $g = f \vee \varepsilon g$

jusqu'à convergence

ou également dans une seule opération comme,

faire :
$g = (f \vee \varepsilon g) \wedge \delta g$
jusqu'à convergence

Notons que l'édition du marqueur se poursuivra jusqu'à ce qu'aucun changement n'ait plus lieu dans la dernière itération. Cela signifie qu'à la fin du processus, g vérifiera le critère des nivellements par rapport à f dans tous les pixels de l'image.

Dans la Figure 3.15 nous avons illustré ce processus sur un exemple en une dimension. Nous montrons la première étape de l'algorithme, ainsi que le nivellement résultant. Conformément à notre notation, f symbolise la fonction de référence, et g la fonction marqueur. A chacun des pixels (a,b,c,...) nous avons associé une valeur dans une échelle de 1 à 8.

A chaque itération, avant de procéder à la transformation de la fonction marqueur, nous allons construire deux fonctions auxiliaires : l'une contenant l'érosion de g, et l'autre sa dilatation. Ensuite, là où la fonction g est au dessus de la référence, nous allons faire $g = f \vee \varepsilon g$. Au contraire, là où g est en dessous de f nous prendrons $g = f \wedge \delta g$.

Remarquons que l'ordre de ces opérations n'a aucune influence sur le résultat. On peut observer cette transformation de plus près sur les pixels (e) et (i) :

– comme $g_e < f_e$, g_e doit prendre la plus petite des valeurs parmi $\delta g_e = 4$ et $f_e = 6$, soit 4.

– comme $g_i > f_i$, g_i doit prendre la plus grande des valeurs parmi $\epsilon g_i = 4$ et $f_i = 2$, soit 4.

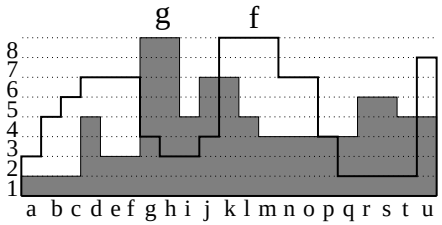
Nous allons conclure cette mise en œuvre en proposant trois possibilités concernant l'implémentation des nivellements. Pour chacune d'elles nous allons faire le point sur ces principaux atouts mais aussi sur ces inconvénients. Notre but ici est d'évaluer la viabilité de l'usage des nivellements dans le cadre des applications en temps réel :

Implémentation itérative

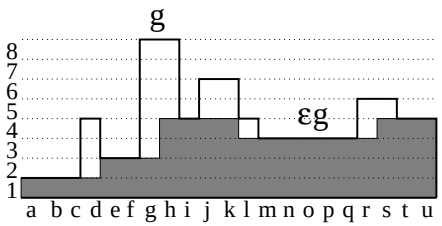
C'est la mise en œuvre la plus simple, mais aussi la plus lente. Elle reprend l'algorithme tel qu'il a été formulé auparavant en itérant le processus d'édition de l'image marqueur. A chaque étape, toute l'image est balayée pixel à pixel, le résultat étant gardé dans une image de sortie. Lorsque cette image de sortie est égale à l'image marqueur, le processus a fini. Autrement, l'image de sortie est prise comme marqueur dans l'itération suivante. Notons que dans les dernières itérations, même si toute l'image est balayée, très peu des pixels changent leur valeur.

Implémentation par des files d'attente hiérarchiques

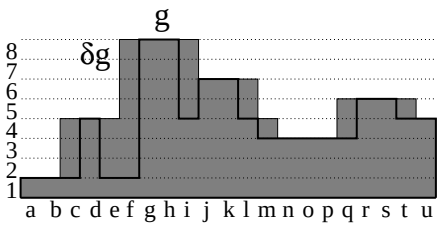
Ici nous cherchons à réduire le temps de calcul du processus itératif en introduisant une structure de données particulière connue comme la file d'attente hiérarchique. Il s'agit d'éviter



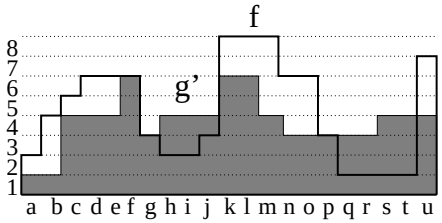
Entrée de l'algorithme :
f=fonction de référence, g=fonction marqueur



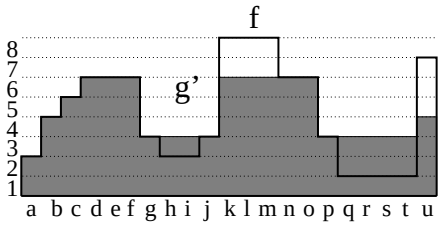
Erosion du marqueur



Dilatation du marqueur



Première itération : $g' = (f \vee \varepsilon g) \wedge \delta g$



Nivellement résultant

FIG. 3.15 – Illustration du processus de nivellement.

l'accès aux pixels qui ont déjà leur valeur finale, et traiter seulement ceux qui sont encore susceptibles de changer. Pour cela l'algorithme doit incorporer une première étape d'initialisation dans laquelle les pixels non conformes sont détectés et gardés dans la file d'attente.

Bien qu'il existe d'autres critères, nous proposons d'initialiser la queue avec tous les pixels ayant au moins un voisin qui ne leur permet pas de vérifier le critère des nivellements. Par la suite, pendant le processus d'édition du marqueur, l'accès aux pixels ne se fera plus par balayage systématique de l'image, mais en sortant les pixels gardés en file d'attente hiérarchique. La particularité de celle-ci étant de sortir les pixels en ordre décroissant lorsque le marqueur est en-dessous de la référence, et en ordre croissant dans le cas symétrique. Ceci évite de traiter plusieurs fois un même pixel.

Chaque fois que la valeur d'un pixel est modifiée, la validité du critère est à nouveau mise en question pour chacun de ces voisins. Ainsi, tous ceux qui ont une valeur non conforme rentrent à leur tour dans la file d'attente. De cette façon, les changements sur l'image marqueur se font en suivant un front de propagation. Le processus s'arrête lorsque la file d'attente est vide.

Cet algorithme a aussi un coût, car l'accès aux pixels de l'image se fait de façon désordonnée, au fur et à mesure qu'ils sortent de la file d'attente. Cet accès à la mémoire est beaucoup plus lent que lorsque la récupération de donnés se fait de façon séquentielle.

Pour cette raison, la construction d'un algorithme mixte serait envisageable, qui commencerait en itérant le processus récursif jusqu'à ce que le nombre de changements diminue suffisamment pour rendre plus rapide l'accès aux pixels par file d'attente.

Mais par ailleurs, dans une architecture dédiée, il serait aussi possible de diviser l'algorithme en deux parties qui pourraient se traiter en parallèle. Ainsi, nous allons proposer une dernière variante.

Implémentation en parallèle

Ici le nivellement d'une image de référence va être obtenu par deux algorithmes s'exécutant en parallèle : l'un nivelle les pixels pour lesquels le marqueur est en dessous de la référence $\{g < f\}$, tandis que l'autre agit sur les zones complémentaires où $\{g > f\}$. Lorsque les deux processus finissent, leur résultats respectifs sont regroupés pour construire le nivellement global de la fonction de référence.

Néanmoins, si chacun de ces algorithmes prend l'image marqueur telle qu'elle est au départ, le résultat sera strictement un aplatissement. Pour se rendre compte de cela, il suffit de regarder l'exemple de la Figure 3.16, où coïncident une transition de la référence avec une transition en sens inverse de l'image marqueur.

Sur cet exemple, si nous calculons le nivellement de f contraint par g séparément dans les zones où $\{f > g\}$ et $\{f < g\}$, nous obtiendrions $g' = g$, ce qui n'est pas un nivellement de f, mais un aplanissement.

Pour calculer correctement le nivellement, nous devons modifier la fonction marqueur à l'entrée de chacun des nivellements partiels. Dans l'un des cas, là où $\{f > g\}$ nous prendrons

$g_0 = \delta g$, et ailleurs $g_0 = f$. Le nivellement agissant sur g_0 sera un arasement car pour tous les pixels $g_0 \leq f$. Dans l'autre cas, là où $\{f > g\}$ nous ferons $g_1 = \varepsilon g$, et ailleurs $g_1 = f$. Par dualité, le processus agissant sur g_1 sera une inondation.

Notons qu'on pourrait également remplacer g par $g' = (f \vee \varepsilon g) \wedge \delta g$, bien que cette approche ne tire pas profit du parallélisme de cette implémentation.

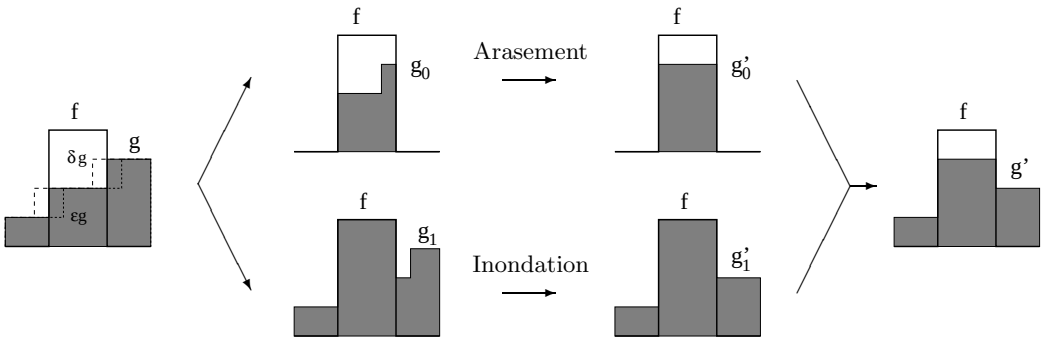


FIG. 3.16 – Implémentation des nivellements en parallèle.

Par la suite nous allons représenter chacune des implémentations proposées en pseudo-code. A l'entrée, nous disposons de l'image de référence f, ainsi que de l'image marqueur g. Pendant le processus de nivellement, l'image marqueur sera transformée jusqu'à ce qu'elle devienne un nivellement de f.

Dans la section suivante, nous verrons comment générer cette image marqueur et quelle est son influence sur le nivellement résultant.

◇ IMPLÉMENTATION ITÉRATIVE ◇

Processus
Faire :
Pour tout pixel p de l'image :

- Si $g_p > f_p$
  Pour tout pixel q voisin de p :
  - Si $g_q < g_p \rightarrow g'_p = f_p \vee g_q$
- Si $g_p < f_p$
  Pour tout pixel q voisin de p :
  - Si $g_q > g_p \rightarrow g'_p = f_p \wedge g_q$

Si $(g \neq g')$
Faire $g = g'$ et initier une nouvelle itération

◇ IMPLÉMENTATION PAR DES FILES D'ATTENTE HIÉRARCHIQUES ◇

Initialisation
Pour tout pixel p de l'image :
Pour tout pixel q voisin de p :
- Si $(g_p > g_q$ et $g_q < f_q)$ ou $(g_p < g_q$ et $g_q > f_q)$
  Garder p en file d'attente
  Passer aux pixel p suivant

Processus
Faire :
Extraire un pixel p de la file d'attente
Pour tout pixel q voisin de p :
- Si $g_p > f_p \rightarrow g_p = f_p \vee g_q$
- Si $g_p < f_p \rightarrow g_p = f_p \wedge g_q$

- Si $(g_p > g_q$ et $g_q < f_q)$ ou $(g_p < g_q$ et $g_q > f_q)$
  Garder q en file d'attente

tant que la file d'attente est non vide

◇ IMPLÉMENTATION EN PARALLÈLE ◇

Processus 0 $\rightarrow$
Construction du marqueur g_0
Calculer δg
Pour tout pixel p de l'image :
- Si $f_p \leq \delta g_p \rightarrow {g_0}_p = f_p$
- Si $f_p > \delta g_p \rightarrow {g_0}_p = \delta g_p$

Nivellement ${g'}_0$
Implémentation par des files d'attente (f, g_0)

Processus 1 $\rightarrow$
Construction du marqueur g_1
Calculer εg
Pour tout pixel p de l'image :
- Si $f_p \geq \varepsilon g_p \rightarrow {g_1}_p = f_p$
- Si $f_p < \varepsilon g_p \rightarrow {g_1}_p = \varepsilon g_p$

Nivellement ${g'}_1$
Implémentation par des files d'attente (f, g_1)

Résultat
Pour tout pixel p de l'image :
- Si $f_p \geq g_p \rightarrow g_p = {g_0}_p$
- Si $f_p < g_p \rightarrow g_p = {g_1}_p$

3.3.2 Etude des applications

Comme le lecteur l'aura sûrement déjà constaté, l'image marqueur joue un rôle très important dans le processus du nivellement. Nous commencerons en soulignant que toute image peut être prise comme image marqueur. Néanmoins, nous verrons qu'un choix astucieux permet d'envisager de nouvelles fonctionnalités sur la base des nivellements dans un large domaine d'applications. Parmi celles-ci, nous avons étudié quelques-unes en détail, d'autres restent ouvertes à toute poursuite éventuelle de ce travail.

Les nivellements comme mesure de ressemblance

Plus que la simple différence parmi deux images, le processus lié au nivellement nous apporte plusieurs indices pour évaluer la ressemblance entre l'image prise comme marqueur et celle considérée comme référence :

- le *nombre d'itérations* nécessaires jusqu'à la convergence de l'algorithme nous informe de la distance existante entre leurs contours.

- le *résidu* accumule les changements produits sur l'image marqueur jusqu'à ce qu'elle soit complètement nivelée. L'importance de ces changements dépendront du fait qu'ils restent à proximité des contours, ou qu'ils s'étendent à la totalité des régions.

- la *différence* parmi les niveaux des gris de l'image de référence et ceux de l'image nivelée, montre la capacité de l'image marqueur de récréer les valeurs de la référence. Car le nivellement porte à terme un recalage des contours, cette mesure est très robuste face aux petits déplacements.

De plus, on doit remarquer que ces mesures de ressemblance ne sont pas symétriques, car il est possible qu'une des images prise comme marqueur soit capable de mieux reproduire les détails de l'autre prise comme référence. Pour se rendre compte de cela il suffit de s'imaginer deux images, dont l'une est une simplification de l'autre. L'image originale, ayant toute l'information, sera capable de recréer une version simplifiée d'elle-même. Par contre, l'image déjà simplifiée ne pourra jamais récupérer l'information perdue.

Ainsi, lorsque nous voulons comparer deux images, nous pouvons procéder au calcul de deux nivellements par échange des rôles d'image de référence et d'image marqueur.

Illustré sur la Figure 3.17 nous avons un exemple de nivellement croisé sur deux images complètement différentes (A,B). Le nivellement C a été calculé en prenant l'image A comme référence et l'image B comme marqueur, tandis que dans un deuxième nivellement D les rôles ont été inversés.

Dans les images résidu et différence les zones claires correspondent aux changements les plus forts. On remarque que quelque soit le marqueur, l'image nivelée préserve toujours les contours de l'image de référence.

Par contre, c'est l'image marqueur qui détermine les niveaux de gris qui vont se propager dans le processus de nivellement.

FIG. 3.17 – Nivellement croisé de deux images.

Les nivellements comme mesure de mouvement

Une deuxième utilisation se dégage toutefois, lorsque le nivellement est calculé parmi deux images d'une même séquence. Dans ce cas-là, nous verrons que certaines informations de mouvement apparaissent liées à l'évolution du marqueur. Ainsi, si on compare la référence avec le marqueur, on distingue facilement deux types de zones différents :

- zones où le marqueur et la référence suivent la même évolution.
- zones où le contour de marqueur se décale fortement par rapport à celui de la référence.

Les premières englobent les petites variations causées par le bruit, les dernières correspondent normalement à des mouvements dans la scène.

On peut facilement observer cela sur la coupe verticale des deux images d'exemple de la Figure 3.18. La valeur 0 correspond au bord supérieur de l'image, la valeur 144 au bord inférieur. En comparant ces deux signaux on aperçoit rapidement, parmi de petites déviations, un fort décalage correspondant au mouvement d'un doigt.

A ce stade il serait possible de suivre, à chaque itération du processus de nivellement, le déplacement des contours du marqueur vers ceux de la référence. Le sens de ces propagations, ainsi que le nombre d'itérations nécessaires afin que chaque contour rejoigne son correspondant, définissent un vecteur de mouvement.

Une étude plus approfondie sur la pertinence de ces vecteurs en comparaison avec d'autres techniques reste à faire.

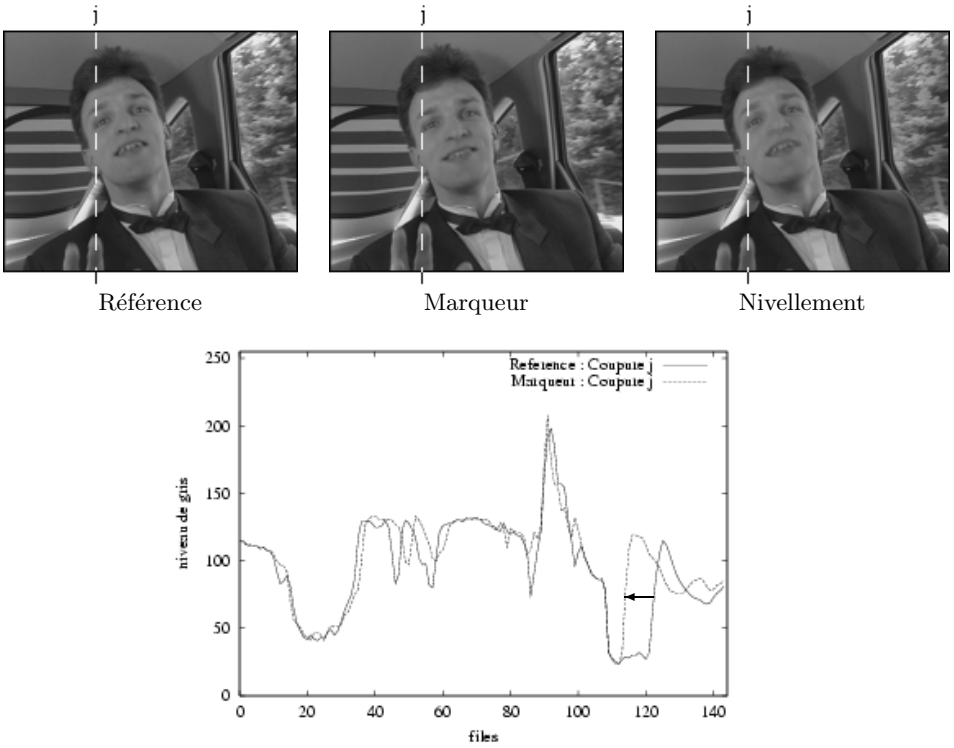


FIG. 3.18 – Etude du mouvement par nivellement de deux trames.

Les nivellements comme outils de filtrage

Celle-ci est probablement l'application la plus immédiate qu'on peut déduire à partir des nivellements. Dans ce but, pour construire la fonction marqueur g il faut avoir à l'esprit que : les maxima (minima) de la référence englobant un maximum (minimum) de g de valeur égale ou supérieure (inférieure), seront préservés. Autrement, par inondation ou arasement, ils deviendront une plus large zone plate.

En pratique, lorsqu'il s'agit de filtrer uniformément une image, on prend comme marqueur une simplification encore plus forte que celle désirée comme nivellement.

Sur une image de la rétine, la Figure 3.19 montre quelques exemples de création de marqueurs en utilisant des filtres classiques : a) filtrage alterné séquentiel ; b) filtre médian ; c) filtrage gaussien. Les images en dessous correspondent au résultat du nivellement. On remarque facilement que l'image originale a été simplifiée, et cependant la précision des contours est préservée.

Néanmoins, dans certains cas, nous voudrions pouvoir spécifier des zones de simplification, ainsi que des zones à préserver. Cela demande une création plus élaborée de l'image marqueur.

Ainsi, en continuant l'exemple précédant, nous avons repris quelques opérateurs bien connus dans le milieu des filtres : d) fermeture de contraste obtenue par inondation de la référence ; e) de la même manière, nous avons les ouvertures de contraste dont l'image originale est diminuée d'une constante H. Le but de ces filtres est d'éliminer les composantes à faible contraste. Finalement dans f) nous avons illustré un exemple de reconstruction par marqueurs. Comme tels, nous avons choisi les minima du signal de taille supérieure à un pixel.

Observer que grâce à cette démarche l'image nivelée ne contient plus les petites particules foncées.

Par ailleurs, en sachant que, plus la distance est large entre la référence et le marqueur, plus la simplification de l'image résultante est forte, il est possible de construire des simplifications multiéchelles comme celle de la Figure 3.20.

Les deux niveaux montrés font partie d'une série de nivellements dont les marqueurs ont été créés par filtrage alterné séquentiel de taille croissante. Le marqueur de gauche correspond à une taille 2, et celui de droite à une taille 10.

En comparant les nivellements correspondants, on remarque que sur le premier, tous les objets sont encore reconnaissables bien que leur textures aient été lissées.

Par contre, sur le deuxième, la simplification est déjà beaucoup plus forte que celle d'auparavant, même certains des pics arasés ne correspondent pas au bruit mais à de petits objets (voire les tableaux accrochés au mur). Sur l'image nivelée, l'effet visuel d'une telle simplification nous fait croire que les petits détails sont graduellement absorbés par leur entourage. Cependant, les contours des objets encore présents sont parfaitement préservés.

Notamment, si on veut obtenir directement un lissage encore plus fort, il suffit de prendre un marqueur d'autant plus plat. En guise d'exemple dans la Figure 3.21, nous avons pris comme marqueur une partition de l'image originale.

Le nombre de régions contrôle l'ordre de la simplification. Chacune de ces régions est plate, ayant comme valeur la moyenne de l'image originale à l'intérieur de la région. Ce type de marqueur simplifie fortement l'intérieur des régions et préserve leurs contours.

Ainsi, lorsqu'on utilise comme marqueur une partition de l'espace, il serait également envisageable de préserver les valeurs de l'image originale à l'intérieur de certaines des régions afin de faire une simplification sélective.

Au long de cette thèse nous verrons que, pour les applications de suivi d'objets, les filtres

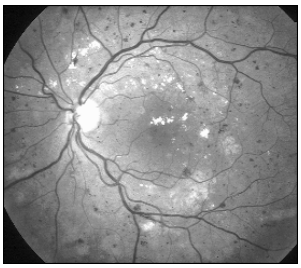
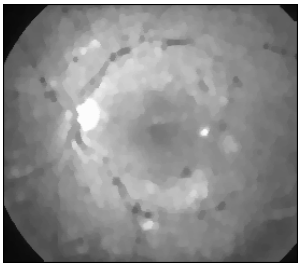
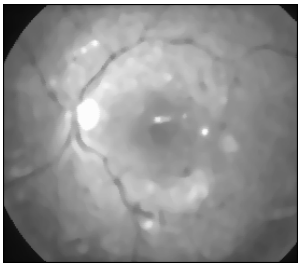


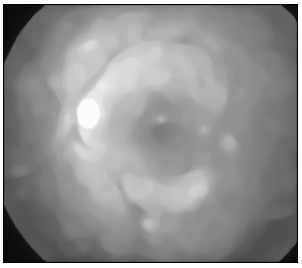
Image de référence



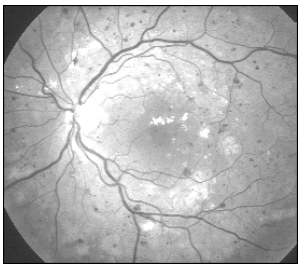
(a) FAS



(b) Filtre médian

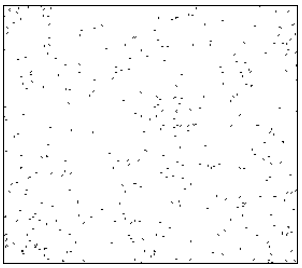


(c) Filtre gaussien



(d) Référence + H

(e) Référence - H



(f) Sélection des minima

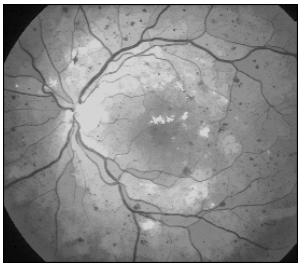


FIG. 3.19 – Exemples de marqueurs pour la simplification d'une image.

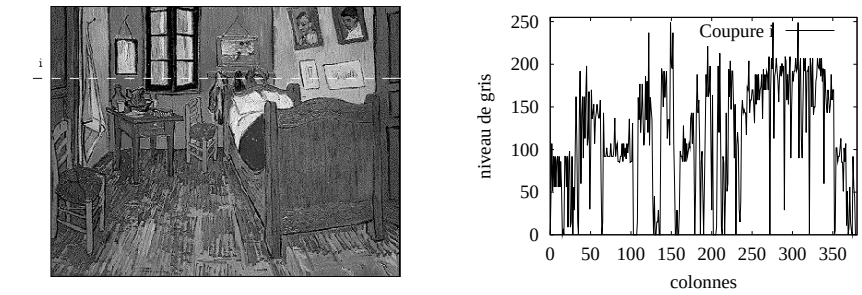
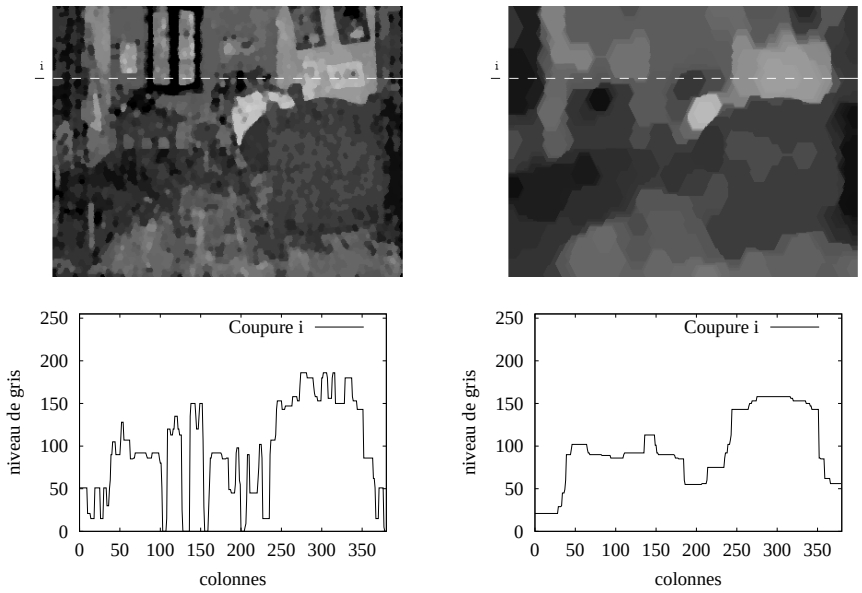
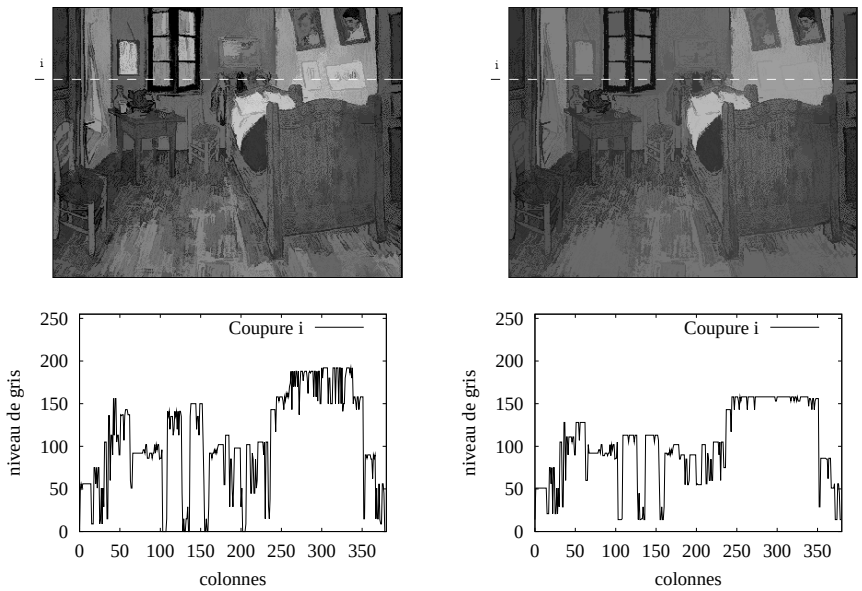


Image de référence



Images marqueurs



Images nivelées

FIG. 3.20 – Simplification multiéchelle. Image d'exemple : Chambre de Van Gogh à Arles. 1899. Musée d'Orsay ©, Paris.

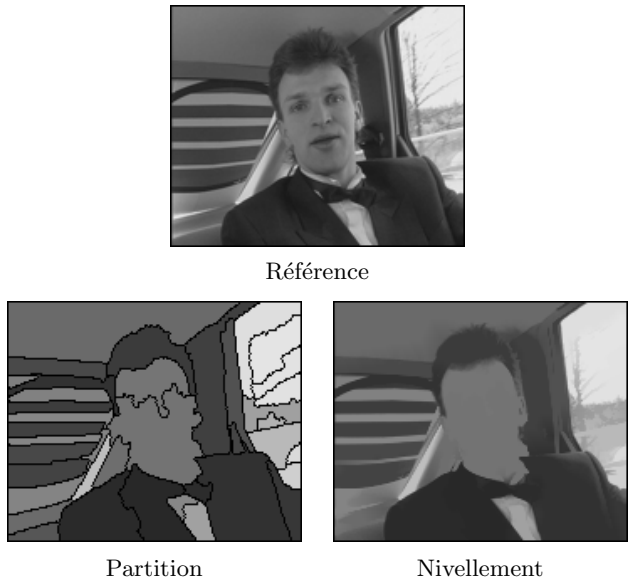


FIG. 3.21 – Simplification forte d'une image par nivellement.

visent à rehausser l'information liée aux objets d'intérêt. Là aussi les nivellements ont donné des résultats extrêmement intéressants. De plus, une amélioration considérable a été obtenue en exploitant la connaissance du masque précédent.

Pour illustrer par un exemple, nous avons traité dans la Figure 3.22 deux images d'une séquence routière. Le but de leur analyse est de détecter sur l'image courante les lignes blanches délimitant la voie de circulation qui ont été déjà détectées sur l'image précédente. Pour cela nous avons mis en œuvre un double nivellement calculé en échelle :

– Première nivellement : la référence correspond à l'image au temps (t) que nous voulons simplifier. L'image marqueur est construite en préservant les valeurs de l'image originale au temps (t) dans le masque à $(t-1)$ légèrement dilaté, et en prenant une valeur maximale ailleurs. Car le marqueur est au dessus de la référence, ce nivellement n'est qu'une inondation.

– Deuxième nivellement : il ne nous reste qu'à achever le processus dual en prenant comme image de référence le résultat de l'inondation. Le marqueur est généré de façon symétrique, donc la zone de valeur maximale devient maintenant noire. Dans ce cas-là, l'arasement assombrit les zones qui ne nous intéressent pas et rehausse par comparaison les bandes recherchées.

Dans ce dernier cas, nous avons vu comment les nivellements nous permettent de maximiser la simplification en dehors d'une zone d'intérêt, tandis qu'à l'intérieur de cette zone les valeurs de l'image de référence restent inchangées. De fait, nous reprendrons ces filtres dans l'application que nous avons développée dans la troisième partie de ce mémoire.

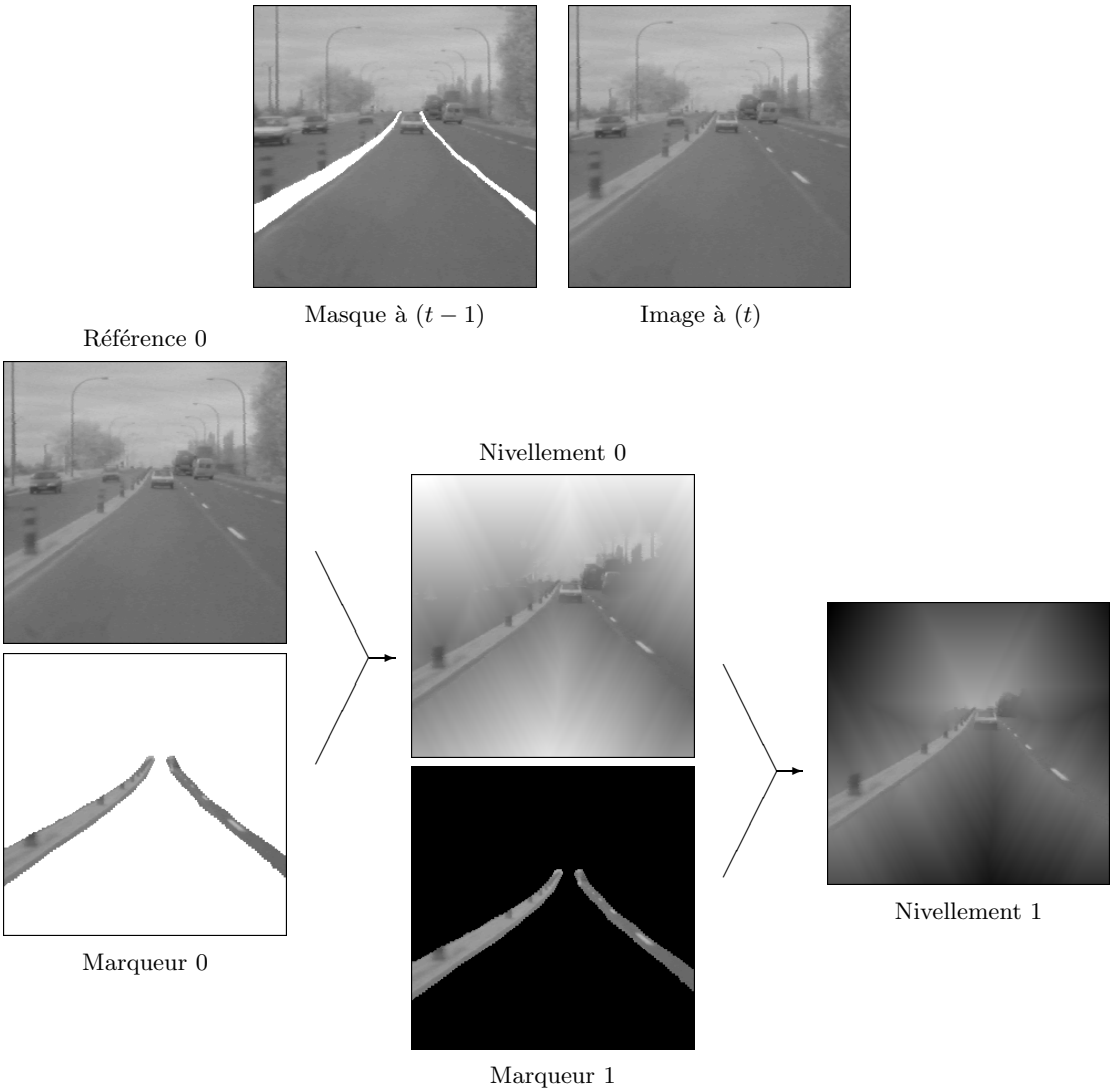


FIG. 3.22 – Nivellement adapté à un masque pour des applications de suivi d'objets.

3.4 Extension des nivellements aux espaces vectoriels

Couramment les images multispectrales ont été filtrées en appliquant des algorithmes scalaires indépendamment sur chacune des composantes. La solution finale étant alors obtenue par recombinaison de l'ensemble des composantes traitées. Cette même extension a été aussi faite en première instance pour les nivellements, dont Meyer a établi une interprétation géométrique sur la notion de rectangles minimaux [65].

Illustrant un exemple, la Figure 3.23 montre le nivellement d'une image vectorielle qui correspond à un champ de mouvement calculé par un algorithme de *Block Matching*. Cette technique, que nous verrons plus en détail dans la Section 10.3, découpe l'image au temps (t) en blocs de $N \times N$ pixels. Ensuite, pour chacun de ces blocs, elle propose comme vecteur de mouvement celui qui donne le meilleur emplacement du bloc dans l'image en $(t+1)$. Néanmoins, à cause du bruit et des faibles textures, les vecteurs de certains des blocs n'expriment pas leur vrai mouvement.

L'objectif d'une étape de filtrage est donc de supprimer ces erreurs, car si elles sont nombreuses par rapport à la taille d'un objet, elles peuvent empêcher la modélisation correcte du mouvement.

Le principal atout des nivellements par rapport à d'autres filtres, c'est qu'ils rendent uniforme le champ des vecteurs tout en laissant les frontières du mouvement inchangées.

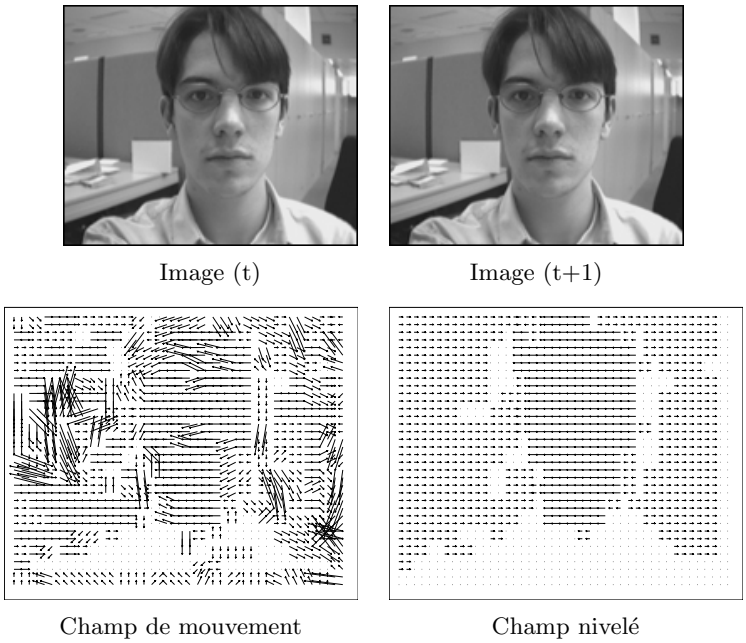
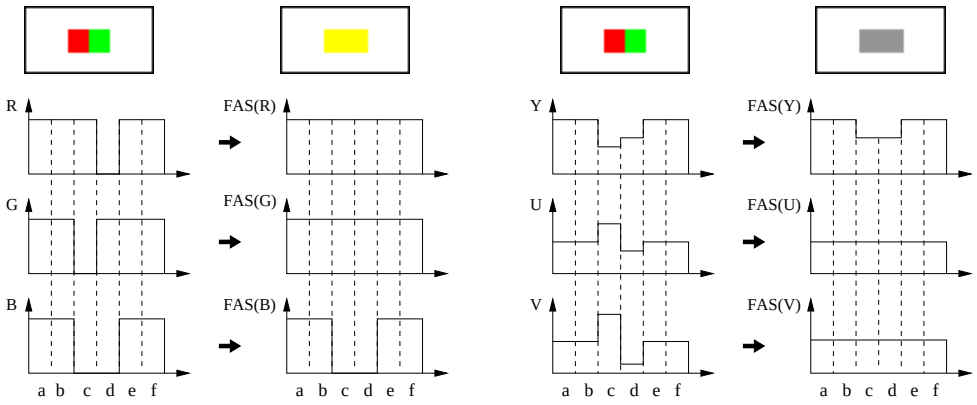


FIG. 3.23 – Nivellement d'un champ de mouvement.

Cependant, l'un des principaux inconvénients du filtrage vectoriel porté sur chacune des composantes est la possibilité d'introduire de nouveaux vecteurs dans les images filtrées. Ainsi, lorsqu'il s'agit d'images couleur, des tonalités qui n'étaient pas présentes sur les images de

départ peuvent apparaître dans les images filtrées.

La Figure 3.24 fournit un exemple sur une image de test synthétique ayant en voisinage un pixel rouge $\{c\}$ et un pixel vert $\{d\}$. A gauche, on retrouve la décomposition sur l'espace des couleurs RGB, ainsi que le résultat obtenu après filtrage alterné séquentiel de taille 1 de chacun des canaux. Il apparaît alors la couleur jaune des pixels $\{c, d\}$, couleur qui n'existait pas auparavant. En outre, ce résultat dépend de l'espace de couleur traité, comme il est montré dans l'exemple à droite. Ici l'image de test a été décomposée dans l'espace YUV, et on peut observer comment, dans ce cas-là, les pixels $\{c, d\}$ sont devenus gris. Ces mêmes effets apparaissent sur l'image couleur de la Figure 3.25 que nous avons choisie comme exemple.



Filtrage des composantes RGB — Filtrage des composantes YUV

FIG. 3.24 – Effets du filtrage sur la couleur.

Pour faire face à ces problèmes, des solutions plus adaptées sont apparues en tenant compte des corrélations existantes parmi les différents canaux couleur [3, 78]. Ainsi, la définition des filtres classiques a pu être élargie sur la base de différents critères de classement multi-variable établis par Barnett dans [4].

Egalement, différentes stratégies ont été proposées dans le domaine de la morphologie mathématique afin de surmonter l'absence d'un ordre absolu dans les espaces vectoriels. Mentionner la théorie de Wilson sur la morphologie des matrices [105] ; l'introduction à la morphologie mathématique multi-évaluée dans un treillis complet et partialement ordonné faite par Serra dans [88] ; et une nouvelle approche développée par Hanbury et Serra [40] définissant des opérateurs morphologiques cycliques sur le cercle chromatique.

Cependant, il n'existe pas un consensus général sur la définition des opérateurs morphologiques de base dans un espace multi-dimensionnel, ayant comme conséquence l'apparition de nombreuses méthodes de filtrage, dont chacune définit ses opérateurs sur l'heuristique d'un unique espace des couleurs.

Nous verrons par la suite que travailler en vectoriel laisse de multiples questions ouvertes car, s'il est bien clair que simplifier une image à niveaux de gris comporte exclusivement le lissage des textures, les avis sont partagés lorsqu'il s'agit de la simplification d'une image

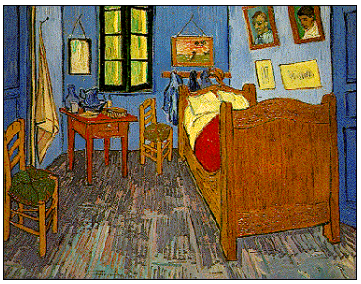


Image originale

Filtrage des composantes RGB

Filtrage des composantes YUV

FIG. 3.25 – Filtrage alterné séquentiel appliqué à chacune des composantes couleur.

couleur. Ainsi, l'introduction de nouvelles couleurs dans l'image filtrée peut être critique pour certaines des applications, mais elle peut aussi être négligeable pour d'autres.

Bien que le filtrage des images couleur restera l'une des principales applications, nous avons abordé l'extension des nivellements scalaires dans le but de fournir des algorithmes de filtrage généralisables à tout espace vectoriel. Finalement, nos recherches ont abouti à deux nouvelles approches [33] : l'une prend le nom de *pseudo-scalaire* car elle se base sur le nivellement d'un produit scalaire, alors que la deuxième est un véritable algorithme vectoriel.

Nous allons décrire ces deux techniques en considérant la fonction de référence $\vec{f}$ et la fonction marqueur $\vec{g}$ comme étant des fonctions vectorielles. Ensuite, nous conclurons en faisant une étude comparative de leur performances.

3.4.1 Les nivellements pseudo-scalaires

Les nivellements pseudo-scalaires procèdent en trois étapes : a) la première transforme l'image vectorielle en une image scalaire par projection des vecteurs sur un certain axe de l'espace ; b) l'image mono-dimensionnelle ainsi obtenue est ensuite nivelée par les algorithmes scalaires présentés auparavant ; c) finalement, le nivellement résultant est rétroprojeté dans l'espace vectoriel afin de récupérer sa dimension de départ. Chacune de ces étapes est détaillée par la suite :

Projection vectorielle : la première démarche du processus consiste à projeter par produit scalaire ordinaire la fonction de référence $\vec{f}$ sur un certain axe de l'espace que nous appellerons $\vec{v}$. Le choix de $\vec{v}$ n'est pas fixé par notre algorithme car il dépend du

cadre de l'application. S'il s'agit du filtrage des images couleur, $\vec{v}$ est pris sur la ligne achromatique afin de traiter toutes les couleurs équitablement. Rappeler que dans tout espace des couleurs la ligne achromatique est la ligne reliant la couleur noire au blanc. Ainsi nous avons $\vec{v}_{YUV} = (1, 0, 0)$, $\vec{v}_{RGB} = (1, 1, 1)$, $\vec{v}_{HLS} = (0, 1, 0)$,... En outre, dans le domaine de l'analyse de mouvement, de bons résultats ont été obtenus en prenant $\vec{v}$ comme le vecteur du mouvement global. Le résultat de cette étape est une fonction scalaire que nous désignerons par $\widehat{f} = \vec{f} \cdot \vec{v}$.

Nivellement scalaire : une fois que nous disposons d'une fonction de référence scalaire, la fonction marqueur $\widehat{g}$ peut être obtenue de deux façons différentes : soit par projection d'un marqueur vectoriel $\widehat{g} = \vec{g} \cdot \vec{v}$, soit par simple filtrage scalaire de la fonction de référence projetée $\widehat{g} = \Gamma(\widehat{f})$. A ce stade, l'image de référence $\widehat{f}$ ainsi que son marqueur $\widehat{g}$ sont deux fonctions scalaires. Par conséquent, on peut leur appliquer les nivellements scalaires tels qu'ils ont été décrits dans la Section 3.2.2, ce qui nous donne $\widehat{g}' = niv(\widehat{f}, \widehat{g})$.

Projection vectorielle inverse : la dernière étape de l'algorithme doit mener à terme la projection inverse du nivellement scalaire $\widehat{g}'$ dans l'espace vectoriel du départ. Notre but est de rendre une fonction vectorielle $\vec{g}'$ qui soit le nivellement de $\vec{f}$. Afin de préserver la propriété d'idempotence des nivellements scalaires, nous allons imposer que le processus de projection (directe + inverse) vérifie que :

$$\vec{f} = (P^{-1} \circ P)(\vec{f}) \tag{3.10}$$

Notons qu'une telle transformation peut créer des vecteurs qui n'étaient pas présents dans l'image originale. Cependant, l'importance de ce fait va dépendre du critère de projection inverse choisi. Celui que nous allons proposer a une propriété très attractive, car la simplification multi-échelle d'une image couleur achève un lissage de plus en plus fort des textures, ainsi qu'une diminution progressive de la saturation des couleurs. A chaque étape le résultat est une image nivelée plus lisse et avec des couleurs moins vives. La simplification extrême donne comme résultat une image complètement plate et monochromatique. Le critère mentionné peut se formuler comme suit (voir Figure 3.26) :

- Si $\widehat{g}' > \widehat{f}$, alors $\vec{g}'$ doit pointer sur le vecteur $(\vec{f} - \vec{w})$.
- Si $\widehat{g}' \leq \widehat{f}$, alors $\vec{g}'$ est placé sur $\vec{f}$.

dont $\vec{w}$ est le vecteur associé au blanc.

Cette méthode donne des résultats très proches les uns des autres indépendamment de l'espace couleur utilisé. A titre d'exemple, dans la Figure 3.27 nous avons montré les résultats obtenus par nivellement pseudo-scalaire d'une image couleur dans les espaces RGB et YUV. En faisant la différence pixel à pixel, on s'aperçoit que les nivellements ne sont pas identiques, bien que visuellement ils soient rarement différentiables.

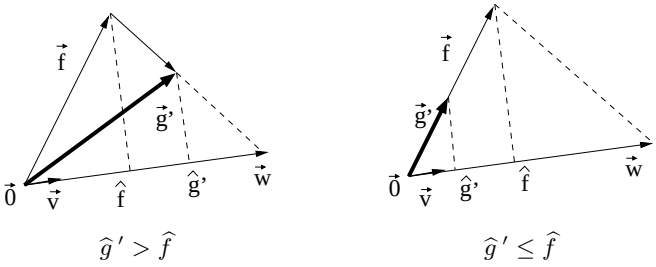


FIG. 3.26 – Projection inverse dans l'espace vectoriel.

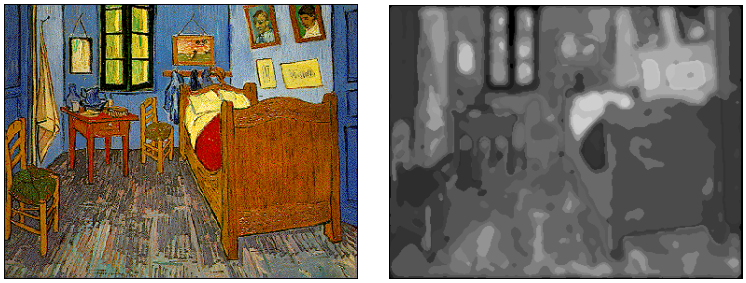


Image originale | Marqueur scalaire

Nivellement en RGB | Nivellement en YUV

FIG. 3.27 – Nivellements pseudo-scalaires appliqués à différents espaces des couleurs.

On doit remarquer ici que le marqueur est une fonction scalaire chargée de simplifier les textures mais sans aucune influence sur la teinte des couleurs de l'image nivelée. Celles-ci dépendent exclusivement de l'image prise comme référence, qui est simplifiée sans changer pour autant notre perception des couleurs.

Les images couleur ainsi nivelées héritent des caractéristiques des nivellements scalaires : une forte simplification des détails et des textures tandis que la précision des contours est préservée. Par contre, au niveau des zones de couleur plate, cette approche n'arrive pas à achever le degré de simplification du nivellement scalaire appliqué composante par composante.

3.4.2 Les nivellements vectoriels autarciques

Nous présenterons ici un algorithme complètement vectoriel créé par extension directe des nivellements scalaires. Autrement dit, nous allons reproduire le processus de construction d'un nivellement scalaire ayant comme données cette fois-ci, des vecteurs. De ce fait, nous caractériserons les nivellements vectoriels autarciques en partant d'une description intuitive des nivellements scalaires. Pour ces derniers, le nivellement sur un point p agit de la façon suivante :

- *Si toutes les valeurs des pixels de la fonction marqueur dans le voisinage N_p, sont « en dessous » (resp. au dessus) de la fonction marqueur, $g_q \leq f_p$ (resp. $g_q > f_p$), alors g_p acquiert la valeur « la plus haute » (resp. base) de g dans N_p.*

- *Autrement, s'il existe des pixels dans le voisinage N_p qui prennent des valeurs $g_s \leq f_p$ tandis que pour d'autres $g_t \geq f_p$, alors g_p prend la valeur de f_p.*

Les nivellements vectoriels autarciques découlent du même principe de fonctionnement. Etant donné deux fonctions vectorielles, l'une comme référence $\vec{f}$, l'autre comme marqueur $\vec{g}$, le nivellement vectoriel sur un point p agit de la façon suivante :

- *Si tous les vecteurs de la fonction marqueur dans le voisinage N_p sont « sur un même côté de » $\vec{f}_p$, alors $\vec{g}_p$ est remplacé par le vecteur marqueur « le plus proche de » $\vec{f}_p$.*

- *Autrement, $\vec{g}_p$ prend la valeur de $\vec{f}_p$.*

Afin de pouvoir poursuivre ce programme, il ne nous reste qu'à donner un sens aux expressions « *être le plus proche de* » et « *être sur le même coté de* » pour des vecteurs :

Etre le plus proche de : nous dirons que $\vec{g}_p$ est plus proche de $\vec{f}_k$ que $\vec{g}_q$ si et seulement si $\|\vec{g}_p - \vec{f}_k\| \leq \|\vec{g}_q - \vec{f}_k\|$, dont $\|\cdot\|$ représente la norme Euclidienne.

Etre sur un même côté de : nous considérerons que $\vec{g}_p$ et $\vec{g}_q$ se trouvent à un côté de $\vec{f}_k$ si et seulement si, l'angle $\widehat{g_p f_k g_q}$ est aigû. C'est-à-dire, si $\vec{f}_k$ n'appartient pas à la sphère ayant $\vec{g}_p$ et $\vec{g}_q$ comme extrémités du diamètre (voir Figure 3.28).

Ainsi, nous dirons que tous les vecteurs $\vec{g}_q$ de la fonction marqueur dans le voisinage N_k sont "à un coté de" $\vec{f}_k$, si et seulement si $\vec{f}_k$ reste en dehors de la sphère minimale contenant tous les $\vec{g}_q$. Autrement, $\vec{f}_k$ pourrait se trouver entouré par eux. Dans cette approche, le plan divisant l'espace en deux parties se courbe jusqu'à devenir une surface sphérique.

Les nivellements vectoriels autarciques ainsi définis peuvent être calculés de façon très efficace en considérant que les changements de la fonction marqueur dépendent exclusivement du signe d'un produit scalaire dans la forme qui suit :

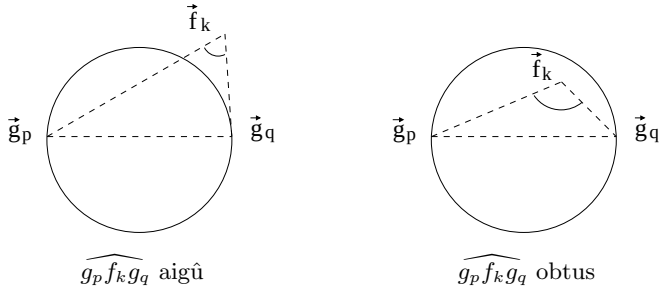


FIG. 3.28 – Angle compris entre le vecteur de référence et les vecteurs marqueurs.

$$\vec{g}_k{}' = \begin{cases} \vec{f}_k & si \quad (\vec{g}_p - \vec{f}_k) \cdot (\vec{g}_q - \vec{f}_k) < 0 \\ \vec{g}_i & si \quad (\vec{g}_p - \vec{f}_k) \cdot (\vec{g}_q - \vec{f}_k) \geq 0 \end{cases} \tag{3.11}$$

dont, parmi les vecteurs du marqueur dans le voisinage N_k, $\vec{g}_p$ et $\vec{g}_q$ qui donnent lieu à la sphère de diamètre maximale, et $\vec{g}_i$ correspond au vecteur le plus proche à $\vec{f}_k$.

Ce critère n'est pas le seul valable comme extension vectorielle des nivellements scalaires, et laisse la porte ouverte à de multiples variations. Cependant notre approche a de très bonnes propriétés vis à vis du filtrage des images vectorielles. Les nivellements vectoriels autarciques :

- permettent d´établir une analogie avec le processus de nivellement scalaire : ils préservent la propriété d'idempotence, et ils aboutissent aux mêmes résultats lorsqu'ils s'appliquent aux images à niveau de gris.

- sont valables dans tout espace vectoriel et, par conséquent, ils s'appliquent correctement au filtrage des images couleur indépendamment de l'espace utilisé.

De cette façon, le nivellement des images couleur se traduit tout simplement en un ensemble de déplacements des vecteurs couleur présents dans le voisinage, soit de l'image de référence, soit de l'image marqueur. L'opérateur ainsi créé mérite le nom de nivellement vectoriel autarcique puisqu'il n'introduit jamais de nouveaux vecteurs lorsqu'il élargit les zones plates. Les objets encore présents dans l'image marqueur apparaîtront dans l'image nivelée comme étant une version simplifiée des objets originaux. Autrement, ils seront couverts par leur entourage. La Figure 3.29 montre un exemple, dont l'image marqueur a été générée par filtrage médian vectoriel [3].

3.4.3 Etude comparative des performances

Nous voudrions finir ce chapitre en faisant une étude comparative des performances achevées pour les nivellements lorsqu'ils s'appliquent au filtrage des images couleur. Les approches analysées sont : les nivellements scalaires appliqués composante par composante, les nivellements pseudo-scalaires, et les nivellements vectoriels autarciques.

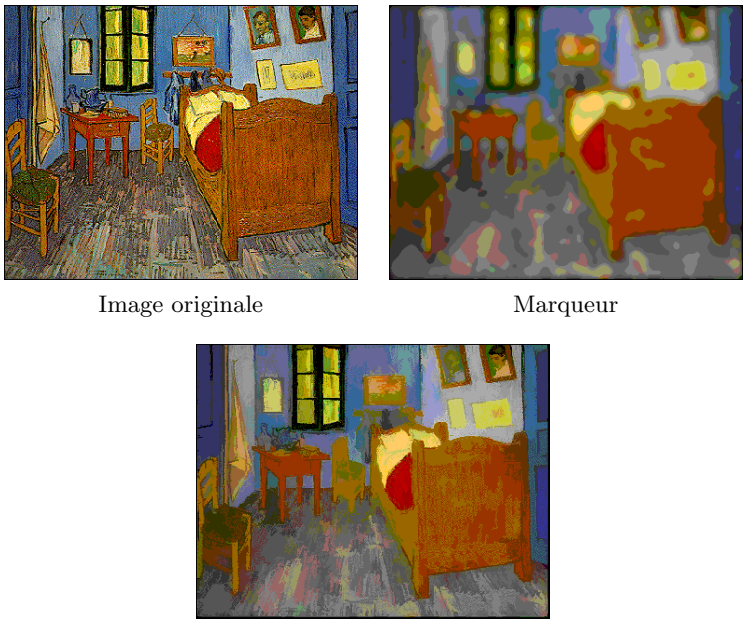


Image originale | Marqueur

Nivellement

FIG. 3.29 – Nivellement vectoriel autarcique d'une image couleur.

Pour chacune de ces techniques nous avons fait ressortir les atouts et les défauts au niveau de leur traitement de la couleur, de leur capacité de simplification, ainsi que de leur temps de calcul. Nos commentaires reposent sur l'expérience acquise. Cependant, le lecteur pourra évaluer par lui-même certaines des caractéristiques citées, sur les images d'exemple des Figures 3.30 et 3.31.

Traitement de la couleur

Nivellements scalaires : nous avons vu qu'ils peuvent introduire des couleurs parasites très voyantes dans l'image filtrée. Cependant ce comportement est rare lorsqu'on part d'une image marqueur qui reste fidèle aux couleurs de l'image originale. Par ailleurs, l'un des inconvénients d'une telle façon de procéder est lié à la variabilité des résultats en fonction de l'espace de couleurs traité.

Nivellements pseudo-scalaires : dans cette approche, l'ensemble des projections fait varier les vecteurs de l'image de référence, ce qui implique strictement l'introduction de nouvelles couleurs dans l'image filtrée. Par contre, du côté visuel, cette technique est très appréciée car le processus de simplification agit sur les textures mais aussi sur les couleurs, en les rendant moins vives. De plus, cette méthode donne des résultats très proches les uns des autres indépendamment de l'espace de couleurs utilisé.

Nivellements vectoriels autarciques : cette technique a la propriété de ne jamais utiliser des couleurs inexistants dans l'image de référence ou dans l'image marqueur de départ.

Néanmoins, lors de fortes simplifications, il est frappant de voir sur l'image nivelée que la couleur d'un objet puisse se propager à d'autres régions voisines.

Elargissement des zones plates

Nivellements scalaires : ici la simplification doit être évaluée après l'intersection des zones plates sur l'ensemble des composantes traitées. Néanmoins, le fait de pouvoir caractériser les zones d'homogénéité séparément pour chacun des canaux couleur nous mène, dans certains cas, à des niveaux de simplification comparables à ceux des approches purement vectorielles.

Nivellements pseudo-scalaires : l'algorithme, tel qu'il a été défini, simplifie l'image en appliquant un seul nivellement scalaire à l'une des dimensions de l'espace. Par conséquent, ils n'arrivent pas à atteindre le degré de simplification des nivellements scalaires. Remarquons seulement que cette procédure, ainsi que celle purement scalaire, peuvent adopter toutes les extensions faites pour les zones plates sans qu'aucune reformulation des algorithmes ne soit nécessaire.

Nivellements vectoriels autarciques : sans l'ombre d'un doute, ceux-ci créent les plus larges zones plates. Cette propriété peut être expliquée à partir de la définition des nivellements elle-même. Le processus du nivellement vectoriel élargit les zones plates tout simplement en propageant inchangés les vecteurs de couleur présents sur l'image originale.

Temps de calcul

Nivellements scalaires : cette approche, appliquant les nivellements composante par composante, est rapide et hautement parallélisable. On peut profiter également de toute optimisation faite auparavant pour les images monochromes. Cependant, la plus forte contrainte est liée au temps de calcul de l'image marqueur lors des applications visuelles, dont on doit assurer la qualité des couleurs sur l'image nivelée.

Nivellements pseudo-scalaires : ceux-ci sont de loin les algorithmes les plus rapides car le calcul d'un seul nivellement scalaire est nécessaire. Notons que sur les espaces de couleur ayant une composante sur l'axe achromatique, seule la projection inverse doit être calculée. De plus, l'image marqueur peut être obtenue par simple filtrage scalaire, car la couleur sur elle n'a aucune importance sur le nivellement résultant.

Nivellements vectoriels autarciques : du point de vue du temps de calcul les nivellements purement vectoriels sont en clair désavantage par rapport aux autres. Ceci est dû à la taille des données à traiter ainsi qu'à l'impossibilité de les paralléliser. Par conséquent, tout usage dans des applications en temps réel ne serait pas envisageable.

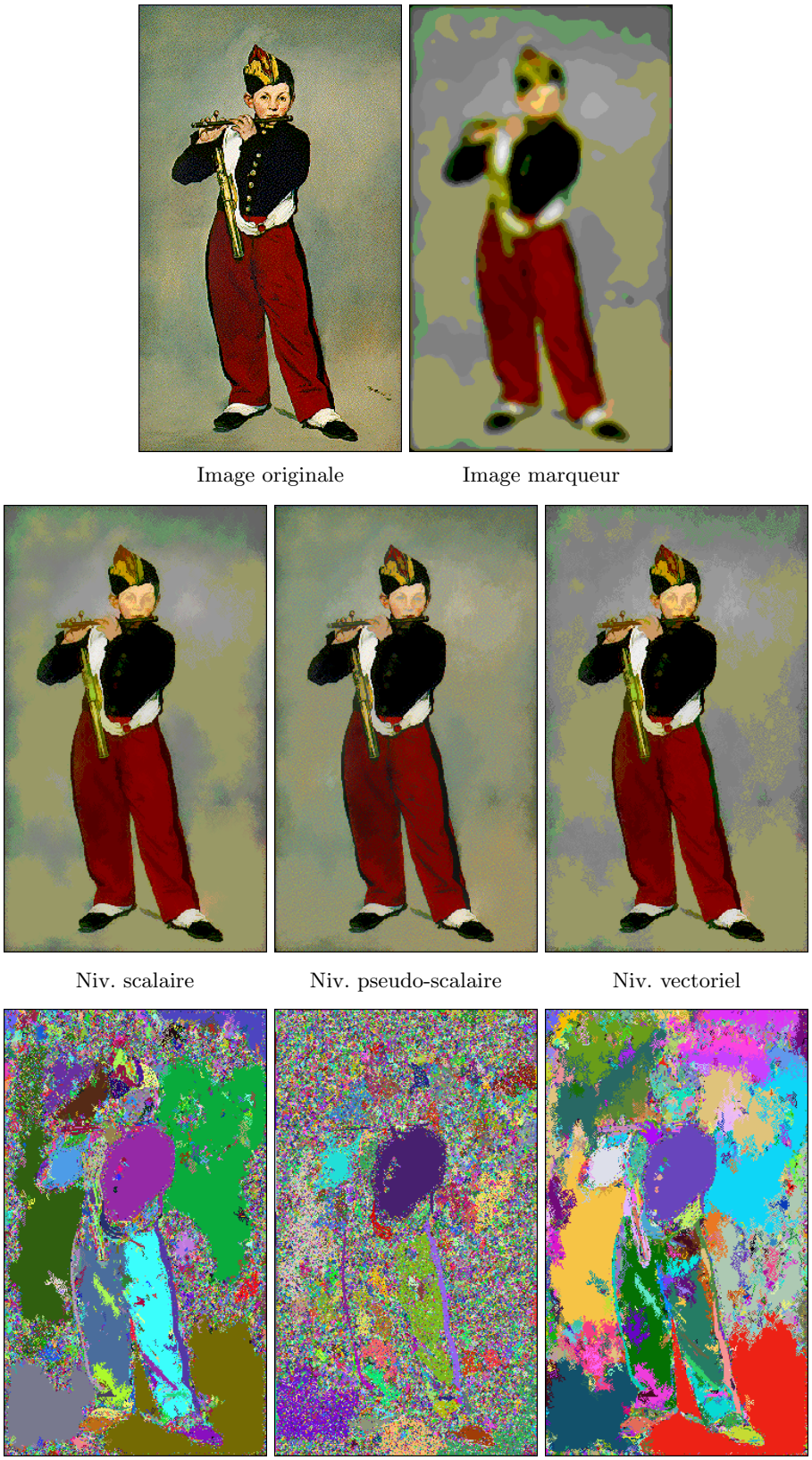


FIG. 3.30 – Comparaison des différents algorithmes de nivellement lorsqu'ils s'appliquent au filtrage des images couleur. Image d'exemple : Le fifre. Edouard Manet, 1866. Musée d'Orsay ©, Paris.

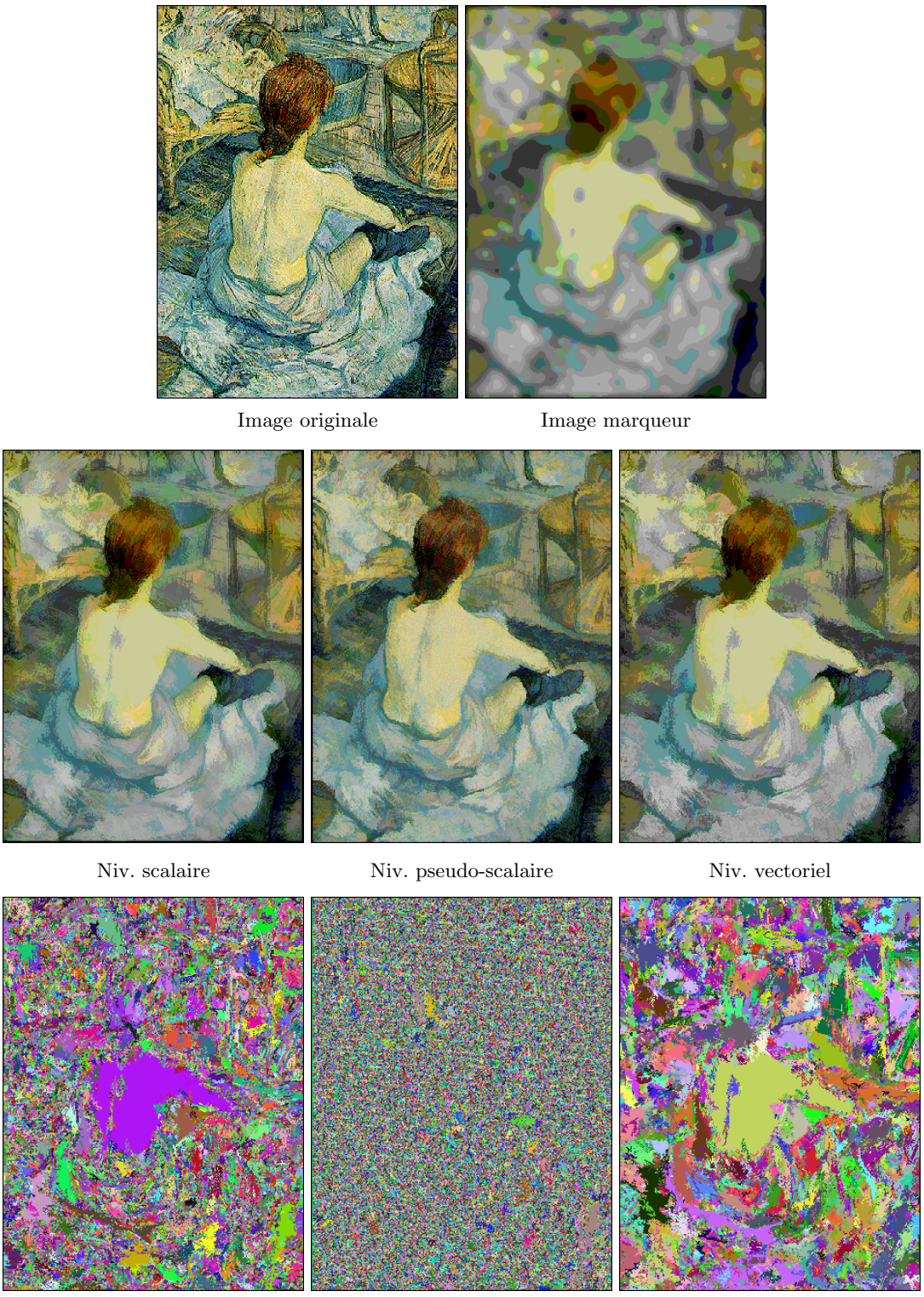


FIG. 3.31 – Comparaison des différents algorithmes de nivellement lorsqu'ils s'appliquent au filtrage des images couleur. Image d'exemple : La toilette. Toulouse-Lautrec, 1896. Musée d'Orsay ©, Paris.

Suite à cette analyse nous pouvons conclure que :

Les nivellements pseudo-scalaires sont rapides et très compétitifs dans le cadre des applications visuelles grâce à leur traitement pertinent de la couleur. Pour tout cela, ils deviennent des outils très puissants pour simplifier des images en vue du codage.

Les nivellements vectoriels autarciques réussissent les niveaux de simplification les plus remarquables, tout en gardant de forts contrastes parmi les objets présents dans l'image filtrée. Ces bonnes propriétés font d'eux des outils idéaux pour le filtrage des images couleur en vue de la segmentation, lors des applications sans contrainte de vitesse.

Les nivellements scalaires calculés composante par composante, offrent un bon compromis par rapport aux deux autres techniques. Bien qu'ils ne soient pas les plus rapides, ils peuvent être facilement parallélisés. Bien que le niveau de simplification achevé ne soit pas le plus fort, en faisant usage des zones quasi-plates, il est également remarquable. De même, la perception de la couleur reste acceptable conditionnée à l'usage d'une image marqueur appropriée. Pour toutes ces raisons, celle-ci sera l'option choisie pour des applications de segmentation en temps réel.

3.5 Conclusions

Nous avons commencé ce chapitre en introduisant la notion de *zone plate*, ainsi qu'une nouvelle classe de filtres connexes nommés *nivellements*. Par leurs bonnes propriétés ces opérateurs se sont révélés d'être des filtres puissants ayant comme principal atout leur capacité à lisser l'image tout en préservant la position et la clarté des contours des objets. En outre, une étude plus détaillée a montré qu'ils sont aussi puissants que versatiles car ils peuvent être appliqués au calcul de la distance parmi des images, à la détection du mouvement, au filtrage multi-échelle, ou même au filtrage sélectif dans des applications de suivi. Nous avons vu également que les nivellements scalaires peuvent être facilement étendus aux espaces vectoriels lorsqu'ils s'appliquent composante par composante. Cependant il existe deux inconvénients à cette approche : a) le résultat dépend de l'espace des couleurs utilisé ; b) des couleurs parasites peuvent parfois apparaître.

Conscients de ce fait, nous avons proposé deux nouveaux algorithmes faisant l'extension aux espaces vectoriels. Le premier, nommé pseudo-scalaire produit des couleurs visuellement agréables, idéales pour des applications de codage. Néanmoins, la contrainte liée à la perception de la couleur limite notamment l'élargissement des zones plates. Les nivellements vectoriels autarciques surmontent ce inconvénient. Leur calcul est fait par simple déplacement des vecteurs de couleur présents sur l'image originale. Même si la propagation de la couleur peut entraîner des effets visuels désagréables pour le spectateur, ils présentent des performances très attractives pour des applications basées sur la segmentation. Dans le cadre d'une poursuite de ces travaux, de nouvelles extensions vectorielles sur la base des zones quasi-plates pourraient être envisagées.

Chapitre 4

Segmentation de l'Image : de la zone plate à la région

Ce chapitre est consacré à la segmentation, car cet outil est à la base de tout système d'analyse d'images au niveau du contenu. Les techniques de segmentation ont pour but de produire une partition de l'image aussi proche que possible de celle faite par l'œil humain.

Cependant, la segmentation peut devenir une tâche extrêmement difficile lorsque les images à traiter sont bruitées, même dégradées pour la compression d'un système de codage, ou lorsqu'on n'a aucune connaissance à priori sur le contenu de la scène.

Dans la ligne de nos travaux, nous reprendrons un algorithme morphologique connu pour ses bonnes propriétés, à savoir la Ligne de Partage des Eaux. Plus concrètement, nous proposerons une extension du processus d'inondation classique face à des applications pour lesquelles on dispose de marqueurs pour les zones d'intérêt. Cette approche repose sur un ensemble de lois d'absorption contrôlant la pertinence des fusions des lacs.

Finalement, nous verrons comment il est possible de créer des hiérarchies de partitions qui apportent des solutions aux applications n'ayant aucune information concernant le contenu des images à segmenter.

4.1 Introduction

Dans le domaine de la segmentation d'images, un long chemin a été parcouru depuis que les premières techniques virent le jour [14, 103, 45, 110, 107, 11]. Cependant, tous ces travaux ne se groupent pas autour d'une théorie générale. Ils forment un ensemble de techniques hétérogènes, ayant chacune leurs atouts et leurs inconvénients. Il est bien au-delà de nos possibilités de faire ici un état de l'art détaillé sur ce sujet. Nous nous limiterons à reporter le lecteur vers deux études générales : celle de Haralik et Shapiro dans [41], comparant principalement des techniques de seuillage, de classification, de croissance de régions et de fusion de

régions ; et celle de Pal et Pal [71] focalisée sur les approches probabilistes : des techniques de relaxation, de logique floue, de classification par des modèles de Markov, et de classification par des réseaux neuronaux.

Dans la ligne de nos travaux, nous avons repris les algorithmes de segmentation morphologiques décrits originalement par Beucher et Lantuéjoul dans [11]. Par leur versatilité et leur robustesse, ces algorithmes nous offrent un vaste support à partir duquel il est possible d'envisager de nouvelles extensions dans le cadre des applications multimédia.

4.1.1 Définitions de base

Avant de rentrer dans la description des algorithmes proprement dits, nous voudrions préciser les définitions de *partition* et *hiérarchie de partitions*, celles-ci étant utilisées à plusieurs reprises tout au long de ce chapitre. Dans un sens large, une partition peut être formellement définie comme suit,

Définition 13 (Partition) *Une partition $\mathcal{P}$ de l'espace E est un ensemble de composantes connexes $\{R_i\}$ disjointes, dont l'union est l'espace complet :*

$$\begin{array}{lll} \forall\, i,j \;\; i \neq j \quad R_i \cap R_j & = & \emptyset \\ \cup R_i & = & E \end{array}$$

où chacune des composantes R_i est une classe de la partition $\mathcal{P}$ que nous nommerons région.

Ainsi, la partition associée à une image numérique n'est qu'une autre image, de la même taille que l'originale, établissant des contours qui délimitent un ensemble de régions. Notons également que cette partition de l'espace n'est pas unique.

En partant de cette définition, nous dirons qu'une partition $\mathcal{P}^0$ est incluse dans une partition $\mathcal{P}^1$ si toute région R_j^1 est complètement incluse dans une unique région R_i^0. Ceci nous permet de définir les *hiérarchies de partitions* comme,

Définition 14 (Hiérarchie de partitions) *Soit $\mathcal{H}$ un ensemble de partitions $\{\mathcal{P}^i\}$. Nous dirons que $\mathcal{H}$ forme une hiérarchie de partitions s'il est possible d'établir un ordre d'inclusion parmi toute paire d'éléments de $\mathcal{H}$,*

$$\forall\, i,j \quad \mathcal{P}^i \cap \mathcal{P}^j \in \{\mathcal{P}^i, \mathcal{P}^j\}$$

En guise d'exemple, la Figure 4.1 montre une hiérarchie $\mathcal{H}$ composée de trois partitions $\{\mathcal{P}^0, \mathcal{P}^1, \mathcal{P}^2\}$. Chacune des partitions propose un découpage complet de l'image et il est possible d'établir un ordre complet d'inclusion parmi elles. Par exemple, nous dirons que $\mathcal{P}^2$ est une partition plus fine que $\mathcal{P}^1$, car $\mathcal{P}^2$ inclut tous les contours de $\mathcal{P}^1$, ainsi que d'autres.

Dans cet exemple nous avons différencié chacune des régions de la partition en introduisant un label du type R_i. Cependant, lorsqu'il s'agira de la partition d'une image réelle, nous

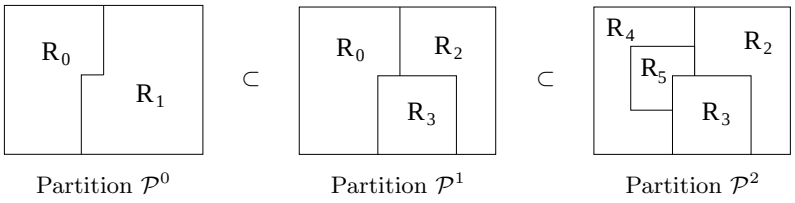


FIG. 4.1 – Exemple d'une hiérarchie de partitions.

remplacerons les labels par la valeur moyenne des pixels à l'intérieur des régions de l'image originale, afin de mieux percevoir la correspondance des régions avec les objets.

4.2 La Segmentation Morphologique

La segmentation morphologique est basée sur un algorithme déjà classique, *la Ligne de Partage des Eaux* (LPE), introduite par Beucher et Lantuéjoul dans [11]. Il existe de nombreux ouvrages de référence à ce sujet, parmi lesquels la thèse de Beucher [10], et les travaux de Meyer [67, 61] et Vincent [101] pour une implémentation plus efficace.

Nous avons organisé cette section en quatre points, en suivant les différentes étapes de la segmentation : premièrement nous aborderons le calcul de l'image gradient, car celle-ci offre un meilleur support pour placer les lignes de crête de la LPE ; ensuite nous détaillerons le principe de fonctionnement de la LPE, en présentant deux extensions : celle qui inonde à partir de marqueurs pour éviter la sursegmentation, et une version améliorée qui inclut des contraintes sur le processus d'inondation pour éviter des fusions non désirées.

4.2.1 Calcul de l'image gradient

Dans les chapitres précédents, nous avons introduit la notion de zone plate ainsi que celle des filtres connexes, dont le but était de rendre l'image aussi lisse que possible. A présent, en partant de l'image simplifiée, nous allons nous intéresser à la détection des contours. Pour cela, commençons en regardant de plus près la distribution des zones plates par rapport au contenu de l'image. On remarque aisément que :

- les zones plates les plus larges, étant des régions homogènes à niveau de gris, se placent normalement à l'intérieur des objets. Au contraire,

- les zones plates les plus petites entourent les fortes variations du signal.

Ces dernières, nous les nommerons *zones de transition*, car c'est sur elles que se trouvent les frontières parmi les régions.

Dans le but de rehausser les zones de transition par rapport aux zones plates, nous allons utiliser le *gradient morphologique* décrit originalement par Beucher dans [10]. Cet opérateur, s'appliquant à l'image à niveau de gris, rend comme résultat une image où les zones plates

deviennent des minima et les contours des maxima. Plus une valeur sur l'image gradient est haute, plus la transition de l'image originale à ce point-là est contrastée. Pour calculer le gradient morphologique d'une image, il suffit d'assigner à chacun des pixels la plus grande différence entre les valeurs de son voisinage.

En termes morphologiques, l'image gradient est la différence entre l'image dilatée et l'image érodée :

$$\Delta(f) = \delta(f) - \varepsilon(f) \tag{4.1}$$

L'image érodée étant en dessous de l'image dilatée, le gradient morphologique est toujours positif. Il ne reflète pas le signe des transitions, d'autant que cette information n'est pas requise pour les algorithmes de segmentation morphologiques. Pour illustrer la façon dont cet opérateur agit, nous avons présenté trois exemples sur une dimension :

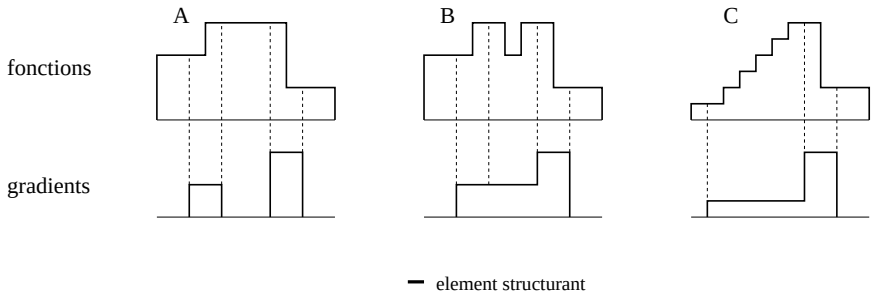


FIG. 4.2 – Gradient morphologique calculé sur trois signaux en une dimension.

Nous avons commencé avec un exemple très simple A, pour montrer comment les pics du gradient suivent parfaitement avec les transitions du signal. La hauteur de chacun des pics reflète la valeur du contraste. De façon plus générale, on peut affirmer que le gradient morphologique est non-nul sur toute transition de l'image de départ.

Néanmoins, à cause de la taille des éléments structurants, il n'a pas la précision suffisante pour détecter les transitions entre des petites zones plates comme celle de B. Finalement dans C, nous avons étudié le comportement du gradient sur les zones de légère pente. Celles-ci nous intéressent spécialement car elles sont produites en nombre par les lambda nivellements.

Ainsi, on se rend compte que le gradient classique traite ces zones comme étant des transitions, et non pas comme des zones quasi-plates dans le sens de nos définitions.

Afin d'assurer la cohérence dans la chaîne du traitement, nous allons calculer le gradient morphologique en reprenant les définitions de *lambda érosion* et *lambda dilatation* présentées dans le chapitre précédent (équations 3.8 et 3.9). Dans tout ce qui suit, nous calculerons le *lambda gradient* comme,

$$\Delta_\lambda(f) = \delta_\lambda(f) - \varepsilon_\lambda(f) \tag{4.2}$$

Nous verrons dans les paragraphes qui suivent que le processus de segmentation, agissant exclusivement sur le gradient, sera identique pour les images couleur ainsi que pour les images monochromes. La seule différence, au niveau des résultats obtenus, va résider dans la façon de générer le *gradient couleur*. Or, à cause de la diversité d'espaces de couleur, il n'existe pas une définition unique.

Pour ce qui concerne l'espace YUV, la composante de luminance Y est la plus discriminante vis-à-vis de la segmentation. Néanmoins, dans certains cas, les composantes de chrominance seront les seules à séparer deux régions de teintes différentes. Par conséquent, nous procéderons au calcul du gradient sur des images couleur par recombinaison des gradients calculés sur chacune des composantes,

$$\Delta_\lambda(YUV) = w_y\,\Delta_\lambda(Y) + w_u\,\Delta_\lambda(U) + w_v\,\Delta_\lambda(V) \tag{4.3}$$

Car l'espace YUV n'est pas uniforme, un vecteur de poids (w_y, w_u, w_v) est nécessaire pour tenir compte des différents ordre de valeurs des composantes.

En prenant cette dernière définition, la Figure 4.3 montre le comportement du gradient lorsqu'il est calculé sur des images réelles.

Image originale | Zones quasi-plates | Image gradient

Image nivelée | Zones quasi-plates | Image gradient

FIG. 4.3 – Détection des zones de transition par calcul du gradient morphologique.

Sur ces exemples il est facile d'observer comment les zones de transition ont été fortement rehaussées par rapport aux zones plates. Cependant, et contrariant une première impression, les contours ne peuvent pas être détectés par simple seuillage puisque,

– même après avoir simplifié fortement l'image, sur le gradient restent de nombreuses transitions liées aux textures,

– sur les régions peu contrastées on obtient des contours non-fermés et imprécis.

Le calcul du gradient étant insuffisant pour obtenir une partition de l'image, nous allons rentrer dans le domaine des algorithmes de segmentation. Ces techniques cherchent à diviser l'image en régions par un certain critère d'homogénéité.

4.2.2 La Ligne de Partage des Eaux

La ligne de partage des eaux permet d'extraire une partition de l'image en prenant comme point de départ l'image gradient. Une présentation visuelle et très intuitive de cette méthode se fait par analogie avec le processus d'inondation d'un relief topographique comme celui de la Figure 4.4. L'altitude du relief suit, point à point, les variations de l'image gradient : au fond de chaque vallée nous retrouvons un minimum régional (zones plates de l'image à niveau de gris), tandis qu'au sommet des cols se placent les maxima (zones de fort contraste sur l'image à niveau de gris). Par la suite, nous allons supposer que le processus de segmentation est équivalent au processus d'inondation de cette surface.

Pour déclencher l'inondation, nous allons percer un trou en chacun des minima et commencer à plonger lentement ce relief dans l'eau. Dès que les minima les plus profonds touchent la surface de l'eau, celle-ci commence à couler à travers de leur trous. Au fur et à mesure que le niveau des eaux monte, de plus en plus de minima deviennent des sources. Chacune de ces sources donne lieu à un lac. Pendant tout le processus d'inondation le niveau de l'eau monte à vitesse constante jusqu'à ce qu'il atteigne l'altitude maximale du relief. A ce point-là, toute la surface est couverte par les eaux.

Pour connaître la zone d'influence de chacune des sources, il suffit de construire des barrages sur les lignes de crête chaque fois que les eaux provenant de bassins versants différents risquent de se mélanger. De cette façon, chaque bassin versant (minima du gradient) donnera naissance à l'une des région dans la partition résultante.

A la fin du processus, l'ensemble des digues, nommées lignes de partage des eaux (LPE), forment un réseau de contours fermés divisant la totalité de l'image.

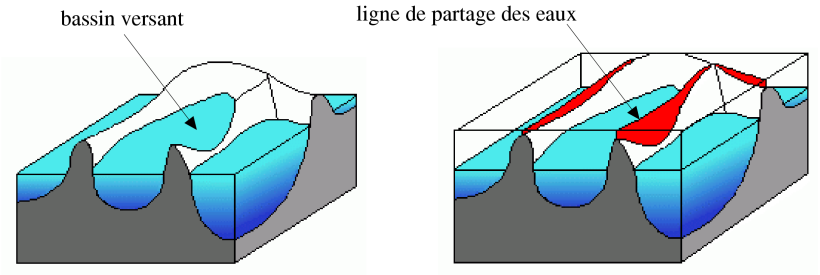


FIG. 4.4 – La Ligne de Partage des Eaux.

Nous verrons par la suite qu'il y a plusieurs façons de procéder à l'inondation du relief qui aboutissent au même résultat. Celle que nous avons utilisée pour décrire l'algorithme, connue comme *inondation uniforme*, est la plus intuitive. Cependant, c'est *l'inondation synchrone* qui

permettra de formuler plus facilement les extensions que nous avons dérivées de l'algorithme classique.

Pour les comparer, nous avons reproduit les deux processus dans la Figure 4.5. Afin de mieux distinguer l'ensemble des régions de la partition, l'eau de chacun des bassins versants a été colorée différemment. Notons que les deux méthodes partent de la même image gradient et arrivent à la même partition. Par contre, il est simple de différencier les deux processus en tenant compte des traits suivants :

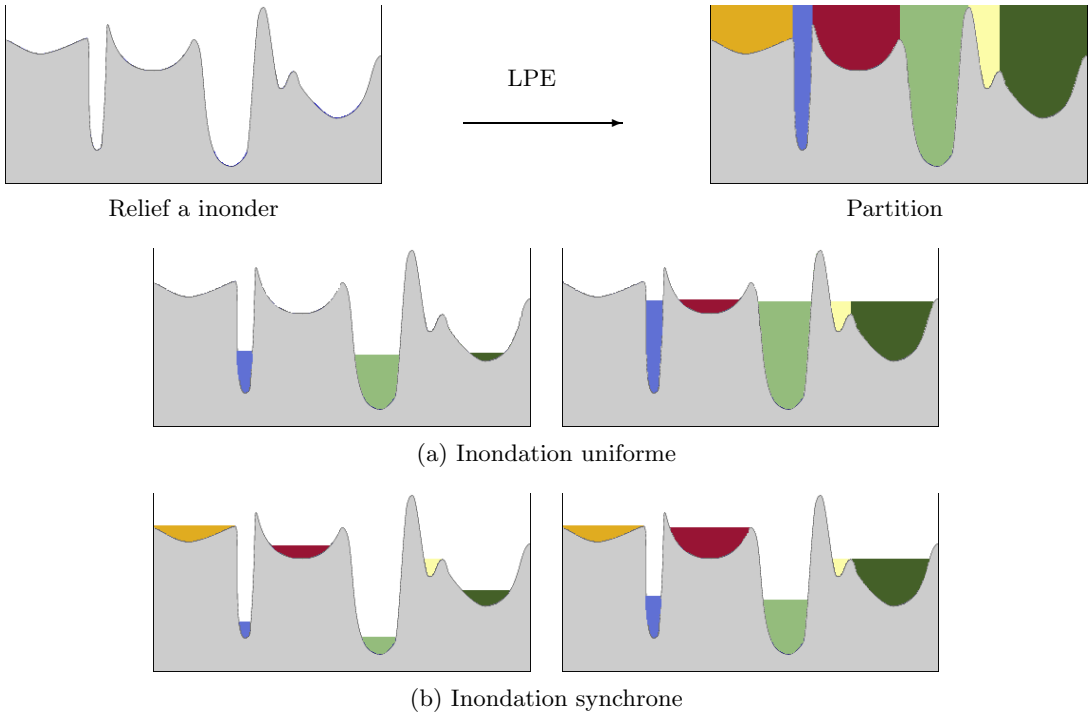
Inondation uniforme : en suivant cet algorithme l'inondation se produit à vitesse constante. Au fur et à mesure que le niveau des eaux augmente, de plus en plus de minima deviennent des sources. Il est simple de démontrer que, à tout moment, l'inondation à la hauteur H peut se calculer en prenant le maximum parmi la fonction du relief et une fonction constante de valeur H.

Inondation synchrone : la différence la plus importante par rapport à l'algorithme précédent se trouve tout au début du processus, car ici, de l'eau commence à couler simultanément au travers de tous les minima. La propagation se fait en gardant uniforme la profondeur dans tous les lacs jusqu'à ce que les moins profonds atteignent une ligne de crête. Pour continuer à grandir, ces derniers lacs doivent attendre jusqu'à ce que de l'eau d'un bassin versant voisin arrive à la même hauteur. A partir de ce moment, les deux lacs continuent à croître séparés par une ligne de partage des eaux.

Par la suite, en passant des simulations au traitement des images réelles, on se rend compte que l'inondation du gradient à partir de tous les minima provoque une sur-segmentation. La partition résultante est trop fine pour permettre, à partir d'une seule des régions, de reconnaître un objet quelconque présent dans la scène.

Même en simplifiant l'image, sur les zones de transition se groupent de nombreuses petites régions n'ayant aucun intérêt au niveau visuel. La Figure 4.6 illustre un exemple de ce fait en comparant la segmentation d'une image (a) avec une version filtrée d'elle même (b) : sur la gauche on retrouve les images gradient ; au milieu, les minima à partir desquels commence le processus d'inondation ; et à droite, les partitions résultantes. Dans cet exemple nous avons comparé deux approches différentes concernant l'obtention de l'image gradient :

Gradient calculé sur l'image originale : bien que celle-ci soit la procédure la plus utilisée, on remarque sur les partitions résultantes (a1 et b1) que les plus larges zones plates ont été fragmentées. Ce fait, étonnant au premier coup d'œil, peut être expliqué en tenant compte de la définition que nous avons donnée pour les zones de couleur quasi-plate, et la façon dont le gradient couleur est construit. Ainsi, une zone de couleur quasi-plate apparaît lors de l'existence d'un chemin de pente inférieure ou égale à λ sur chacune des composantes, même si ces chemins ne coïncident pas forcément. Et c'est précisément cette dernière remarque qui explique le comportement du gradient



(a) Inondation uniforme

(b) Inondation synchrone

FIG. 4.5 – Différents processus de calcul de la Ligne de Partage des Eaux.

couleur. Or, l'assemblage des gradients calculés composante par composante, peut créer des contours et des minima à l'intérieur des zones plates.

Gradient calculé sur l'image des zones plates : une façon de préserver l'unicité des zones plates consiste à appliquer le gradient directement sur la partition des zones plates, au lieu de le calculer sur l'image originale. Ces deux démarches sont identiques lorsqu'on considère des zones strictement plates sur des images à niveau de gris. Par contre, le résultat diffère sur les images couleur. Pour ces derniers, nous allons remplacer le label de chacune des zones plates par la couleur moyenne de l'image originale à l'intérieur de la région. Ensuite le gradient est calculé sur cette nouvelle image. On observe dans (a2,b2) que les plus petites zones plates se trouvant sur les zones de transition ne donnent pas lieu à des minima. Par conséquent, la segmentation résultante aura moins de régions, et il existera une correspondance parmi les plus larges zones plates et les régions de la partition.

En appliquant cette dernière stratégie sur les images d'exemple, on observe une différence remarquable au niveau de la densité et de la taille des régions. Cependant, pour que chaque objet puisse être associé à une seule région, il est nécessaire d'empêcher à certains des minima de se propager. Dans les sections suivantes nous verrons que ceci est possible lorsqu'on impose des *marqueurs* au processus d'inondation.

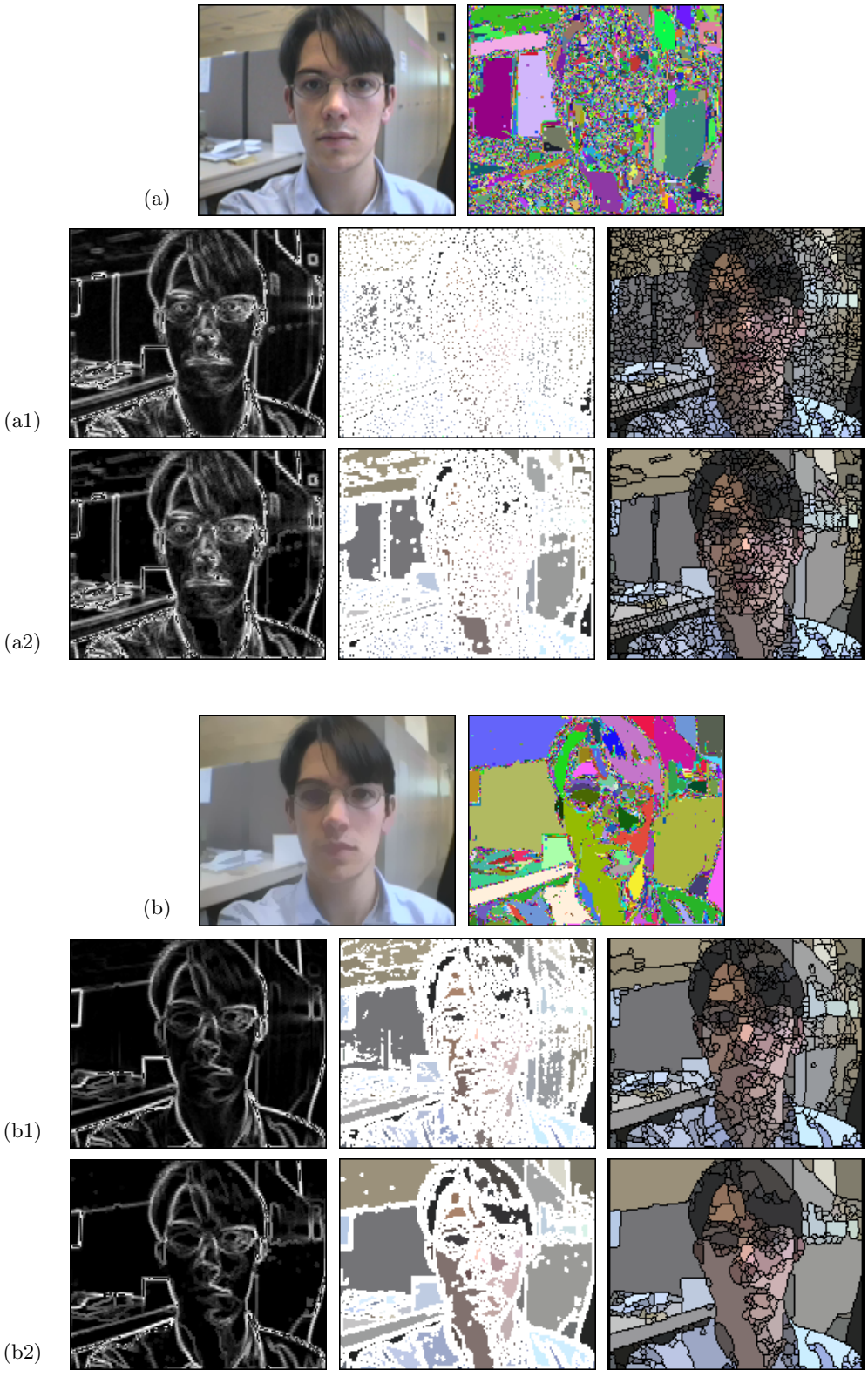


FIG. 4.6 – LPE calculée sur les minima du gradient. (a) Image originale. (b) Image nivelée. (a1,b1) Gradient calculé sur les composantes de l'image couleur. (a2,b2) Gradient calculé sur la couleur moyenne de l'image des zones plates.

4.2.2.1 Inondation à partir de marqueurs

Une des solutions possibles au problème de la sur-segmentation consiste à inonder l'image à partir de marqueurs, idée proposée par Meyer au 81. De façon intuitive, il s'agit ici de ne pas percer un trou sur chacun des minima du relief à inonder, mais uniquement à certains endroits de notre choix. Ainsi, les marqueurs se constituent comme les seules sources inondant le relief (voir Figure 4.7). Les bassins versants n'ayant pas de source seront inondés par les eaux d'un lac voisin lorsque celles-ci dépassent le point col le plus bas. La partition ainsi créée aura autant de régions que de marqueurs.

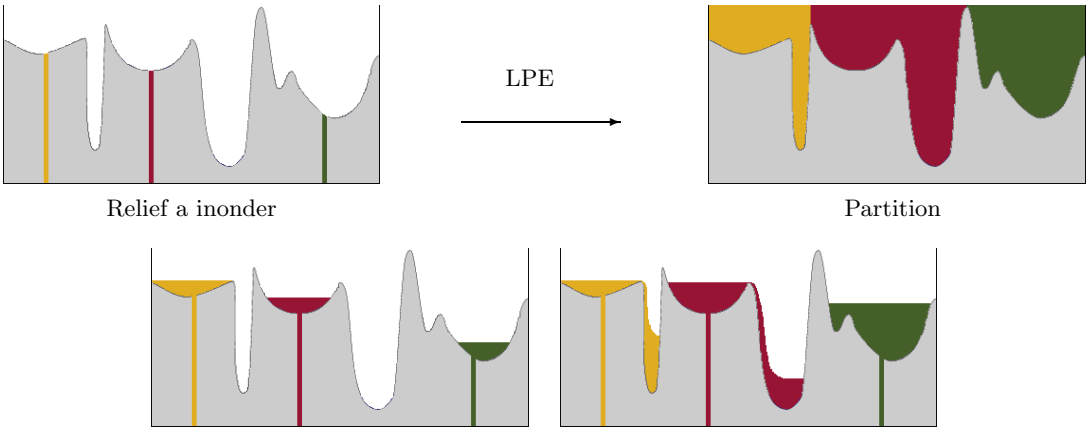


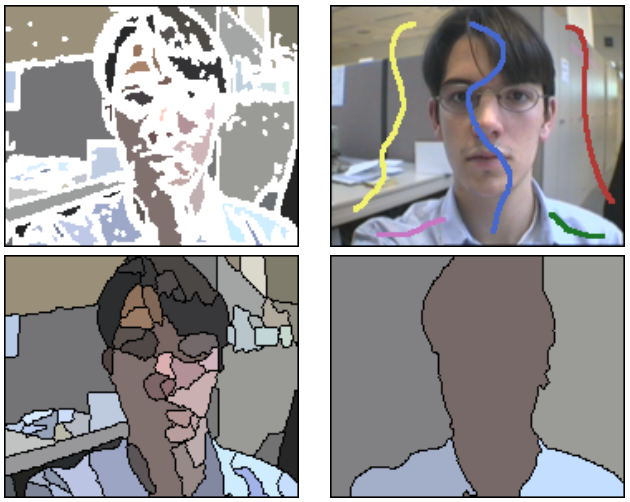
FIG. 4.7 – LPE construite à partir de marqueurs.

Sur des images réelles, la pertinence de cette segmentation sera directement liée au choix des marqueurs. Ceux-ci peuvent être générés manuellement ou de façon automatique en fonction du domaine de l'application :

Extraction manuelle des marqueurs : cette approche est seulement valable lorsqu'on dispose d'un interface pour interagir avec le système, et lorsqu'on traite un nombre réduit d'images sans contrainte de temps. Ces conditions sont largement remplies dans le cadre des applications multimédia permettant de l'interactivité [51]. Or pour elles, c'est l'utilisateur qui, en plaçant un jeux de marqueurs, choisit les régions d'intérêt.

Extraction automatique des marqueurs : à l'extrême opposé se situent les méthodes automatiques, qui à leur tour, se divisent en deux catégories : celles qui, ayant une connaissance à priori de la position des objets d'intérêt, peuvent placer des marqueurs retracés (applications de suivi d'objets [35]) ; et celles qui, n'ayant aucune information sur les objets à segmenter, ne peuvent que garder comme marqueurs les minima de plus grande taille, en considérant que les plus petits ne marquent pas un objet mais seulement une partie.

La Figure 4.8 montre deux exemples des partitions qu'on peut obtenir en construisant la LPE à partir de marqueurs. Dans (a) nous avons mis en œuvre un système d'extraction automatique. Ici, tous les minima du gradient de taille supérieure à 5 pixels ont été gardés comme marqueurs, les autres pas. De cette façon, les lacs les plus petits vont être absorbés par les bassins versants plus larges. On observe que grâce à cette sélection préalable, la partition résultante réduit fortement le nombre de régions par rapport aux résultats obtenus auparavant. Cependant, dans (b), pour avoir une partition encore plus simple, nous avons eu recours à une sélection manuelle des régions d'intérêt. Dans cette mise en œuvre, en tant que utilisateurs, nous avons marqué par un trait chacun des objets à extraire. Ces traits deviennent des marqueurs dans le processus de segmentation, et donnent lieu à l'ensemble des régions de la partition finale.



(a) Sélection automatique (b) Sélection manuelle

FIG. 4.8 – Partitions générées par LPE avec des marqueurs.

Il nous reste encore à préciser les critères à suivre pour établir la qualité d'une segmentation. Malheureusement, car il est difficile de formuler une mesure objective, nous devons nous appuyer sur l'évaluation visuelle des traits suivants :

– *les contours doivent être précis, fermés et simples ;*

– *les régions doivent représenter un seul objet, étant uniformes par rapport aux caractéristiques discriminantes utilisées pour les extraire.*

Bien que la LPE accomplisse toujours avec la première des conditions requises, l'expérience nous a montré que, dans quelques cas, le bon comportement par rapport au deuxième critère reste lié à la sélection et au placement des marqueurs.

Un exemple de ceci est illustré dans la Figure 4.9. Cette image montre quatre objets différents placés sur un fond uniforme. Dans un premier temps, avant le processus de segmen-

tation, on a procédé au calcul du gradient couleur, puis à la sélection des marqueurs. Celle-ci a été faite de façon manuelle à deux reprises, dans le but de segmenter à chaque fois un objet différent (premièrement la bouteille, ensuite le poivron rouge). Pour cela nous avons suivi un critère très intuitif, nous menant à placer un marqueur à l'intérieur de l'objet choisi ; plus un marqueur dans le fond[1], celui-ci étant supposé d'englober toutes les autres régions.

Finalement, pour chacun des jeux de marqueurs, nous avons procédé à l'inondation du gradient. Mais, contrairement au résultat attendu, les deux segmentations aboutissent à la même partition, n'isolant pas les objets sélectionnés.

La partition proposée par la LPE dans cet exemple ne peut pas être considérée comme erronée, car elle est la conséquence logique de la façon dont la propagation des marqueurs est faite. Or, le processus d'inondation s'abstrait complètement des propriétés intrinsèques des lacs lors des fusions. Ainsi, la ligne de partage des eaux se place toujours sur la plus haute ligne de crête séparant deux bassins versants.

Pour des images au niveau de gris, ceci équivaut à situer la frontière parmi deux marqueurs sur la transition la plus contrastée les séparant, même si cela entraîne la fusion d'objets de couleur, texture ou mouvement différents.

Dans le cadre de certaines applications, ce comportement peut être critique, car des fusions, parfois très choquantes, apparaissent dès que le placement des marqueurs n'est pas suffisamment précis.

Ce fait a été mis en évidence lors de l'application des algorithmes à la segmentation de séquences de vidéophonie. Dans la dernière partie de ce mémoire, il deviendra d'avantage évident que la mise en œuvre des outils pour traiter des problèmes réels porte les techniques existantes à leur limites.

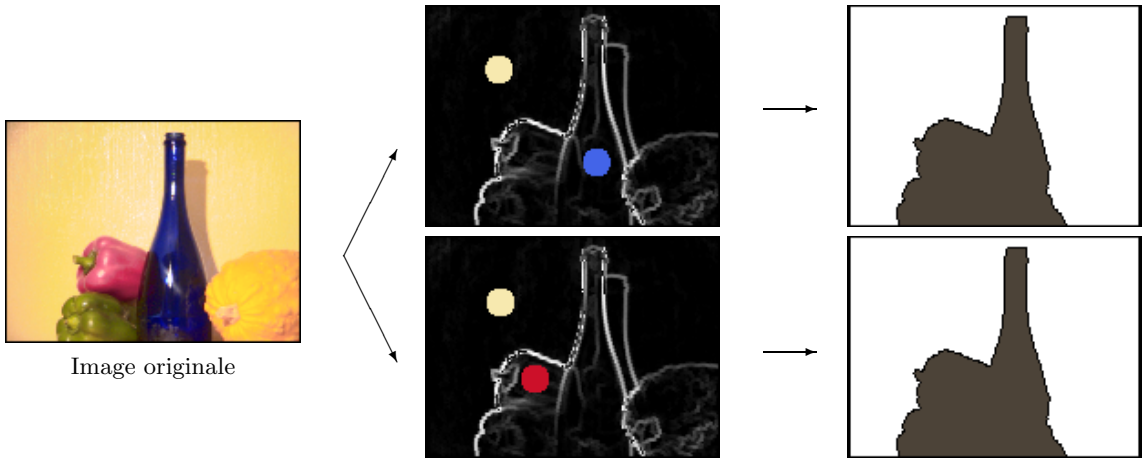


FIG. 4.9 – LPE classique calculée à partir de marqueurs.

[1]Cette notion de *marqueur du fond*, va être très utilisée lors des applications de la segmentation lorsque l'on ne s'intéresse pas à la partition globale de l'image mais seulement à quelques-unes de ses régions.

4.2.2.2 Inondation à partir de marqueurs contraints

A l'origine, la Ligne de Partage des Eaux peut être considérée comme un *algorithme de croissance de régions*, car elle propage pixel à pixel un ensemble de marqueurs jusqu'à ce que la totalité de l'image soit partitionnée.

A l'opposé de ces techniques se trouvent les *algorithmes de fusion de régions*. Ces méthodes, en partant déjà d'une partition très fine de l'image, procèdent par itération à la fusion des régions les plus proches par un certain critère d'homogénéité. Une telle façon de propager l'information permet,

- d'établir un contrôle strict sur la pertinence des fusions, et
- d'inclure un critère d'arrêt pour les fusions.

Ces deux propriétés, manquantes dans la LPE classique, présentent une solution possible aux cas pour lesquels, à cause de fusions inappropriées, les résultats sont peut satisfaisants. Une première approche des outils de segmentation morphologiques vers les techniques de fusion de régions a été faite par Marcotegui dans sa thèse [50], concernant la segmentation de séquences en vue du codage. Par ailleurs, ces travaux n'ont jamais été étendus à la LPE car, dans le cadre de cette application, la précision du gradient, vis-à-vis de la détection de petits objets, n'était pas suffisante.

L'extension que nous allons proposer à la LPE classique peut être formulée à partir d'un ensemble de lois d'absorption concernant la fusion des lacs. Celles-ci peuvent se résumer comme suit,

Lois d'absorption : dès que deux lacs, A et B, se rencontrent pendant le processus d'inondation, leur fusion sera contrôlée par une des quatre lois suivantes,

A $\not\curvearrowright$ B	:	A ne sera jamais absorbé par B B ne sera jamais absorbé par A
A $\curvearrowright$ B	:	A sera toujours absorbé par B
A $\overset{?}{\curvearrowright}$ B	:	A peut être absorbé par B si et seulement si, une condition d'absorption est vérifiée
A $\overset{?}{\not\curvearrowright}$ B	:	A sera absorbé par B ou B sera absorbé par A dans la mesure où la condition d'absorption est vérifiée

Nous verrons qu'au moyen de cet ensemble de lois il est possible de formuler de façon synthétique le processus de la LPE classique, avec ou sans marqueurs. En outre, ayant une formulation plus généraliste, elles permettent aussi de créer des extensions qui rapprochent la LPE des algorithmes de fusion de régions.

Inondation synchrone : dans cette approche, le relief commence à être inondé à partir de tous les minima du gradient. Chacun des minima est considéré comme un marqueur. Ici une seule loi d'absorption s'applique pour empêcher toute fusion des lacs,

$$L_i \not\curvearrowright L_j \tag{4.4}$$

Inondation synchrone avec des marqueurs : la différence par rapport au processus précédent, c'est qu'ici, au fur et à mesure que le niveau des eaux monte, se constituent deux types de lacs : ceux qui contiennent un marqueur, et ceux qui n'en contiennent aucun. Pour les différencier, sur l'exemple de la Figure 4.10 nous avons laissé ces derniers lacs sans couleur. Le signe d'interrogation apparaissant à l'intérieur fait référence à l'incertitude de leur label. Or, sur la partition finale seulement les lacs colorés feront surface. Ainsi, lorsque se retrouvent

1) Un lac contenant un marqueur (L^{mrq}) avec un lac non-marqué (L), ce dernier est absorbé et prend la couleur du marqueur correspondant,

$$L_i \curvearrowright L_j^{mrq} \tag{4.5}$$

2) Deux lacs ayant le même marqueur (L^{mrq^m}), ou deux lacs sans marqueur (L) fusionnent toujours et continuent à croître ensemble en formant un plus large lac. Pour établir un ordre des fusions, un certain critère peut être évaluée,

$$\begin{gathered} L_i^{mrq^m} \overset{?}{\curvearrowright} L_j^{mrq^m} \\ L_i \overset{?}{\curvearrowright} L_j \end{gathered} \tag{4.6}$$

3) Deux lacs contenant de marqueurs différents, L^{mrq^m} et L^{mrq^n}, une ligne de partage des eaux est construite pour les séparer sur la partition finale,

$$L_i^{mrq^m} \not\curvearrowright L_j^{mrq^n} \tag{4.7}$$

Inondation synchrone avec des marqueurs contraints : cette procédure suit la stratégie précédente sauf pour le premier des critères d'absorption, qui, ici devient conditionnelle. Ainsi,

1') Si un lac contenant un marqueur L^{mrq} retrouve un lac non marqué L, ce dernier sera absorbé si et seulement si une certaine condition est vérifiée,

$$L_i \overset{?}{\curvearrowright} L_j^{mrq} \tag{4.8}$$

Cette nouvelle procédure d'inondation, étant une généralisation de toutes les précédentes, mérite déjà quelques commentaires.

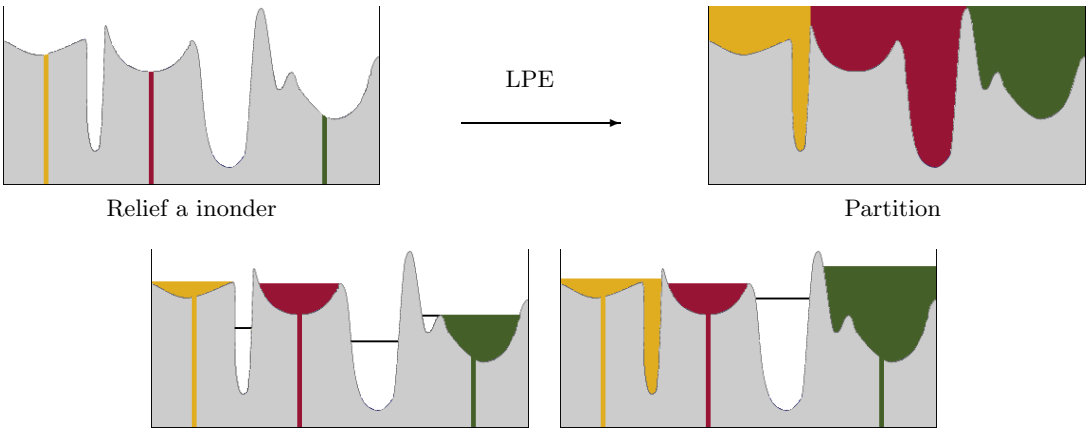


FIG. 4.10 – Processus d'inondation synchrone avec marqueurs.

Il faut souligner tout d'abord que de nouvelles régions peuvent apparaître sur la partition finale lors de l'inclusion d'un critère conditionnel d'absorption. En effet, il est possible que certaines régions non-marquées, entourées par des régions ayant des contraintes d'absorption, ne fusionnent jamais. La Figure 4.11 concrétise ce fait en reprenant l'exemple antérieur en une dimension. Ainsi, en fonction des conditions appliquées, l'inondation d'un même relief à partir d'un même jeu de marqueurs peut aboutir à des partitions différentes dans les distributions des contours mais aussi dans le nombre des régions. Ceci peut être un avantage ou un inconvénient en fonction de l'application.

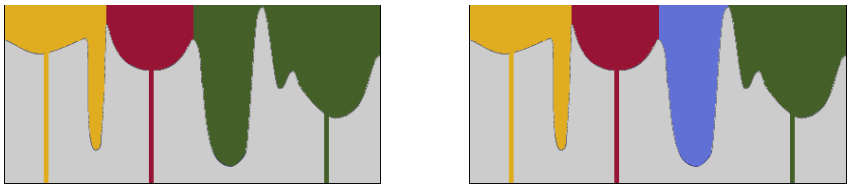


FIG. 4.11 – Possibles résultats obtenus par inondation synchrone avec des marqueurs contraints.

Mais, sûrement, le trait le plus marquant de cette nouvelle formulation est la versatilité qu'elle apporte au processus de segmentation par LPE. Car pour chaque type de marqueur, on peut choisir s'il est pertinent d'imposer une condition d'absorption et, si ceci est le cas, on peut choisir aussi les caractéristiques associées à ce type de marqueur (modèle de couleur, de texture, de mouvement, ...), et quels seront les critères d'absorption à évaluer lors d'une rencontre avec d'autres lacs.

Il ne nous reste qu'à faire quelques remarques sur la façon dont les caractéristiques des marqueurs peuvent être obtenues. Dans ce sens, on peut procéder de deux façons différentes :

– fournir les caractéristiques directement au système au moyen d'un modèle.

– calculer les caractéristiques à partir des données disponibles à l'intérieur du lac incluant le marqueur.

Dans le premier des cas, toute comparaison concernant une fusion sera toujours calculée par rapport aux données du modèle, qui restent inchangées tout au long du processus d'inondation. Au contraire, dans la deuxième approche, ces valeurs peuvent être mises à jour au fur et à mesure que le lac grandit. Cependant, lors de l'initialisation du processus on doit disposer d'un marqueur suffisamment large et bien placé pour pouvoir abstraire de façon robuste les traits de la région marquée.

Néanmoins, cette souplesse à un coût, car la variabilité dans le nombre et le rôle des marqueurs augmente le temps de calcul et rend plus complexe l'implémentation de l'algorithme d'inondation.

Afin d'éclaircir quelques points concernant ce sujet, nous allons détailler par la suite une mise en œuvre possible des algorithmes. L'extension à d'autres domaines d'application pourrait être formulé par adaptation de notre code.

Mise en œuvre dans le cadre d'une application

Nous allons particulariser la formulation générale concernant les inondations contraintes par un ensemble donné de règles de propagation. Nous avons ciblé ici un type d'application très concrète, mais très générale en même temps : le but de notre algorithme est d'extraire de la scène un objet ayant une couleur particulière et uniforme. Avant la mise en œuvre, certaines caractéristiques du système doivent être précisées :

1. Types de marqueurs : dans le cadre de notre application, le système travaillera avec deux types de marqueurs, chacun soumis à des règles d'absorption différentes :

1) *Marqueur de type « fond »*
2) *Marqueur de type « objet »*

Cela n'empêche qu'on puisse avoir en même temps plusieurs marqueurs de type objet, ou des marqueurs formés de plusieurs composantes connexes. Cependant, pour éviter des conflits lorsque deux marqueurs différents se placent à l'intérieur d'un même bassin versant, avant de procéder à l'inondation du gradient nous allons marquer tous les minima de façon à ce que,

– tous ceux qui sont touchés par un seul marqueur prennent le label de celui-ci ;

– s'il existe des minima touchés par plusieurs marqueurs, on ne leur assigne aucun label et on laisse le processus d'inondation décider de leur assignation.

2. Règles d'absorption : deuxièmement, on doit fixer les règles d'absorption qui gouverneront le processus d'inondation. Dans notre cas, le marqueur du fond n'aura aucune restriction concernant l'absorption d'autres régions. Par contre, le marqueur de type objet devra préserver une certaine homogénéité à l'intérieur de la région qu'il crée.

Dans tout ce qui suit, nous dénoterons par L^{fnd} le lac créé à partir d'un minima recouvert par le marqueur du fond, par L^{obj} le lac inondé à partir d'un marqueur d'objet, et finalement par L le lac créé à partir d'un minimum du gradient sans marqueur. Compte tenu de cette notation, les règles peuvent se formuler comme suit :

1) L^{fnd} ↛ L^{obj} : *empêche la disparition des marqueurs*
2) L → L^{fnd} : *permet au fond d'absorber toute sorte de régions*
3) L ⇢ L^{obj} : *préserve l'homogénéité du marqueur associé à l'objet*
4) L ⇢ L, L^{obj} ⇢ L^{obj}, L^{fnd} ⇢ L^{fnd} : *permet aux lacs du même type de fusionner entre eux*

On doit remarquer que :

- la première des règles rend fermes les assignations, de façon à ce que, si un lac sans marqueur est absorbé par un des marqueurs (fond ou objet), il ne soit jamais absorbé par l'autre.

- lors de la rencontre de deux lacs appartenant à la même classe, la fusion se produit toujours. Pourtant, la condition qui apparaît sur l'ensemble de ces règles a pour seul but de discerner quel est le lac absorbant et quel est le lac absorbé.

La Figure 4.12 montre un exemple dans lequel différentes règles d'absorption doivent être appliquées. Sur la gauche on peut observer le relief à inonder, celui-ci étant composé de six bassins versants (a, b, c, d, e, f). Suite au placement des marqueurs, le lac créé à l'intérieur de a fera partie de l'objet, le lac créé à l'intérieur de f fera partie du fond. Les autres lacs devront attendre le déroulement de l'inondation pour connaître leur assignation finale.

Sur l'image de droite, on a condensé trois types différents de fusion : la rencontre d'un lac non-marqué b avec un lac appartenant à l'objet a ; la rencontre de deux lacs non-marqués (c et d) ; et finalement la rencontre d'un lac non-marqué e avec un lac qui appartient au fond f.

Tenant compte de nos règles d'absorption, la première des fusions est conditionnelle. Cela signifie que b devra vérifier un certain critère d'homogénéité avec le marqueur de l'objet a pour pouvoir le rejoindre. Au contraire, c et d vont toujours fusionner, et la condition d'absorption va seulement décider du sens de l'absorption. A son tour, n'ayant pas de contrainte sur le marqueur du fond, e sera absorbé par f.

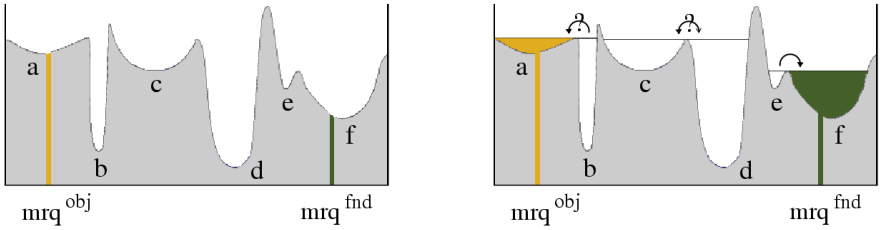


FIG. 4.12 – Exemple d'application des règles d'absorption.

3. Critères d'absorption : finalement il ne nous reste qu'à préciser quels sont les critères à évaluer dans des fusions conditionnelles :

1) *Sens des absorptions internes* ( ) : afin d'établir un ordre d'absorption lors de la rencontre de deux lacs du même type nous allons comparer leur volume, en concluant que le plus grand lac absorbe le plus petit. D'autres critères, basés par exemple sur des mesures de surface ou de profondeur, seraient aussi valables.

2) *Condition d'absorption externe* ( ) : lors de la rencontre d'un lac non-marqué avec un lac appartenant à l'objet, la fusion peut se produire ou pas. Dans ce type de situation, pour permettre au lac non-marqué de rejoindre l'objet, nous allons imposer qu'un certain degré d'homogénéité au niveau de la couleur moyenne des régions soit atteint.

A ce point, ayant défini le type de marqueurs, les règles d'absorption et les conditions à évaluer lors des fusions conditionnelles, nous disposons de toute l'information nécessaire pour mener à terme l'implémentation des algorithmes.

Cependant, étant donné que notre système de décision des fusions est basé sur une estimation de la couleur moyenne des lacs, des erreurs peuvent apparaître à cause de

– la présence de bruit ou de fortes textures sur l'image de départ. En effet, si au moment de questionner la pertinence de la fusion de deux lacs, le calcul de la couleur moyenne donne une valeur aberrante, la fusion ne se produira pas et de nouvelles régions peuvent apparaître.

– le calcul de la moyenne sur un support de la mauvaise taille : soit parce que le lac était trop petit pour permettre d'estimer une valeur de couleur robuste, soit parce que le lac était trop large pour être modélisé avec une valeur moyenne.

Ainsi, lors de la mise au point des algorithmes,

1) pour gagner de la robustesse face au bruit, nous proposons de calculer la couleur moyenne des lacs sur l'image après nivellement. Car sur celle-ci les petites variations du signal ont été fortement atténuées.

2) pour assurer un support optimal pour le calcul de la couleur moyenne nous allons garder l'*historique des absorptions* des bassins versants afin de les traiter séparément ou de façon conjointe, selon notre convenance.

Nous allons expliquer ce dernier point sur l'exemple de la Figure 4.13 que nous connaissons déjà. A ce stade de l'inondation, l'historique des absorptions est composée de trois fléchages indiquant que : b a été absorbé par a, c par d, et e par f. En disposant de ces données, il s'agit à présent de décider de la pertinence de la fusion de $c \cup d$ avec $a \cup b$ lorsque les eaux de b et c se rencontrent sur le point col.

Dans une première approche on peut suggérer de comparer la couleur moyenne de $c \cup d$ avec la couleur moyenne de $a \cup b$. Cependant, $c \cup d$ étant un très grand lac, il se peut que la couleur moyenne ne soit pas suffisamment représentative des valeurs autour du point de rencontre.

Face à ce fait contrariant, on peut penser à restreindre le calcul de la couleur moyenne aux lacs b et c ; car en fin de compte ce sont eux qui sont voisins. Néanmoins, ceci n'est pas non plus une bonne solution, car tout au long du processus d'inondation nous rencontrons de très petits lacs comme b, trop dépendants des gradients de couleur.

La solution que nous avons estimée optimale consiste à prendre comme support de départ les deux lacs voisins. Pour chacun d'eux nous allons considérer :

- si la taille du lac est au-delà d'un certain seuil considéré comme minimal, le lac est accepté comme seul support de calcul.

- si la taille du lac est en dessous de ce seuil, on agrandit le support de calcul en prenant des lacs connectés à celui-ci dans l'historique des fusions jusqu'à ce que le seuil soit dépassé ou jusqu'à ce que le lac arrive à sa taille maximale.

Dans notre exemple précis, ceci implique que nous devrions comparer la couleur de c avec celle de $a \cup b$, b étant considéré comme trop petit pour donner une mesure fiable de couleur.

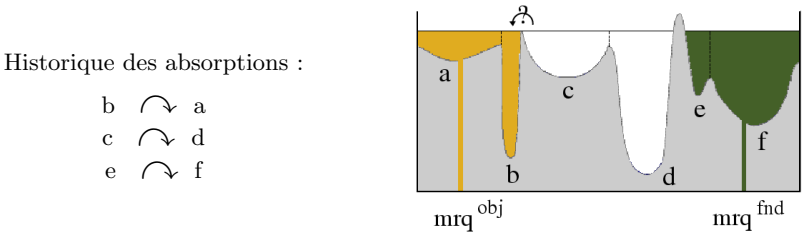


FIG. 4.13 – Exemple des lacs intervenant dans le calcul de la couleur moyenne.

Notons la façon dont la fusion des lacs appartenant à la même classe rend plus robuste la décision lors des fusions conditionnelles. Or, s'il existe un historique de fusions il est possible d'agrandir le support de calcul et d'obtenir une caractérisation de la région plus fiable.

Cette façon de procéder est extensible à tout algorithme de fusion de régions présentant des régions trop larges pour être modelées globalement.

Pour illustrer les atouts de cette nouvelle approche, nous avons repris dans la Figure 4.14 l'image d'exemple pour laquelle la LPE classique n'arrivait pas à extraire les objets marqués. Le gradient et les marqueurs étant les mêmes que ceux utilisés auparavant, nous avons procédé à l'inondation en incluant les contraintes décrites dans cette section.

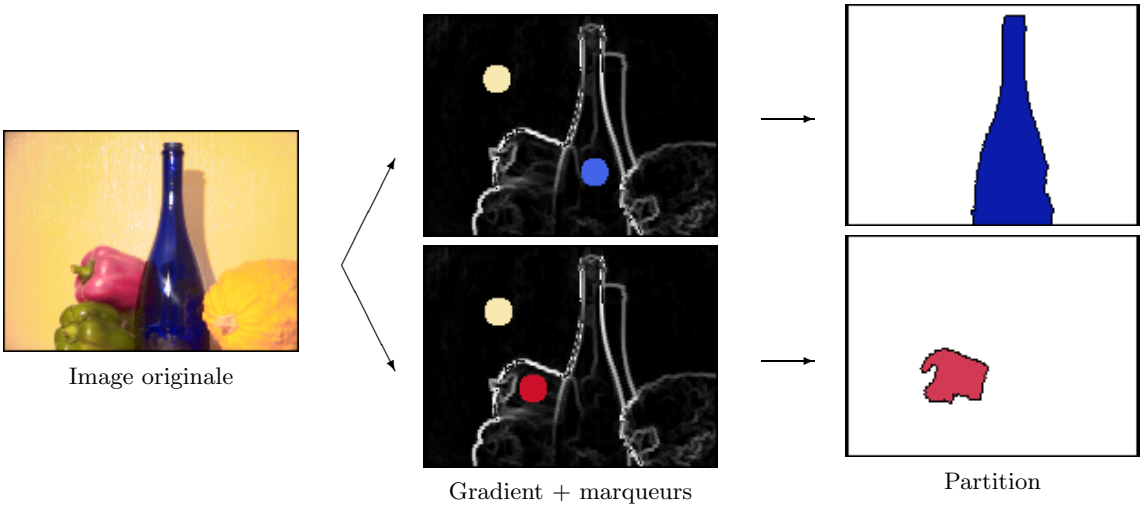


FIG. 4.14 – LPE calculée à partir de marqueurs contraintes.

Les résultats obtenus à présent différent fortement de ceux générés par la LPE classique, car le critère d'homogénéité basé sur la couleur freine l'expansion du marqueur associé à l'objet.

Comme possibles extensions de ce travail, notons que d'autres pondérations pourraient renforcer le critère de décision, dans le but de rendre l'application moins dépendante du placement des marqueurs.

Pour conclure, nous avons résumé dans la page qui suit le flux des opérations concernant l'implémentation d'un algorithme d'inondation à partir de deux marqueurs contraints. Ce même code peut être utilisé comme point de départ d'autres extensions.

Dans la section qui suit nous allons aborder le problème de la segmentation dans un cas plus général, lorsque l'on ne dispose pas d'une information précise sur le contenu de l'image.

◇ IMPLÉMENTATION DE L'INONDATION CONTRAINTE ◇

Initialisation

Pour tout pixel p de l'image :

- Si p fait partie d'un minimum du gradient
  - a = label du bassin versant (fond, objet ou non-marqué)
  - Actualiser la couleur et le volume du lac L_a
  - Pour tout pixel q voisin de p :
    - Si q n'appartenant pas à un minimum et q n'est pas déjà en file d'attente :
      - Garder q en file d'attente avec le label a
      - Marquer q sur l'image comme étant un pixel en file d'attente

Processus d'inondation

Faire :

Extraire un pixel p de la file d'attente

- a = label du bassin versant auquel p appartient
- A = label du plus grand lac englobant le bassin versant a
- Actualiser la couleur et le volume du lac L_a

- Pour tout pixel q voisin de p :
  - Si q n'a encore été assigné à aucun lac et q n'est pas en file d'attente :
    - Garder q en file d'attente avec le label a
    - Marquer q sur l'image comme étant un pixel en file d'attente
  - Si q appartient déjà à un lac
    - b = label du bassin versant auquel q appartient
    - B = label du plus grand lac englobant le bassin versant b
    - Déterminer le type de rencontre parmi les lacs A et B :
      - cas $L^{fnd} \not\curvearrowright L^{obj}$:
        - Il n'y a pas d'absorption, rien n'est fait
      - cas $L \curvearrowright L^{fnd}$:
        - Actualiser la couleur et le volume du lac absorbant
        - Actualiser les labels de tous les bassins versants inclus dans L
      - cas $L \overset{?}{\curvearrowright} L^{obj}$:
        - Critère de décision : étude de la similarité couleur
          - Si la couleur de L est semblable à celle de L^{obj} il y a absorption :
            - Actualiser la couleur et le volume du lac absorbant
            - Actualiser les labels de tous les bassins versants inclus dans L
          - Si la couleur de L est trop différente :
            - Il n'y a pas d'absorption, rien n'est fait
      - cas $L \not\overset{?}{\curvearrowright} L$:
        - Critère de décision : le lac de plus grand volume absorbe le plus petit.
        - Actualiser la couleur et le volume du lac absorbant.

tant que la file d'attente est non vide

4.3 La Segmentation Multiéchelle

La segmentation est une tâche particulièrement difficile lorsqu'il s'agit d'analyser des contenus absolument divers, comme c'est bien souvent le cas pour les applications multimédia. La problématique réside ici dans l'ambiguïté qui entoure le processus de segmentation, car

- *en l'absence totale d'information sur le contenu de l'image, l'inclusion de marqueurs au début du processus d'inondation n'est plus faisable, et*

- *même en disposant de marqueurs, il se peut que certains objets soient de nature trop complexe pour être correctement segmentés en partant d'une hypothèse d'homogénéité.*

Dans ces situations, les procédures d'inondation que nous avons vu jusqu'ici sont mises en défaut ; or, la partition qu'elles créent ne correspond pas avec un découpage naturel de la scène. A la recherche d'une solution plus adaptée, une nouvelle classe de techniques de segmentation est apparue sur la base des *hiérarchies de partitions* présentées au début de ce chapitre. De ce fait, on les appelle couramment *techniques de segmentation hiérarchique* ou *multiéchelle* [66]. Ces algorithmes procèdent en deux étapes : ils construisent dans un premier temps une hiérarchie de partitions, pour ensuite en s'appuyant sur cette représentation de l'information, procéder à la création d'une partition précise de la scène.

4.3.1 Construction de la hiérarchie de partitions

Nous verrons par la suite la façon dont la construction d'une hiérarchie de partitions peut être abordée avec les outils de segmentation morphologiques que nous connaissons déjà.

Dans un premier temps il ne s'agit pas de segmenter l'image dans le but de produire une seule partition, mais tout un ensemble de partitions capables de représenter l'image à différents niveaux de finesse. De plus, ces partitions doivent former une hiérarchie, afin de pouvoir établir un ordre d'inclusion parmi elles.

Par rapport à cet ordre, les régions forment une structure pyramidale : à la base on retrouve la partition la plus fine, contenant une multitude de petites régions, et au sommet la partition la plus grossière qui considère toute l'image comme une seule région. Par ailleurs, nous dirons que la hiérarchie est *complète* si, à chaque marche qu'on monte dans la pyramide, la finesse de la partition diminue d'une région.

Dans la Figure 4.15 nous avons illustré un exemple. De gauche à droite et de haut en bas, nous avons sorti quelques-unes des partitions faisant partie de la hiérarchie associée à une image réelle. Elles représentent l'image à différents niveaux de détail, allant d'une partition grossière à 10 régions, jusqu'à atteindre un maximum de finesse à plus de 1500 régions.

En sachant que la finesse de la partition obtenue par inondation du gradient est proportionnelle au nombre de sources placées initialement sur le relief, on peut envisager la construction d'une hiérarchie de partitions par inondations successives de l'image gradient.

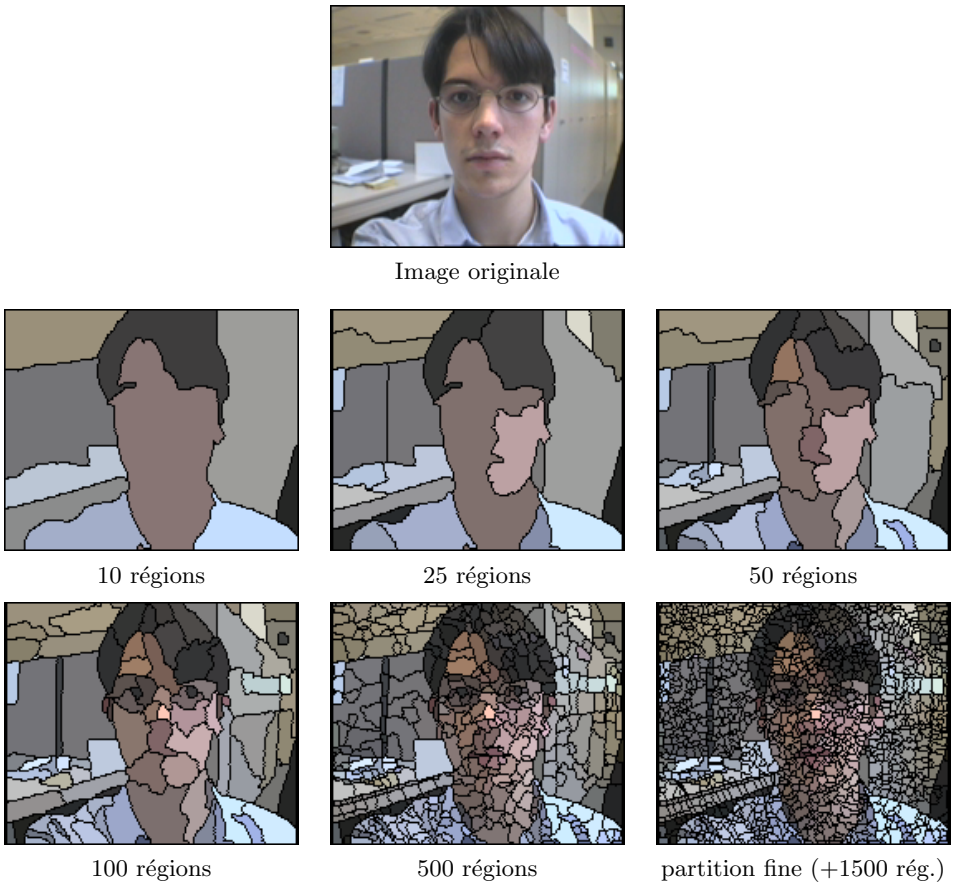


FIG. 4.15 – Exemple de la hiérarchie de partitions associée à une image réelle.

La construction de la base de la pyramide revient à l'inondation du gradient à partir de tous les minima. Ensuite, pour obtenir les autres partitions de la pyramide, il suffit de répéter le processus d'inondation en supprimant à chaque fois la source d'un des minima.

Bien que cela soit la façon la plus intuitive d'approcher la construction d'une hiérarchie de partitions à partir des algorithmes de segmentation morphologiques, cette approche est très coûteuse en temps de calcul. Car, le processus complet entraîne autant d'inondations qu'il y a de régions dans la partition fine.

Nous verrons par la suite qu'il existe des algorithmes plus performants, qui introduisent dans le processus d'inondation la notion d'un arbre d'absorptions. Ces algorithmes font une seule inondation du gradient, laissant tous les minima se propager. Cependant, le but de cette inondation est double : d'un côté on cherche à obtenir une partition de l'image, mais de l'autre côté on cherche aussi à établir un *historique des absorptions parmi les lacs*.

Classiquement, l'information liée aux absorptions s'organise sous la forme d'un arbre nommé *arbre de lacs critiques* (voir Figure 4.16). La construction de l'arbre de lacs critiques est faite parallèlement à l'inondation du relief. Au début du processus d'inondation chacun des minima se constitue comme une feuille de l'arbre. Lorsque deux lacs se rencontrent, ils

forment ce qu'on nomme un *lac critique*. Sur la structure de l'arbre cet instant est représenté par un nœud sur lequel deux arêtes s'embranchent. De cette fusion, seulement l'arête du lac absorbant survit. Ce processus se termine avec une unique région à la racine de l'arbre.

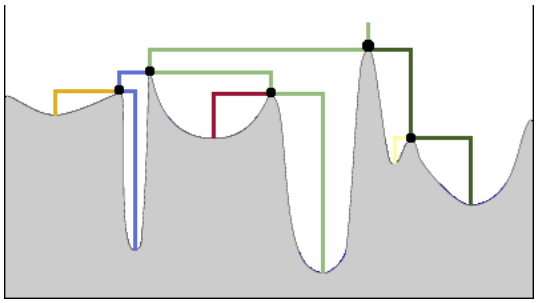


FIG. 4.16 – Construction de l'arbre des lacs critiques lors de l'inondation du gradient.

En procédant de cette façon les bassins versants séparés par les point cols les plus bas vont fusionner en premier, tandis que ceux qui sont séparés par les lignes de crête les plus hautes seront les derniers à disparaître. L'hypothèse sous-jacente consiste à supposer que les contours les plus forts entourent les vrais objets, par opposition aux frontières faibles qui s'établissent à l'intérieur des zones texturées.

L'arbre de lacs critiques permet de classer toutes les régions dans un ordre hiérarchique. Cependant, il nous reste à faire quelques remarques concernant les critères auxquels on peut avoir recours pour établir le sens des absorptions, comme nous l'avons déjà fait pour les marqueurs contraints. Parmi les critères possibles, mentionner :

- *Critère d'absorption basé sur la dynamique* : on considère que le lac le plus profond absorbe le plus superficiel. Sur la partition finale ce critère retient les régions les plus contrastées que celles-ci soient grandes ou petites (Figure 4.17 (a)).

- *Critère d'absorption basé sur la surface* : on considère que le lac ayant la surface la plus grande absorbe le plus petit. Sur la partition finale ce critère donne priorité aux régions les plus larges (Figure 4.17 (b)).

- *Critère d'absorption basé sur le volume* : comme mélange des deux critères antérieurs, celui-ci utilise le volume d'un lac comme mesure discriminante. Sur la partition finale c'est ce critère qui approche le mieux le sens visuel des objets (Figure 4.17 (c)).

De nombreux exemples comparant ces trois critères peuvent être trouvés dans la thèse de Vachier [100].

(a) dynamique (b) surface (c) volume

FIG. 4.17 – Sélection des 30 régions les plus importantes en utilisant différents critères d'absorption.

4.3.2 Création d'une partition précise de la scène

En s'appuyant sur la souplesse de cette représentation emboîtée de l'information, on doit procéder en dernier lieu à la composition de la partition finale. Or, le but des techniques de segmentation multiéchelle ne s'arrête pas à l'obtention d'une description hiérarchique du contenu de l'image, mais à la création d'une partition précise de la scène.

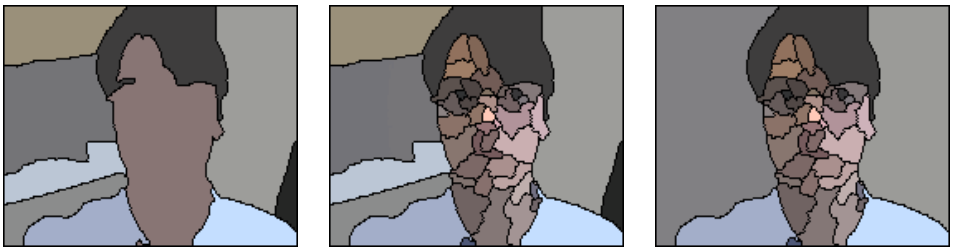
En l'absence de marqueurs ou de toute autre information concernant les objets à segmenter, le système peut éviter la sur-segmentation en proposant directement comme résultat une des partitions de la hiérarchie. Pour décider du nombre de régions de cette partition, on fait appel à des hypothèses concernant la taille ou le niveau d'homogénéité des objets.

Cependant, lorsqu'il s'agit de la segmentation d'un objet complexe, il se peut qu'aucune des partitions ne s'y adapte complètement. Ceci arrive lorsqu'on précise le nombre de régions et certains objets ont fusionné de façon incorrecte. C'est également le cas lorsqu'on voudrait que la partition ait plus de précision sur l'un des objets et moins dans le reste.

Pour faire face à ce type de situation, certains systèmes de segmentation multiéchelle ont présenté la composition de la partition finale sous la forme d'un processus interactif [108]. Ces approches donnent à l'utilisateur des moyennes pour manipuler la hiérarchie, tout en mélangeant des régions en provenance de différentes partitions jusqu'à ce que le résultat soit satisfaisant. Il suffit de prendre comme point de départ la partition de la hiérarchie qui s'approche le plus du découpage recherché. Ensuite on peut *descendre dans la pyramide* à l'intérieur des régions en demandant un plus grand nombre de régions, de la même façon qu'on peut *monter dans la pyramide* en permettant aux régions sans intérêt de fusionner.

En guise d'exemple, dans la Figure 4.18 nous allons nous fixer comme but l'extraction du personnage en préservant les traits de son visage. Parmi toutes les stratégies qui seraient envisageables, celle que nous avons choisi procède en deux étapes : sur la partition à 10 régions (a), nous avons demandé dans un premier temps de descendre dans la pyramide à l'intérieur de la région du visage (b), et ensuite nous avons fusionné le reste des régions dans le fond pour mieux apercevoir le contour du personnage (c). Notons que les partitions (b) et (c) ne font pas partie de la hiérarchie, car elles ont été créées par composition de régions provenant

de plusieurs niveaux de la pyramide.



(a) partition de la hiérarchie à 10 régions

(b) augmentation du détail dans le visage

(c) simplification du fond

FIG. 4.18 – Création de la partition finale par composition de régions procédant de différents niveaux de la pyramide.

4.4 Conclusions

Dans ce chapitre nous avons abordé, d'un point de vue généraliste, le problème lié à la segmentation d'images. Dans la ligne de nos travaux, nous avons repris un algorithme morphologique, la Ligne de Partage des Eaux, réputé pour sa versatilité et sa robustesse. En premier lieu nous avons décrit son mode de fonctionnement, pour ensuite rentrer dans l'analyse des résultats. Pour éviter une sur-segmentation de la scène, l'usage de marqueurs s'est montré très efficace. Par contre, la propagation des marqueurs étant indépendante des caractéristiques intrinsèques des régions, dans certains cas la LPE ne réussit pas à isoler les objets marqués. Dans le but de donner une solution à ce problème, nous avons proposé une nouvelle formulation du processus d'inondation classique en introduisant un ensemble de lois d'absorption comme le font les algorithmes de fusion de régions.

Nous sommes à présent en possession d'un ensemble d'outils puissants pour segmenter une scène. Parmi les nombreuses applications qui peuvent être envisagées lorsqu'on dispose d'une image segmentée, nous allons dédier la deuxième partie de ce mémoire au domaine du suivi d'objets.

Deuxième partie

La Mise en Correspondance : de l'image à la séquence

Chapitre 5

Introduction

Jusqu'à présent nous avons illustré la difficulté de la segmentation sur des images fixes. De plus, nous avons vu que, dans certains cas, l'interaction humaine était le seul moyen de reconnaître les objets en leur associant un sens sémantique. Cependant, lorsque nous voulons retrouver un objet dans une large collection d'images, ou suivre l'évolution d'un objet au long d'une séquence, la surveillance de chacune des partitions devient impossible. S'impose alors le développement d'algorithmes de mise en correspondance capables de propager l'information dans une série d'images, en limitant l'interaction humaine à une seule étape d'initialisation.

5.1 Caractérisation du problème

L'interprétation humaine des images s'appuie au plus bas niveau sur un découpage élémentaire de la scène. C'est pour cette raison que nous avons abordé l'analyse des images numériques, nous consacrant en premier lieu à l'étude et au développement de nouvelles techniques de filtrage, ensuite à la segmentation. Cependant, il nous reste à achever la tâche la plus difficile, concernant la *reconnaissance des objets* présents dans la scène. Chez les humains, le processus d'identification d'un objet fait appel à une *mémoire visuelle*, essayant de retrouver cet élément dans l'ensemble d'images faisant déjà partie de notre expérience. Cette recherche peut remonter à une image vue il y a longtemps, ou à une image vue à l'instant où l'on suit un objet du regard.

Par analogie, dans le domaine de l'analyse des images numériques, le problème de la reconnaissance d'objets consiste à répondre à cette question : en supposant qu'on dispose d'une ou de plusieurs images sur lesquelles on connaît déjà les objets d'intérêt, pourrions-nous les identifier s'ils apparaissent à nouveau dans une autre image ? Ce problème peut être posé pour atteindre deux objectifs différents : la recherche d'images dans une base de données, et le suivi d'objets dans une séquence vidéo.

5.1.1 La recherche d'images dans une base de données

Il apparaît aujourd'hui essentiel de concevoir des systèmes de tri automatique, capables de mener à terme la recherche d'information au sein des gigantesques collections d'images numériques. Or, la quantité de données disponibles augmente exponentiellement et leur gestion manuelle devient impossible. Dans ce contexte, il s'agit d'analyser des contenus extrêmement divers dans les délais les plus courts.

Les mises en œuvre les plus simples procèdent en deux étapes : dans un premier temps, de façon automatique ou interactive, elles cherchent à extraire des attributs visuels de l'image à chercher (couleurs, textures, formes, ...) ; pour ensuite lancer une requête basée sur cet ensemble de descripteurs du contenu [70]. Cependant, la performance de ces approches reste limitée tant qu'elles se basent sur une caractérisation globale de l'image.

Des requêtes plus précises peuvent être faites lors d'un accès particularisé sur une zone de l'image [94], ou sur un seul objet [29]. Pour cette dernière approche, la segmentation de l'image est requise.

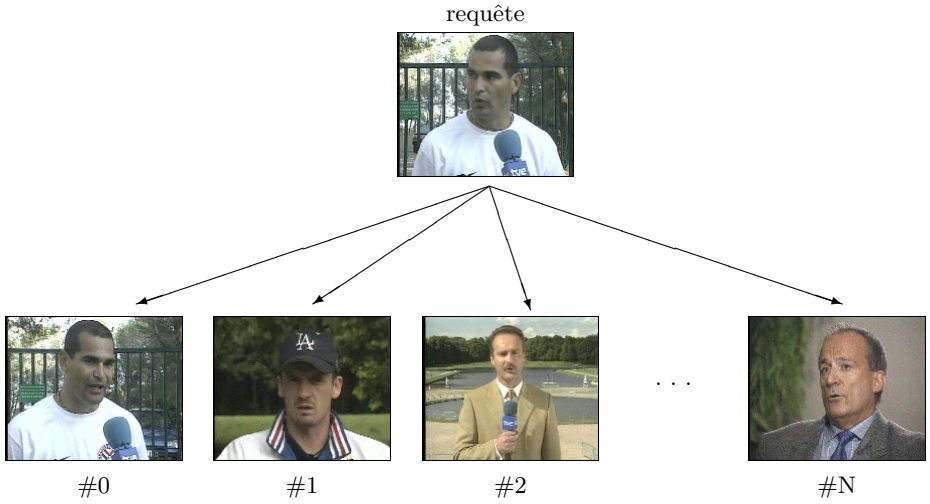


FIG. 5.1 – Recherche d'une image dans une base de données.

5.1.2 Le suivi d'objets dans une séquence vidéo

Les applications pour lesquelles on s'intéresse au suivi d'objets sont nombreuses. Certains systèmes visent à extraire une description des activités se déroulant dans une scène : la vidéo-surveillance des activités humaines [22] ou du trafic routier [48], l'analyse des gestes pour lire les lèvres [18] ou comprendre le langage des signes [72], ainsi que l'analyse de scènes dans le domaine de la vision par ordinateur [9]. Pour ce type d'applications le résultat du suivi s'exprime sous la forme d'une trajectoire.

En allant encore plus loin, certains systèmes ne se limitent pas à suivre l'évolution d'un objet mais ils cherchent à le segmenter pour l'extraire du contexte de la scène. Comme exemples les plus significatifs il faut citer les outils multimédia permettant l'accès aux objets vidéo [35], et les applications de codage de séquences qui s'adaptent au contenu [50], donnant un maxi-

mum de qualité sur l'objet suivi.

FIG. 5.2 – Suivi d'objets au long d'une séquence vidéo.

Du point de vue de la segmentation, le suivi d'objets consiste principalement à apparier les régions dans des images consécutives. Notons qu'ici la ressemblance parmi les images définit un rapport « une-à-une », contrairement à la recherche d'un objet dans une base de données dont le rapport est de « une-vers-toutes ».

5.2 Plan de notre approche

Cette deuxième partie du document sera entièrement consacré à l'étude du problème de la mise en correspondance de partitions.

Dans le Chapitre 6 nous allons aborder ce sujet de façon généraliste. Car, du point de vue des techniques de mise en correspondance agissant sur des partitions, il est possible d'établir un lien parmi les procédures de recherche d'un objet dans une base de données et les systèmes de suivi d'un objet au long d'une séquence vidéo. Dans les deux cas, il s'agit de concevoir des algorithmes capables de décider de la pertinence des appariements parmi les régions de deux partitions. A cette fin, nous allons proposer une nouvelle méthode qui combine des algorithmes classiques de mise en correspondance de graphes avec de nouvelles techniques d'édition basées sur les hiérarchies de partitions fournies par nos outils de segmentation.

La différenciation entre les deux domaines de recherche n'apparaît que lors de l'adaptation des algorithmes de mise en correspondance à l'heuristique d'une application. Ainsi, la recherche d'un objet faite à partir d'une prise de vue réelle devrait s'abstraire des paramètres de taille, disposition spatiale ou relation contextuelle ; tandis que les applications de suivi devront baser la correspondance sur la préservation temporelle de ces paramètres, en gérant le déplacement des objets ainsi que la possible apparition ou occlusion des régions.

Dans le Chapitre 7 nous allons nous focaliser sur le suivi d'objets, celui-là étant l'application visée par nos recherches [36, 37]. Ici, l'adaptation des algorithmes de mise en correspondance présentés au Chapitre 6 dans un cadre plus général, va dévoiler une nouvelle approche

très prometteuse. Pour mieux évaluer leur atouts et leur limites, nous entreprendrons aussi une discussion des performances achevées tenant compte de deux approches classiques : celles qui portent à terme une segmentation 3D en considérant toute la séquence comme un volume spatio-temporel ; et celles qui projettent la segmentation à partir d'une prédiction de la position de l'objet par analyse de mouvement.

Du point de vue de la robustesse, nos algorithmes vont surmonter de façon naturelle quelques-uns des inconvénients les plus contraignants concernant l'appariement de partitions, comme ce qui concerne l'apparition ou disparition d'objets dans la scène. Du point de vue du temps de calcul ils seront encore lents, or leur implémentation optimisée reste à faire. Cependant, le but ultime de cette comparaison n'est pas de donner une technique comme vainqueur, mais de rendre le lecteur sensible à la vraie difficulté du problème de suivi, celui-ci étant capable d'évaluer par lui même la convenance de telle ou telle approche en fonction du cadre de l'application.

5.3 Conclusions

Dans ce chapitre nous avions comme objectif d'introduire le sujet de la mise en correspondance de partitions, et plus particulièrement, de présenter le plan de notre approche afin de guider le lecteur au long de cette deuxième partie du document.

Par ailleurs, en établissant un lien parmi le problème de la recherche d'un objet dans une base de données et le problème du suivi d'objets dans une séquence, nous avons voulu montrer un nouvel éventail d'application de nos algorithmes dans le domaine du multimédia.

Chapitre 6

Mise en Correspondance de Partitions

Le problème de la mise en correspondance, comme celui de la segmentation, est l'un des problèmes les plus difficiles auxquels l'analyse d'images soit confrontée, et bien sûr il est loin d'être résolu.

Mais pourtant, nous nous sommes aventurés dans ce domaine avec l'enthousiasme de ceux qui explorent une nouvelle voie. La méthode que nous avons développée peut être définie comme étant de nature hybride, car elle combine des algorithmes classiques de mise en correspondance de graphes avec de nouvelles techniques d'édition, basées sur les hiérarchies de partitions fournies par la segmentation morphologique.

Nous verrons que cette combinaison donne lieu à un algorithme très robuste, capable de surmonter quelques-uns des inconvénients les plus contraignants concernant l'appariement de partitions : l'apparition ou disparition d'objets, l'absence de régions à cause de l'instabilité de la segmentation.

6.1 Technique de Segmentation et Appariement Conjoint

A l'égard des considérations introduites au chapitre précédent, nous avons abordé le problème de la mise en correspondance de partitions ayant à l'esprit le développement d'une technique aussi souple que possible, sur laquelle on puisse bâtir ensuite des applications variées. Nous l'avons nommé technique de Segmentation et Appariement et Conjoint (SAC).

Les *graphes* ont été adoptés comme les éléments de support de nos algorithmes, ceux-ci étant d'excellents outils de synthèse. Ainsi, dans le cadre de notre approche, la mise en correspondance de partitions revient à un problème de *mise en correspondance de graphes*. Par la suite nous présenterons les principes sur lesquels repose cette nouvelle méthode.

6.1.1 De la partition au graphe de voisinage

Les graphes sont des structures relationnelles utilisées depuis longtemps comme support des techniques d'analyse d'images [38]. On doit remonter jusqu'au début des années soixante-dix pour trouver les premiers travaux à ce sujet. Parmi ceux-ci, on peut citer l'approche de Barrow et Popplestone dans [7] qui a été l'une des premières applications des graphes dans le domaine de la vision par ordinateur, et celle de Pavlidis dans [75] introduisant les *graphes de voisinage*. Dans le domaine de la segmentation, ces derniers sont devenus des outils d'abstraction très puissants ; car leur structure s'adapte parfaitement à la topologie des partitions. Il existe

- *une correspondance un-à-un parmi les nœuds du graphe et les régions de la partition,*
- *et un arc parmi chaque paire de nœuds représentant des régions voisines.*

On doit remarquer que la topologie du graphe n'est pas figée, car le nombre et la distribution des arêtes varie en fonction de la définition de voisinage considérée. Dans notre approche nous considérerons que deux régions sont voisines s'il existe un contour en commun sur la partition. Les graphes construits sous ce critère sont connus comme *graphes de voisinage par adjacence de régions*.

La Figure 6.1 illustre un exemple du niveau de synthèse achevé lorsqu'on exprime le contenu d'une partition au moyen d'un de ces graphes. Dans la structure du graphe, chacune des régions a été labellisée et représentée par un *nœud*, et un arc créé parmi toute paire de régions voisines sur la partition.

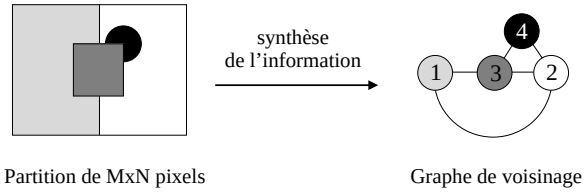


FIG. 6.1 – Description d'une partition au moyen d'un graphe de voisinage.

Il arrive que, pour certaines des applications, l'orientation des arcs (différenciation entre le nœud de départ et le nœud d'arrivée) ne joue aucun rôle. Pour ces cas-là, où on ne s'intéresse qu'à l'existence ou non-existence d'un lien parmi deux nœuds, les arcs prennent le nom d'*arêtes*.

Parallèlement, il existe la possibilité d'assigner des attributs aux éléments du graphe. Un *graphe de voisinage pondéré* assigne à chacun des nœuds, en plus d'un label, certains attributs intrinsèques aux régions. Ces attributs s'associent normalement à des primitives de couleur, de texture ou de forme. De même, certains graphes pondèrent aussi les arêtes pour mieux préciser les relations contextuelles parmi les nœuds. Dans un large nombre d'applications,

les poids associés aux arêtes ne sont que des paramètres concernant la géométrie du graphe (longueur de l'arête, orientation, ...). Néanmoins, lorsqu'il s'agit de décrire la relation de voisinage existant parmi deux régions d'une partition, d'autres possibilités peuvent aussi être exploitées. Pour en citer quelques unes, on peut garder comme paramètre le niveau de la pyramide pour lequel les deux régions fusionnent, l'ordre de profondeur parmi les régions, ou des données se référant au contour qui fait office de frontière.

A l'égard de ces considérations, la formulation de nos algorithmes sera faite en prenant à la base un graphe de voisinage défini comme suit,

Définition 15 (Graphe de voisinage pondéré) *Un graphe de voisinage pondéré est un 4-uplet du type* $\mathcal{G} = (V, E, \mu, \nu)$*, dont*

- V *est un ensemble fini de nœuds,*
- $E \subseteq V \times V$ *est l'ensemble d'arêtes connectant des paires de nœuds,*
- $\mu = \{\mu_v^1, \mu_v^2, \ldots, \mu_v^n\}$ *est l'ensemble des fonctions qui calculent les attributs des nœuds,*
- $\nu = \{\nu_e^1, \nu_e^2, \ldots, \nu_e^m\}$ *est l'ensemble des fonctions qui calculent les attributs des arêtes.*

Le nombre et caractère des attributs va dépendre du but de la mise en correspondance [30, 54]. Dans cette implémentation nous avons retenu quatre attributs pour les nœuds (la couleur, la forme, la position et lors du traitement de séquences, le déplacement).

6.1.2 La mise en correspondance de graphes

L'objectif de toute technique de mise en correspondance de graphes est d'établir un appariement parmi l'ensemble des nœuds des deux structures. Lorsque ces algorithmes s'appliquent aux graphes de voisinage, pour lesquels chacun des nœuds représente une région de la partition, ceci devient un processus de mise en correspondance de partitions. La Figure 6.2 illustre les données présentes à l'entrée et à la sortie d'un algorithme de mise en correspondance de graphes.

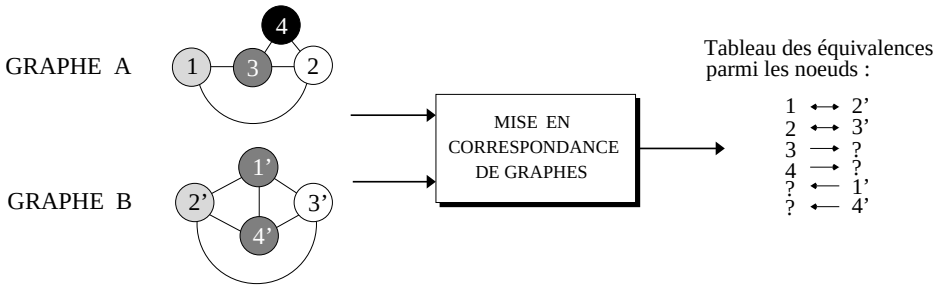


FIG. 6.2 – Mise en correspondance de graphes.

On retrouve à l'entrée les graphes de voisinage obtenus à partir de deux partitions, et à la sortie un tableau d'équivalences parmi leurs ensembles de nœuds. Notons que s'il n'existe pas une équivalence parfaite parmi les structures des graphes, il se peut que certains des nœuds ne trouvent pas de correspondant (symbolisé par « ? » dans le tableau). A l'origine, il suffit qu'un objet ait été segmenté dans un nombre différent de régions (le nœud 3 dans A apparaît fragmenté dans B), ou tout simplement qu'un objet ne soit pas présent dans les deux partitions (le nœud 4 du graphe A n'apparaît pas dans B). Par ailleurs, dans le cas où les graphes s'associent à des partitions d'une même séquence, leurs différences pourraient aussi se justifier à partir de l'analyse du mouvement des objets dans la scène.

Dans un cadre plus général, où le but ultime de l'algorithme est d'évaluer la ressemblance parmi les deux graphes à l'entrée, l'algorithme de mise en correspondance procéderait aussi au calcul d'une fonction de coût pondérant la dissimilarité des structures.

Le sujet de la mise en correspondance de graphes étant traité pour la première fois dans ce mémoire, nous allons faire un bref état de l'art. Cette revue nous permettra ensuite de mieux situer nos algorithmes par rapport aux méthodes déjà existantes.

6.1.2.1 Etat de l'art

Il existe une abondante littérature concernant la théorie des graphes et de mise en correspondance de graphes, au cœur de laquelle il est parfois difficile de s'y retrouver. Notre but ici est donc de donner une vision globale des principales approches dans ce domaine.

Depuis longtemps, les graphes s'appliquent avec succès au problème de la reconnaissance d'objets, comme par exemple, la reconnaissance de caractères manuscrits (Lue et al. [49]), de diagrammes (Messmer et Bunke [60]) ou même de formes dans un sens plus large (Siddiqi et al. [93]). Pour la plupart de ces applications, les graphes servent à modeler les objets connus à priori, ceux-ci étant ensuite stockés dans de larges bases de données. Le problème de l'identification d'un élément inconnu se transforme ainsi en un problème de mise en correspondance de graphes.

Les premières approches à la mise en correspondance de graphes cherchent à établir un appariement parfait parmi les graphes, connu comme *isomorphisme de graphes* (Read et Corneil [81]). Un isomorphisme de graphes est une projection bijective parmi les nœuds de deux graphes. Cette définition impose aux graphes d'avoir le même nombre de nœuds, les mêmes labels, et une structure d'arêtes identique.

Toutefois, on retrouve rarement une correspondance parfaite parmi les graphes modèles et ceux qui sont obtenus à partir des données en provenance d'une prise réelle, soit à cause du bruit dans le processus d'acquisition, soit à cause des erreurs d'interprétation lors de l'abstraction de la structure du graphe.

Pour surmonter ces inconvénients, il a été nécessaire de concevoir des techniques de mise en correspondance *approximatives*, dites aussi *inexactes*. Parmi celles-ci on peut différencier clairement trois types d'approches :

Mise en correspondance par isomorphisme restreint à des sous-graphes

Ces techniques cherchent à identifier des sous-structures identiques à l'intérieur de structures plus larges (Ullman [99], Shapiro [91]). Dans cette catégorie d'algorithmes, Barrow et Burstall [6] formulent pour la première fois le problème de la mise en correspondance de structures comme la recherche de la *clique maximale* dans un *graphe d'association*. Le graphe d'association est un graphe auxiliaire dont chaque nœud est créé par appariement de deux nœuds des graphes d'entré. Notons qu'il est possible d'imposer des contraintes pour limiter le nombre d'appariements et ainsi la taille du graphe. Deux nœuds sont connectés sur le graphe d'association, si et seulement si leurs topologies sur les graphes d'origine sont compatibles. Tout sous-ensemble de nœuds mutuellement connectés dans le graphe d'association est connu comme *clique*. Sous cette approche, la mise en correspondance des graphes revient à la recherche de la clique maximale, c'est-à-dire, celle qui contient le plus de nœuds. Car cette approche ne sera pas reprise dans cette thèse, pour des exemples concernant la mise en œuvre, nous reportons le lecteur aux travaux de Bolles dans [12] pour la reconnaissance d'objets, et ceux de Horaud et Skordas [43] dans le domaine de la mise en correspondance d'images stéréo.

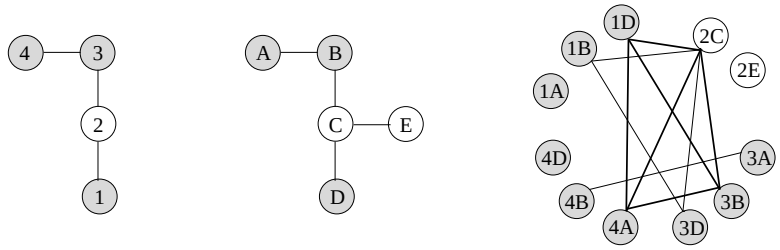


FIG. 6.3 – Graphe d'association et clique maximale.

Mise en correspondance par édition des graphes

Dans cette catégorie nous avons ordonné toutes les techniques qui incorporent des *opérations d'édition* au processus de mise en correspondance. Les opérations d'édition permettent de relabelliser, effacer ou insérer des nœuds ainsi que des arêtes jusqu'à ce que l'isomorphisme de graphes soit atteint. Cette idée a été originalement proposée par Sanfeliu et Fu dans [85] et fortement développée par Bunke dans de nombreux articles [17, 15, 60, 16]. Pour toutes ces approches, il s'agit de calculer une distance parmi les graphes en fonction du nombre d'éditions nécessaires pour obtenir des structures identiques. A chaque classe d'opération on devra lui associer un coût d'édition, de façon à que la mise en correspondance de graphes devienne la minimisation de cette *fonction de coût*.

Mise en correspondance par inclusion de nœuds fictifs

Finalement, faisant partie des techniques de mise en correspondance approximatives, on doit classer dans une catégorie à part les méthodes qui traitent les différences parmi les graphes en introduisant une nouvelle classe de nœuds nommés *nœuds fictifs*. L'idée ici est d'apparier les nœuds qui causent la différence parmi les structures vers des nœuds fictifs. Lors de la mise en correspondance nous allons inclure autant de nœuds fictifs

que nécessaire pour assurer un appariement consistant des voisinages. Shapiro et Haralick [91, 92] utilisaient cette stratégie pour la première fois sans pénaliser l'inclusion de nœuds fictifs, d'autres approches comme celle adoptée par Boyer et Kak dans [13] ainsi que celle de Christmas et al. dans [21] prévoient un coût d'insertion.

Le problème de la mise en correspondance étant très complexe, il n'est pas possible de présenter une seule de ces techniques comme la plus performante. Tout simplement, le choix de telle ou telle stratégie va être relié au type d'application visée.

Mais quelle que soit la démarche choisie, on constate une croissance exponentielle de la complexité de calcul en fonction de la taille des graphes sous analyse. Or il est bien connu que la recherche d'un isomorphisme est un problème NP-complet [28]. Ce facteur a nettement limité l'extension de ces techniques au traitement de données réelles.

De la même façon que pour les techniques de segmentation, les algorithmes de mise en correspondance de graphes se sont fortement diversifiés, à la recherche d'une solution optimale face à une problématique complexe.

La plupart des méthodes appliquées à des situations réelles ont dû incorporer de l'heuristique afin de simplifier le problème et d'augmenter la robustesse des appariements. Pour ceci on a fait appel à des techniques approximatives du type :

- *Relaxation discrète ou probabiliste* [106, 21]
- *Réseaux neuronaux* [26, 76, 42]
- *Algorithmes génétiques* [23, 102]

Quelques unes de ces méthodes arrivent à traiter le problème de la mise en correspondance avec une complexité computationnelle d'ordre polynomial par rapport au nombre de nœuds. Cependant, on doit signaler qu'en faisant de telles approximations il se peut que la solution optimale ne soit pas trouvée.

Nous conclurons ainsi la revue introductive des techniques concernant la mise en correspondance de graphes. Par la suite nous préciserons l'ensemble des choix qui ont été faits dans le cadre de notre approche.

6.1.2.2 Choix adoptés

Lorsqu'on décide de traiter le problème de la mise en correspondance de partitions comme un problème de mise en correspondance de graphes, un certain nombre de précisions doivent être apportées. Nous allons les détailler par la suite, tout l'ensemble étant résumé dans le Tableau 6.1. Cette caractérisation nous permettra en même temps de situer notre approche par rapport à des variantes possibles. Les choix à faire concernent :

Type d'application : on doit commencer en précisant si les algorithmes vont être conçus pour donner des solutions vis-à-vis d'une application précise, ou bien s'ils devront s'appliquer à plusieurs domaines. *Nos algorithmes se rangent dans la deuxième de ces*

catégories, et ça ne sera que dans le chapitre suivant qu'ils seront optimisés au problème du suivi d'objets au long d'une séquence.

Type de graphe : ensuite, on doit spécifier à quelle classe appartient le graphe de voisinage utilisé pour décrire la partition. *Nous allons construire un graphe de voisinage par adjacence de régions à partir d'une partition fournie de l'image. Il existera une équivalence un-à-un parmi les nœuds et les régions, et une arête parmi chaque paire de nœuds ayant un contour en commun sur la partition.*

Contraintes sur la structure du graphe : en fonction du type de graphe, il se peut qu'aucune contrainte ne soit imposée sur la structure, mais il se peut aussi qu'on soit forcé de préserver une certaine topologie. *En considérant la six connexité, notre graphe de voisinage sera, par définition, un graphe planaire. C'est-à-dire, la structure du graphe pourra toujours se représenter sur un plan sans que deux de ses arêtes ne se croisent.*

Type d'attributs : s'il s'agit d'un graphe pondéré, on doit encore préciser le type d'attributs utilisés. Il faut noter qu'agir sur un graphe pondéré augmente la fiabilité de la mise en correspondance, car le poids d'une assignation ne reposera pas exclusivement sur le bon assemblage des structures, mais aussi sur la coïncidence des propriétés du graphe. *Pour cette raison, nous travaillerons avec des graphes pondérés. Cependant, afin de garder le principe de généralité de notre approche, nous ne préciserons pas ici les valeurs abstraites sur la partition ; or celles-ci dépendent nettement de l'application visée.*

Taille du graphe : nous allons classer les graphes en trois catégories en fonction du nombre de nœuds : petits (moins de 100 nœuds), moyens (de 100 à 1000 nœuds) ou grands (plus de 1000 nœuds). Il est fortement recommandé de travailler avec des graphes aussi petits que possible. Autrement, leur mise en correspondance, même en ayant recours à des techniques approximatives, peut donner des temps de traitement trop élevés pour la plupart des applications. *Afin de pouvoir traiter des images réelles dans un délai acceptable, nous allons apparier des partitions n'ayant qu'une cinquantaine de régions. Cette valeur détermine la précision du système, étant en relation avec la taille des objets par rapport à la taille du support de l'image.*

Densité d'arêtes : de la même façon nous parlerons d'une densité d'arêtes faible (du même ordre que le nombre de nœuds), moyenne ou haute (d'ordre quadratique par rapport au nombre de nœuds). Notons que la taille des voisinages varie en fonction de la densité d'arêtes. Par conséquent, si d'un côté une forte densité d'arêtes permet une mise en correspondance plus robuste, elle cause aussi un incrément du nombre d'opérations nécessaires. *Dans un graphe de voisinage planaire, le nombre d'arêtes est limité à* $3n-6$*, où* $n \geq 3$ *est le nombre de nœuds.*

Type de mise en correspondance : un autre point important correspond au choix de l'algorithme de mise en correspondance de graphes. Bien que toute technique de mise en correspondance de graphes puisse être adoptée, on doit préciser s'il s'agit d'un algorithme exhaustif, qui cherche la solution optimale (isomorphisme de graphes), ou s'il s'agit d'une méthode approximative (clique maximale, édition du graphe, ou inclusion de nœuds fictifs). Les algorithmes exacts sont les moins complexes mais sont difficilement

Type d'application	Précise √ **Générique**
Type de graphe	Triangulation de Delaunay Graphe de Gabriel √ **Voisinage par adjacence de régions** Autres
Contraintes sur la structure du graphe	Aucune √ **Planaire** Arbre Autres
Type d'attributs	Aucun Symbolique Numérique √ **Structures**
Taille du graphe	√ **Petit (moins de 100 nœuds)** Moyen (de 100 à 1000 nœuds) Grand (plus de 1000 nœuds)
Densité d'arêtes	Faible (du même ordre que le nombre de nœuds) √ **Moyenne** Haute (d'ordre quadratique par rapport au nombre de nœuds)
Type de mise en correspondance	Exacte : Isomorphisme Inexactes : Recherche de la clique maximale Isomorphisme par édition des graphes √ **Isomorphisme par inclusion de nœuds fictifs**
Catégorie de l'algorithme	Optimal, avec ou sans heuristique Approximatif : Réseaux neuronaux Algorithmes génétiques √ **Relaxation probabiliste** Autres

TAB. 6.1 – Choix concernant la mise en correspondance de graphes. Les options cochées correspondent aux caractéristiques de notre algorithme.

applicables aux problèmes réels. *Notre méthode cherchera à établir une correspondance même s'il n'existe pas d'isomorphisme parmi les graphes, capable de gérer la présence de nœuds non-communs par inclusion de nœuds fictifs.*

Catégorie de l'algorithme : plus en détail, on doit spécifier la catégorie de l'algorithme choisi, différenciant ceux qui incluent des heuristiques, de ceux qui font appel à de techniques récursives (processus de relaxation, réseaux neuronaux, algorithmes génétiques, etc.). *Nous avons choisi un algorithme de relaxation probabiliste, nous permettant d'inclure certaines contraintes heuristiques pour augmenter la performance au niveau du temps de calcul.*

6.1.3 Analyse de la problématique

Jusqu'ici nous avons vu qu'il est possible de formuler le problème de la mise en correspondance de partitions sur la base des graphes. Cependant, cette approche a été classiquement considérée non viable pour deux raisons :

- l'instabilité liée à la création des partitions donne lieu à des graphes de voisinage ayant des structures non-isomorphes. Par conséquent,
- leur mise en correspondance doit être faite au moyen d'une technique approximative, toujours complexe et coûteuse en temps de calcul.

En effet, la difficulté associée à la segmentation automatique d'images est déjà bien connue du lecteur. Il suffit d'un petit changement dans la disposition de la scène, ou même d'une petite variation du contraste, pour que les régions fusionnent dans un ordre différent, susceptible d'altérer la partition finale. La Figure 6.4 montre un exemple de ce fait.

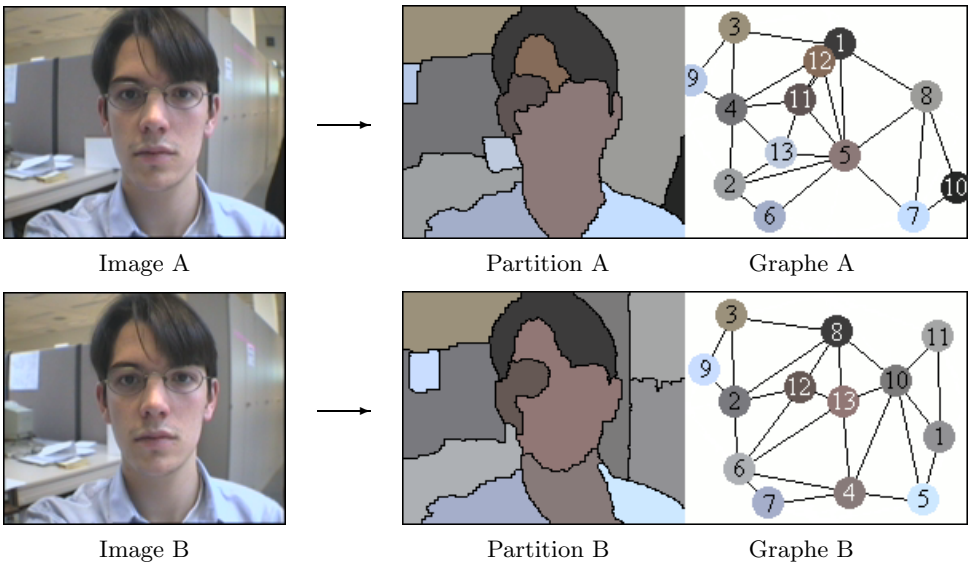


FIG. 6.4 – Problématique liée à la mise en correspondance de partitions.

Nous avons voulu comparer ici les partitions associées à deux images d'une même séquence. Le calcul des partitions a été confié à la segmentation multiéchelle présentée dans la Section 4.3. Sans avoir aucune information a priori sur le contenu des images, la finesse de la segmentation a été fixée arbitrairement à 13 régions. Le graphe de voisinage a été ensuite calculé sur cette partition de la hiérarchie. On peut observer sur cet exemple comment certains des objets communs aux deux trames, apparaissent découpés différemment sur les deux partitions. De même, il est facile de constater jusqu'à quel point de tels changements altèrent les structures des graphes de voisinage.

6.1.4 Recherche d'une solution

A l'égard des solutions classiques qui font face à l'absence d'isomorphisme parmi les structures des graphes, nous avons tenté dans un premier temps de surmonter les différences parmi les partitions à l'aide d'un algorithme de mise en correspondance de graphes permettant l'édition des nœuds conflictuels.

De plus, la hiérarchie de partitions sur laquelle s'appuie le graphe de voisinage permet d'exprimer toute opération d'édition agissant sur les nœuds comme un processus de fusion (suppression des nœuds) ou de resegmentation de régions (inclusion de nouveaux nœuds). Le but étant de regrouper sur la même partition des régions en provenance de différents niveaux de la pyramide, jusqu'à ce que l'isomorphisme parmi les structures des graphes soit atteinte, ou jusqu'à ce que toute possibilité de fusion ou de resegmentation de régions ait été explorée. Néanmoins, un tel processus de mise en correspondance s'est montré lent et complexe ; car

- sans avoir une connaissance a priori de l'enchaînement correct des opérations d'édition nécessaires pour réorganiser les régions de la partition, le calcul de toutes les combinaisons possibles est extrêmement lourd. D'autant plus que,

- pour chacune de ces combinaisons, l'algorithme doit refaire tous les calculs relatifs à la propreté de la mise en correspondance, afin de vérifier si de tels changements favorisent ou empêchent le recalage des structures.

La complexité des calculs faisait de nos algorithmes des outils difficilement utilisables pour des applications réelles, raison pour laquelle nous avons abandonné cette approche.

Ainsi, nous avons reposé le problème de la mise en correspondance de partitions dans le but clair de simplifier au maximum la procédure concernant l'appariement des graphes.

Pour cela, nous avons commencé en étudiant les causes qui sont à l'origine des différences parmi les partitions. Cette analyse nous a permis d'identifier trois typologies différentes (voir Figure 6.5) :

Absences : lorsque le même objet est présent dans les deux partitions, il peut être découpé dans un nombre différent de régions ; ainsi il n'existe pas une correspondance un-à-un parmi les noeuds (Partitions A-B).

Inclusions : lorsque de nouveaux objets apparaissent, les nouveaux noeuds altèrent la structure locale du graphe (Partitions C-D).

Déplacements : lorsque les deux partitions appartiennent à la même séquence, les objets en mouvement changent les structures de leur voisinage ou même le nombre total de noeuds à cause des occlusions temporelles (Partitions E-F).

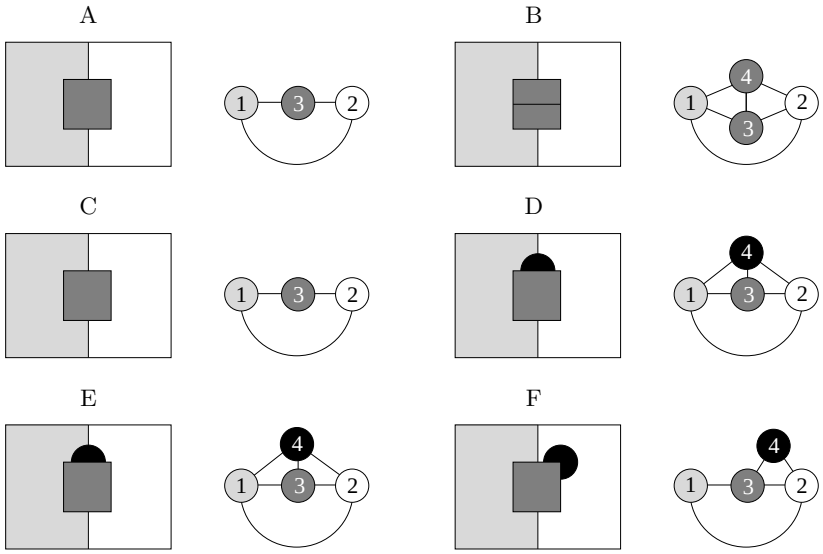


FIG. 6.5 – Causes des différences parmi deux partitions.

Etant donné que les deux derniers groupes englobent des événements qui font vraiment partie de la scène, le processus de mise en correspondance doit être capable de les gérer à l'égard d'une technique approximative. Cependant, les absences sont des différences causées exclusivement par le processus de segmentation. De plus, ce type d'erreur peut fortement déranger le processus de mise en correspondance ; or il change la structure du graphe ainsi que le nombre de nœuds.

En tenant compte de tout ceci, nous avons réorienté nos recherches vers la conception d'une technique de mise en correspondance de partitions de nature hybride, enchaînant des algorithmes morphologiques d'édition de partitions avec des algorithmes classiques de mise en correspondance de graphes. Dans la figure qui suit nous avons illustré le schéma qui résume la chaîne complète de traitement :

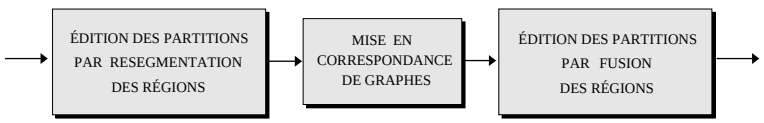


FIG. 6.6 – Schéma de la technique de Segmentation et Appariement Conjoint.

Edition des partitions par resegmentation des régions : celle-ci constitue une étape de pré-traitement ayant pour but de faire se ressembler au maximum le contenu des deux partitions. A l'aide de la segmentation hiérarchique, nous allons faire apparaître toute région non commune aux partitions de départ, mais commune aux images originales. Ceci réduit ce que nous avons définit comme nombre d'absences, avant de procéder au calcul de la mise en correspondance proprement dite.

Mise en correspondance de graphes : suite à l'étape de resegmentation, l'appariement des graphes de voisinage est beaucoup plus simple, bien que restent à résoudre les différences par inclusion ou déplacement des nœuds. Pour cela, nous avons mis en œuvre une technique approximative qui corrige les différences parmi les structures des graphes par inclusion de nœuds fictifs.

Edition des partitions par fusion des régions : une fois que les structures des graphes ont été recalées, une dernière étape d'édition doit fusionner toutes les régions créées par resegmentation qui n'ont pas trouvé de correspondant. Ceci est nécessaire afin de garder le nombre de régions stable.

Les deux étapes d'édition seront détaillées dans la Section 6.2, précédant la présentation de la technique de mise en correspondance de graphes qui sera faite dans la Section 6.3.

6.2 Edition des partitions

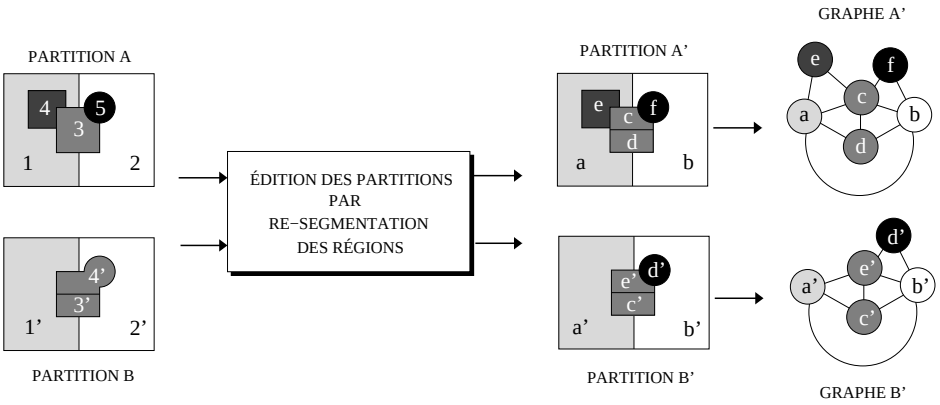
Comme nous l'avons déjà avancé, dans le cadre de nos travaux où les graphes à apparier reflètent le contenu des partitions, il se peut que certaines différences parmi les structures aient leur origine dans le processus de segmentation lui-même. Ce fait complique énormément l'obtention d'une mise en correspondance, étant donné que l'algorithme doit surmonter ces « disparités de segmentation » pour isoler les vraies différences. Nous verrons par la suite comment éviter ces problèmes au moyen de l'édition des partitions.

Une vue globale de la méthode est donnée dans la Figure 6.7 à l'aide d'un exemple synthétique, qui servira de base à la présentation des algorithmes d'édition dans tout ce qui suit. Le lecteur pourra trouver des résultats obtenus sur des images réelles à la fin de ce chapitre.

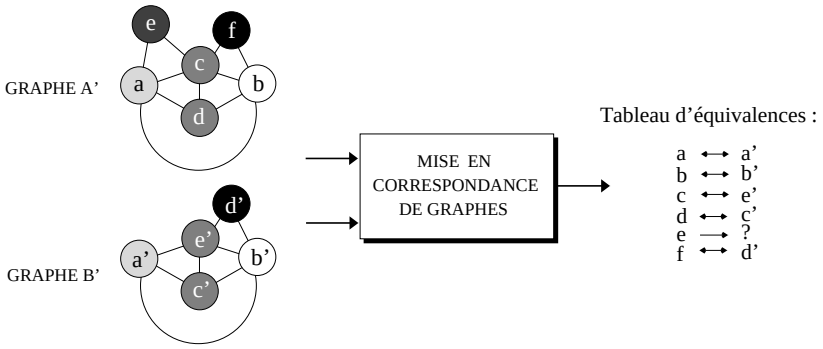
6.2.1 Edition par resegmentation de régions

Comme nous l'avons déjà annoncé, le but du processus d'édition par resegmentation des régions est de rapprocher les découpages de la scène obtenus dans les deux partitions, afin de donner un meilleur support de calcul à l'algorithme de mise en correspondance de graphes.

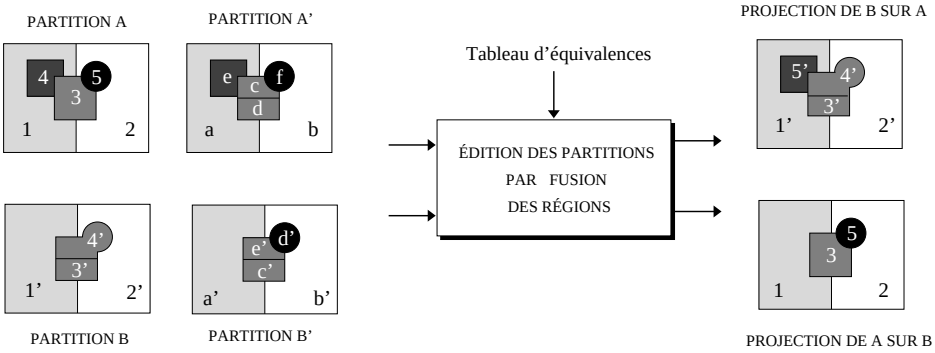
La Figure 6.8 résume la procédure de resegmentation en quelques images. L'algorithme d'édition prend comme données à l'entrée deux partitions, A et B, n'ayant forcément ni le même nombre de régions, ni les mêmes relations de voisinage. Chacune de ces partitions provient d'une hiérarchie, où le découpage de la scène se fait à différents niveaux de détail. Il se peut que certains objets ne soient présents que dans l'une des deux images (région 4 de A),



Etape 1 : édition des partitions par re-segmentation des régions.



Etape 2 : mise en correspondance des graphes par relaxation probabiliste.



Etape 3 : édition des partitions par fusion des régions.

FIG. 6.7 – Mise en correspondance de partitions au moyenne des graphes de voisinage.

mais il se peut aussi que des objets communs aient été segmentés différemment, comme c'est le cas des régions 3 et 5 dans A, qui ne peuvent pas être strictement mis en correspondance avec aucune des régions de B.

Le but de l'édition est donc de resegmenter les partitions A et B de façon à que toute région en commun sur les images de départ apparaisse également découpée sur des partitions plus fines, que nous appellerons A' et B'. La partition A' est le résultat de la resegmentation de la partition A, et de manière similaire, la partition B' est obtenue par resegmentation de B. Plus précisément, la composition des partitions A' et B' est faite à partir des régions obtenues par intersection de A et B. Ainsi nous dirons que,

– la partition A' (resp. B') se compose de toutes les régions de A∩B qu'on peut retrouver dans un niveau quelconque de hiérarchie de A (resp. B).

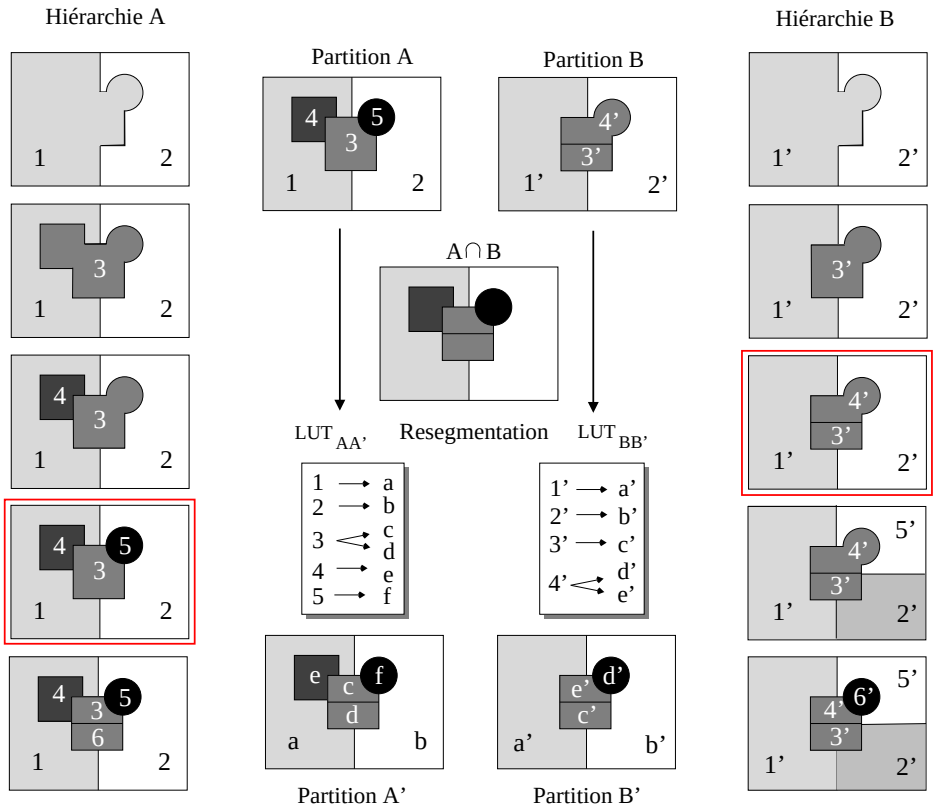


FIG. 6.8 – Processus d'édition des partitions par resegmentation des régions.

A ce point, on doit remarquer que,

– tout contour inclus dans A (resp. dans B) est aussi inclus dans A' (resp. dans B'), ceci étant une propriété de la hiérarchie de partitions utilisée pour faire de la resegmentation des régions.

– les partitions A' et B' peuvent ne correspondre avec aucune des partitions des hiérarchies

d'origine, car elles incluent de régions en provenance de différents niveaux de l'échelle de segmentation.

On doit également souligner qu'un tel processus d'édition n'assure en aucun cas l'isomorphisme parmi les partitions résultantes. Car, il est évident que si un objet n'est pas présent dans la scène, il ne ressortira jamais par resegmentation de l'image.

Cependant, avec ce processus nous avons réussi à transformer les partitions de départ en deux partitions plus fines et beaucoup plus proches l'une de l'autre. De cette étape on devra garder également l'ensemble des équivalences parmi les régions de départ et les nouvelles régions, afin de permettre de revenir en arrière sur l'étiquetage après l'étape de calcul de la mise en correspondance des graphes.

Il faut rappeler ici que la technique de mise en correspondance de graphes sera décrite en détail dans la Section 6.3. Nous nous limiterons à présent à illustrer dans la Figure 6.7 les données à l'entrée et à la sortie de ce processus, le but étant de fournir un tableau d'équivalences parmi l'ensemble des nœuds des graphes de voisinage correspondant aux partitions A' et B'. Par cohérence dans la présentation du processus d'édition, nous détaillerons par la suite l'algorithme d'édition par fusion de régions.

6.2.2 Edition par fusion de régions

Cette dernière étape va conclure le processus de mise en correspondance en établissant un chemin d'étiquetage qui relie les deux partitions de départ (A et B). Dans la Figure 6.9 nous avons regroupé l'ensemble des données nécessaires pour accomplir cette tâche, ainsi que les projections résultantes. L'algorithme de fusion de régions doit disposer des partitions fines (A' et B') créées par resegmentation dans la première étape d'édition, ainsi que des tableaux qui gardent les relations de pères à fils. Les équivalences fournies par l'algorithme de mise en correspondance de graphes nous permettront ensuite de fermer le lien.

A la sortie de l'algorithme, le résultat de la mise en correspondance de partitions peut s'exprimer doublement par :

Projection de A sur B : dans le *sens direct*, la partition A se constitue comme *partition de référence*, et propage ses labels sur la partition B', celle-ci étant la représentation de B la plus proche de A. Le résultat est obtenu en enchaînant les conversions de $LUT_{BB'} \circ LUT_{B'A'} \circ LUT_{A'A}$ sur la partition B.

Projection de B sur A : étant donné que les relations d'équivalence parmi les labels ne sont pas directionnelles, nous pouvons aussi propager les labels dans un *sens inverse*, en inversant tout simplement les rôles de A et de B. Le résultat est obtenu en enchaînant les conversions de $LUT_{AA'} \circ LUT_{A'B'} \circ LUT_{B'B}$ sur la partition A.

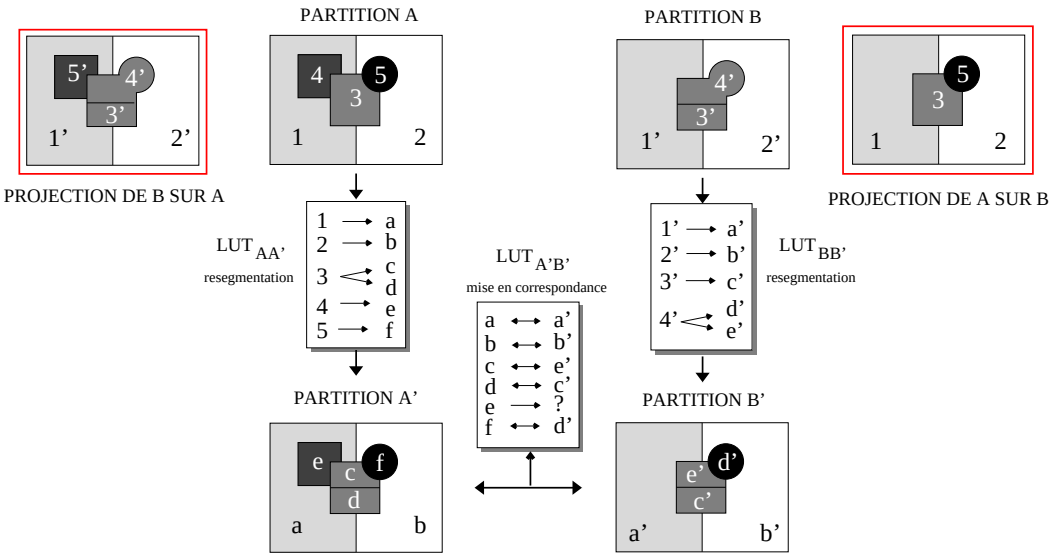


FIG. 6.9 – Processus d'édition des partitions par fusion des régions.

Ce double affichage du résultat constitue un atout supplémentaire de notre méthode, donnant plus de robustesse au calcul d'une fonction distance parmi les deux partitions pour des applications où l'objectif est d'établir une mesure de similarité. Mais, quel que soit le sens de la projection, le but de l'algorithme de fusion est toujours le même : dériver comme résultat de la mise en correspondance la partition la plus proche à celle qui joue le rôle de référence, tout en restant fidèle au découpage fourni par la segmentation multiéchelle. Néanmoins, il se peut que la propagation de labels ne soit pas directe s'il y a des régions dans A' ou B' qui n'ont pas été appariées lors de la mise en correspondance des graphes. Dans ces cas-là, l'algorithme procède de la façon suivante,

Cas 1 : si une région de A' ou B', aussi présente sur les partitions originales, n'as pas trouvé de correspondante, l'algorithme de fusion suppose qu'elle représente un *nouveau objet* et lui assigne un nouveau label lors de la projection. Ceci est le cas de la région 4 de la partition A qui se correspond à aucun des labels de la partition B et qui reçoit pourtant le label 5' lors de la projection de B sur A.

Cas 2 : si une des régions originales apparaît resegmentée dans A' ou B', mais aucune de ces sous-régions n'a trouvé de correspondante, l'algorithme va les fusionner à nouveau en supposant que l'ensemble représente un *nouvel objet*. On doit noter qu'un seul label est assigné.

Cas 3 : si une des régions originales apparaît resegmentée en plusieurs sous-régions dans A' ou B', et au moins la sous-région la plus large a été appariée lors de la mise en correspondance, l'algorithme va fusionner tous les autres morceaux non appariés, en considérant que la resegmentation avait été causée par une variation dans la forme de l'objet.

Cas 4 : si une des régions originales apparaît resegmentée en plusieurs sous-régions dans A' ou B', et la sous-région la plus large n'a pas trouvé de correspondante, l'algorithme suppose qu'elle représente un *nouvel objet*. Tous les autres morceaux n'ayant pas trouvé de correspondant seront fusionnés.

Le but principal de l'algorithme de fusion de régions est donc de gérer l'assignation de labels pour les régions qui n'ont pas été appariées lors de la mise en correspondance. De plus, l'inclusion d'une dernière étape de fusion est nécessaire pour minimiser le nombre de nouveaux labels qui apparaissent lors de la discordance parmi les régions présentes dans les deux partitions.

Pour conclure la présentation des techniques d'édition, nous allons rappeler brièvement leurs atouts. L'édition des partitions sur la base d'une décomposition hiérarchique de la scène :

- permet de résoudre les problèmes d'instabilité dûs à la segmentation.
- simplifie énormément la complexité du processus de mise en correspondance.
- est indépendante de la technique de mise en correspondance de graphes appliquée.

6.3 Algorithme de mise en correspondance de graphes

Dans cette section nous allons présenter l'algorithme de mise en correspondance agissant sur les graphes des partitions après resegmentation. Formellement, notre approche repose sur les principes de labellisation des processus de *relaxation probabilistes* présentés par Rosenfeld et al. dans [82] et élargis par Hummel et Zucker dans [46].

Les algorithmes de labellisation par relaxation sont une classe de mécanismes originalement développés pour gérer la présence d'ambiguïté dans le domaine des systèmes de vision par ordinateur. Le but est d'introduire de l'information contextuelle dans le processus de reconnaissance douteuse d'un objet. Leur principe de fonctionnement repose sur deux point clés :

- *décomposer un problème complexe de type global, en un processus récursif de calculs locaux, et*
- *inclure de l'information contextuelle pour résoudre les ambiguïtés locales.*

Grâce à leur large potentiel, les processus de relaxation ont été appliqués à des domaines d'application très variés, et plus particulièrement, de diverses manières au problème de la mise en correspondance de graphes. Le lecteur qui le désire pourra trouver une étude détaillée sur la convergence, la vitesse et la stabilité des algorithmes faite par Zucker et al. dans [111].

Par la suite nous nous limiterons à introduire leur principe de fonctionnement avant de rentrer dans les détails de notre implémentation. Dans notre approche, nous allons noter par,

- A et B : partitions originales.
- $\mathcal{G}_A$ et $\mathcal{G}_B$: graphes associés aux partitions originales.
- A' et B' : partitions après édition par resegmentation.
- $\mathcal{G}_{A'}$ et $\mathcal{G}_{B'}$: graphes associés aux partitions après édition.
- $f_{B'}^{A'} : V_{A'} \rightarrow V_{B'}$: fonction de projection des nœuds de $\mathcal{G}_{A'}$ sur les nœuds de $\mathcal{G}_{B'}$.
- $f_{A'}^{B'} : V_{B'} \rightarrow V_{A'}$: fonction de projection des nœuds de $\mathcal{G}_{B'}$ sur les nœuds de $\mathcal{G}_{A'}$.

La mise en correspondance de graphes par relaxation procède en trois étapes : les deux premières initialisent le processus en calculant la similarité entre les nœuds de façon locale, la troisième étape consiste à propager par relaxation la cohérence de ces assignations. Nous allons les détailler par la suite :

Etape 1. - Calcul de la similarité intrinsèque parmi les nœuds.

Dans un premier temps, on procède au calcul d'une mesure de similarité sur l'ensemble des appariements possibles parmi un nœud j du graphe A et un nœud i du graphe B. La totalité de ces mesures s'organise dans la forme d'une matrice, dite *matrice des permutations*, pour laquelle chaque valeur exprime la similarité parmi une paire de nœuds :

$$P(j_0 \rightarrow i_0) : \quad \textit{probabilité associée à l'appariement des nœuds } j_0 \textit{ et } i_0, \textit{ à la position } (j_0, i_0) \textit{ de la matrice des permutations.}$$

Etape 2. - Calcul de la conformité du recalage des voisinages.

Deuxièmement, l'algorithme cherche à évaluer la conformité du recalage des structures des voisinages pour chaque appariement possible (j_0, i_0). L'objectif est d'utiliser ensuite cette valeur pour pondérer les assignations de la matrice des permutations :

$$P(N_{j_0} \rightarrow N_{i_0}) : \quad \textit{probabilité associée à la conformité du recalage des voisinages } N \textit{ des nœuds } j_0 \textit{ et } i_0$$

Etape 3. - Processus récursif de mise à jour de la fiabilité des assignations.

Une fois initialisées toutes ces valeurs, l'algorithme rentre dans un processus récursif de mise à jour des valeurs de la matrice des permutations. Ainsi, si les régions j_0 et i_0 ne se correspondent pas parfaitement, on laisserait influencer la probabilité de leur mise en correspondance en fonction de la ressemblance existant parmi leurs voisins :

$$P^{(s+1)}(j_0 \rightarrow i_0) = f(P^{(s)}(j_0 \rightarrow i_0), P^{(s)}(N_{j_0} \rightarrow N_{i_0}))$$

A la fin du processus l'algorithme converge vers un appariement cohérent des nœuds, assurant que tous les labels assignés seront compatibles avec ceux de leur voisinage.

Pour la plupart des applications, le contrôle de la récursivité, ainsi que la particularisation de certains paramètres doivent être empiriquement justifiés. De même, des heuristiques liés à l'application permettent, dans de nombreux cas, d'accélérer la vitesse de traitement.

Dans notre approche nous allons introduire un seuillage permanent des valeurs de la matrice des permutations de façon à augmenter le nombre d'assignations fermes et réduire le dégré d'incertitude globale. Nous verrons ceci en détail dans les prochaines sections.

6.3.1 Calcul de la similarité parmi les nœuds

La probabilité de la mise en correspondance de deux nœuds arbitraires, $j_0 \in V_{A'}$ et $i_0 \in V_{B'}$, va être calculée de la façon suivante :

$$P(j_0 \rightarrow i_0) = \prod_{k=1...n} \Gamma^k(a^k_{j_0}, a^k_{i_0}) \tag{6.1}$$

où a^k sont les attributs des nœuds, et Γ^k sont des fonctions continues dans l'intervalle $[0, 1]$ qui mesurent la similarité parmi ces attributs. La fonction P donne ainsi une mesure de la ressemblance entre la paire de nœuds sous analyse :

- si $P(j_0 \rightarrow i_0)=0$ nous allons conclure que la correspondance parmi j_0 et i_0 est invraisemblable, leurs différences étant trop importantes. Au contraire,
- si $P(j_0 \rightarrow i_0)=1$ nous allons conclure que la correspondance est indubitable.

A partir des attributs retenus dans notre implémentation nous allons calculer la probabilité de la mise en correspondance comme le produit de trois valeurs de similarité :

$$P = \Gamma_{couleur} \cdot \Gamma_{forme} \cdot \Gamma_{position} \tag{6.2}$$

Similarité dans la couleur : elle est calculée en fonction de la ressemblance des couleurs moyennes des régions R_{j_0} et R_{i_0} :

$$\Gamma_{couleur} = \left(1 - \frac{|y_{j_0} - y_{i_0}|}{y_{max}}\right)\left(1 - \frac{\min\{\theta, \theta_{max}\}}{\theta_{max}}\right) \tag{6.3}$$

où y_{j_0} et y_{i_0} sont les valeurs de luminance et $\theta = \measuredangle((u,v)_{j_0}, (u,v)_{i_0})$ est l'angle existant entre les vecteurs chromatiques. La valeur de θ_{max} contrôle la sensibilité aux dérives chromatiques. Nous avons pris $\theta_{max} = 2\pi/3$ en considérant que deux à deux les couleurs primaires ont une similarité nulle ($\Gamma_{couleur} = 0$).

Similarité dans la forme : tant que nous disposons de la séquence de partitions, nous avons décidé de comparer la forme des régions à partir de la relation :

$$\Gamma_{forme} = \frac{\text{Surf}(R'_{j_0} \cap R'_{i_0})}{\text{Surf}(R'_{j_0} \cup R'_{i_0})} \tag{6.4}$$

où R'_{j_0} et R'_{i_0} sont les régions recalées sur un même centre de masses. Ceci évite une étape de paramétrisation des contours, mais rend le système sensible aux rotations et aux changements d'échelle. Dans le cas ou ces déformations soient courantes, nous proposons d'utiliser des descripteurs de Fourier comme l'ont fait Marqués et Gutiérrez dans [54].

Similarité dans position : comme troisième et dernière mesure de ressemblance nous allons calculer la distance entre le centre des régions,

$$\Gamma_{position} = \frac{1}{D_{max}} \sqrt{(x_{j_0} - x_{i_0})^2 + (y_{j_0} - y_{i_0})^2} \tag{6.5}$$

où (x_{j_0}, y_{j_0}) et (x_{i_0}, y_{i_0}) sont les centres de masses des régions R_j et R_i, et D_{max} est le facteur de normalisation. Dans le cas des séquences ces positions sont comparées à partir du déplacement estimé.

L'ensemble de ces mesures de similarité constitue l'initialisation de la matrice des permutations (voir Tableau 6.3 dans la Section 6.4 lors de l'analyse des résultats).

Premier seuillage de la matrice des permutations

Dans les algorithmes de relaxation classiques la probabilité des assignations douteuses de la matrice des permutations se recalcule de façon itérative jusqu'à la convergence vers une assignation optimale. Cependant, une telle approche est très coûteuse en temps de calcul, car à chaque itération toutes les valeurs différentes de 0 ou 1 devront être actualisées.

Ainsi, en suivant le raccourci proposé par Price dans [80], nous avons accéléré le processus de mise en correspondance en incluant une étape de seuillage lors de l'assignation des labels. De cette façon, une assignation ferme sera faite parmi une paire de nœuds (j_0, i_0) si, et seulement si, se vérifient les trois équations suivantes :

$$\begin{cases} P(j_0 \to i_0) & = & \max\limits_{\forall i \in V_{B'}} P(j_0 \to i) \\ P(j_0 \to i_0) & = & \max\limits_{\forall j \in V_{A'}} P(j \to i_0) \\ P(j_0 \to i_0) & > & P_{min} \end{cases} \tag{6.6}$$

C'est-à-dire, si l'appariement $(j_0 \to i_0)$ est le plus probable dans les deux sens de la mise en correspondance, et si cette valeur de P dépasse un certain seuil.

Suite à une assignation ferme, le processus de mise en correspondance va considérer :

$$\begin{array}{lcll} P(j_0 \to i_0) & = & 1 & \\ P(j_0 \to i) & = & 0, & \forall i \neq i_0 \\ P(j \to i_0) & = & 0, & \forall j \neq j_0 \end{array} \tag{6.7}$$

Ce changement de la matrice de permutations forcé par seuillage aura une double répercussion sur le processus itératif de relaxation (voir Tableau 6.4), car :

– d'un côté on ne va plus recalculer la valeur de $P(j_0 \rightarrow i_0)$, puisqu'elle est déjà maximale, et

– de l'autre côté, d'autres assignations vont réduire leur incertitude tenant compte des valeurs de $P(j_0 \rightarrow i)$ et $P(j \rightarrow i_0)$ qui ont été mises à zéro.

Notons que la matrice des permutations ne peut pas être normalisée à cause de l'étape de seuillage, raison pour laquelle P n'est pas une vraie mesure de probabilité.

Inclusion de nœuds fictifs dans la matrice des permutations

Grâce à l'étape de seuillage nous avons pu rendre fermes les appariements les plus probables. Cependant, il reste des nœuds dans la matrice des permutations pour lesquels tous les appariements possibles sont si peu probables que leur assignation est, dans l'autre extrémité de l'échelle, également indubitable : il s'agit de nœuds qui ne sont présents que dans une des deux partitions.

Dans le but de gérer ces cas, nous allons inclure des *nœuds fictifs* qui seront notés par d de l'anglais « dummy », dans l'algorithme de mise en correspondance. Rappelons que ces différences topologiques sont dûes principalement à l'apparition ou disparition d'objets, l'instabilité de la segmentation étant minimisée grâce au processus d'édition qui a précédé le calcul de la mise en correspondance.

Ainsi, une fois que toutes les assignations qui pouvaient être classées comme fermes ont été faites, une mesure de la probabilité de « venir de » ou « d'aller à » un nœud fictif va être calculée comme,

$$\begin{aligned} P(j_0 \rightarrow d) &= 1 - \max_{\forall i \in V_{B'}} P(j_0 \rightarrow i) \\ P(d \rightarrow i_0) &= 1 - \max_{\forall j \in V_{A'}} P(j \rightarrow i_0) \end{aligned} \tag{6.8}$$

Tenant compte de cette formulation, si un nœud a de très faibles ressemblances avec tout nœud du graphe contraire, la probabilité d'être mis en correspondance avec un nœud fictif sera donc élevée (voir Tableau 6.5).

Deuxième seuillage de la matrice des permutations

Finalement, la même procédure de seuillage utilisée pour les appariements des nœuds réels s'applique ici sur les appariements d'un nœud réel avec un nœud fictif. Ainsi,

– si $P(j_0 \rightarrow d) > P_{min}$, l'algorithme assume que la région j_0 de la partition A' n'apparaît plus dans la partition B' ;

– si $P(d \rightarrow i_0) > P_{min}$, nous pouvons affirmer que i_0 est un objet qui apparaît pour la première fois dans B'.

On doit remarquer qu'à la fin du processus de mise en correspondance des partitions il n'y aura pas d'appariement avec des nœuds fictifs. Toutes les régions appariées avec un nœud fictif lors de la mise en correspondance des graphes seront fusionnées ou relabellisées par le processus d'édition de partitions vu auparavant (voir Tableau 6.6).

6.3.2 Calcul de la cohérence du voisinage

Dans la plupart des cas, l'identification ferme d'un nœud peut être faite en s'appuyant exclusivement sur les attributs intrinsèques de la région, c'est-à-dire, en comparant deux nœuds sans tenir compte de la structure du graphe. Néanmoins, la reconnaissance de certains nœuds peut devenir douteuse si les objets qu'ils représentent ont changé légèrement de forme, position ou couleur. Pour ceux-ci, on peut attendre que l'information contextuelle de leur voisinage va aider à décider, ou bien en soutenant l'assignation proposée ou bien en la décourageant.

A l'égard des définitions classiques, le *voisinage par connexité* d'un nœud k_0 est formé par l'ensemble des nœuds k_i directement connectés à k_0 dans le graphe G,

$$N_{k_0} = \{k_i \in V_G \mid k_{0i} \in E_G\} \tag{6.9}$$

où k_{0i} dénote l'arête reliant les nœuds k_0 et k_i. Par la suite nous allons traiter le voisinage à partir de sous-ensembles des nœuds selon la notation suivante,

$$N_{k_0}^{k_1,\ldots,k_m} = \{k_1, k_2, \ldots, k_m\} \quad | \quad N_{k_0}^{k_1,\ldots,k_m} \subset N_{k_0}$$

Dans le but final d'évaluer la cohérence du recalage des voisinages $N_{j_0} \rightarrow N_{i_0}$, nous commencerons en définissant une mesure de similarité qui nous permet de comparer deux des nœuds voisins j_1 et i_1 :

$$P(N_{j_0}^{j_1} \rightarrow N_{i_0}^{i_1}) = P(j_{01} \rightarrow i_{01}) P(j_1 \rightarrow i_1) \tag{6.10}$$

j_{01} et i_{01} étant les arêtes qui unissent j_0 avec j_1 et i_0 avec i_1 respectivement.

Par analogie avec les nœuds, la probabilité de mise en correspondance entre deux arêtes est obtenue par produit des mesures de similarité parmi l'ensemble des attributs des arêtes :

$$P(j_{01} \rightarrow i_{01}) = \prod_{k=1\ldots n} \Gamma^k(a_{j_{01}}^k, a_{i_{01}}^k) \tag{6.11}$$

Typiquement les attributs utilisés pour comparer les arêtes correspondent à des attributs géométriques comme la longueur et l'orientation. Néanmoins, dans le cadre de notre application ces contraintes ne feraient que limiter le recalage parmi les nœuds du voisinage en mouvement. Pour cette raison nous prendrons $P(j_{01} \rightarrow i_{01}) = 1$.

Tenant compte de ceci, l'évaluation de la cohérence globale des voisinages est calculée sur le meilleur des recouvrements possibles comme :

$$P(N_{j_0} \to N_{i_0}) = \frac{1}{M} \sum_{\forall j \in S_{j_0}} \max_{\forall i \in S_{i_0}} W_{i_0 i}^{j_0 j} P(N_{j_0}^{j} \to N_{i_0}^{i}) \tag{6.12}$$

Voyons par la suite comment interpréter cette équation :

Voisinages de projection S_{j_0} et S_{i_0}

A cause des changements dans la scène, spécialement lorsque la mise en correspondance est faite sur des trames de la même séquence, nous ne pouvons pas assurer que les voisins d'un certain nœud restent toujours inchangés : quelques uns des nœuds peuvent apparaître comme faisant partie d'un autre voisinage, ou tout simplement disparaître d'une image vers la suivante. Rappeler ici l'exemple de la Figure 6.4.

Par rapport aux voisinages par connexité, N_{j_0} et N_{i_0}, où tous les voisins du nœud central sont pris en compte, nous allons définir les *voisinages de projection*, S_{j_0} et S_{i_0}, comme,

$$\begin{array}{rcll} S_{j_0} & = & \{N_{j_0} \setminus D_{j_0}\} \cup \{d\}_n, & n = \text{Card}(\{N_{i_0} \setminus D_{i_0}\}) \\ S_{i_0} & = & \{N_{i_0} \setminus D_{i_0}\} \cup \{d\}_m, & m = \text{Card}(\{N_{j_0} \setminus D_{j_0}\}) \end{array} \tag{6.13}$$

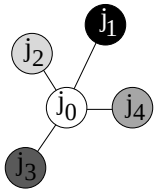
En calculant la mise en correspondance parmi les nœuds de S_{j_0} et S_{i_0}, nous voudrions éviter que la mobilité des voisins puisse faire diminuer la vraisemblance de $P(N_{j_0} \to N_{i_0})$ lorsque j_0 et i_0 sont restés immobiles. Pour cela nous allons,

- faire abstraction de l'ensemble des nœuds voisins qui ont été fermement couplés avec un nœud fictif :

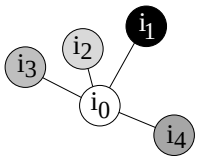
$$\begin{array}{l} D_{j_0} = \{j \in N_{j_0} \mid f_{B'}^{A'}(j) = d\} \\ D_{i_0} = \{i \in N_{i_0} \mid f_{A'}^{B'}(i) = d\} \end{array} \tag{6.14}$$

- inclure autant de nœuds fictifs que nécessaire à chaque voisinage, $\{d\}_n$ et $\{d\}_m$, pour permettre d'établir une connexion un à un parmi les nœuds réels du voisinage dans le cas où il n'y a pas de nœuds qui peuvent être appariés.

En guise d'exemple, dans la Figure 6.10 se comparent les voisinages des nœuds $j_0 \in A'$ et $i_0 \in B'$. Parmi les deux structures on a précisé les valeurs maximales de la matrice des permutations concernant ces nœuds. Observer que S_{j_0} exclut le nœud j_3, or nous pouvons affirmer en toute sécurité qu'il n'apparaît pas dans le graphe B'.



$P(j1 \rightarrow i1) = 1$
$P(j2 \rightarrow i2) = 0.5$
$P(j3 \rightarrow d) = 1$
$P(j4 \rightarrow i4) = 0.8$
$P(d \rightarrow i3) = 0.4$



$N_{j0} = \{j1, j2, j3, j4\}$
$D_{j0} = \{j3\}$
$S_{j0} = \{j1, j2, j4\} \cup \{d, d, d, d\}$

$N_{i0} = \{i1, i2, i3, i4\}$
$D_{i0} = \{\phi\}$
$S_{i0} = \{i1, i2, i3, i4\} \cup \{d, d, d\}$

FIG. 6.10 – Calcul des voisinages de projection.

Permutations dans l'ordre des voisins

La plupart des techniques de mise en correspondance de graphes associées à des images fixes imposent la préservation de l'ordre cyclique dans le voisinage. C'est-à-dire, que l'on fait l'hypothèse que chaque voisin aura toujours les mêmes nœuds à gauche et à droite dans la structure du voisinage.

Un exemple typique concerne le problème du recalage des images aériennes, abordé à de nombreuses reprises par Hancock et ses collaborateurs [23, 106, 104]. Ils procèdent à l'extraction des contours les plus saillants du relief (routes, chemins, rivières, ...) pour ensuite les organiser dans une structure de graphe qui reflète leur disposition spatiale sur le terrain. Le recalage des vues aériennes revient donc à un problème de mise en correspondance de graphes. Dans ce contexte, il est clair qu'aucun changement au niveau des données ne peut justifier une permutation des nœuds.

Par rapport à ce type d'applications, nous allons permettre des permutations parmi les nœuds du voisinage afin de pouvoir suivre les objets ayant du mouvement.

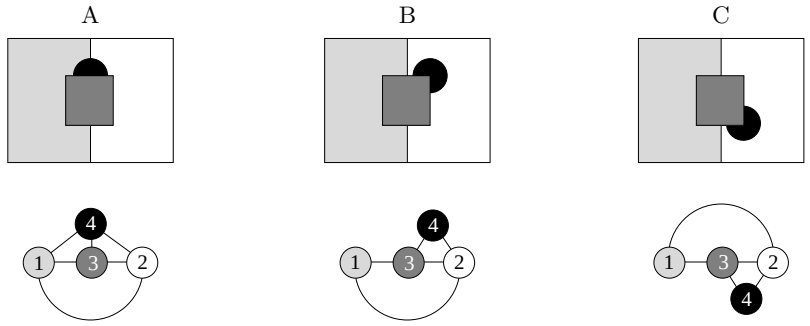
La Figure 6.11 illustre un exemple de la façon dont le mouvement peut changer la structure du voisinage d'un nœud. La séquence montrée est composée de trois partitions synthétiques $\{A, B, C\}$ pour lesquelles une des régions se déplace. En dessous de chacune des partitions on a représenté le graphe de voisinage. Tenant compte exclusivement de la topologie des graphes, on a tendance à affirmer que la partition B est plus proche de la partition C que A, or seulement dans A le nœud 4 est en voisinage avec le nœud 1. Néanmoins, si on construit une liste ordonnée des voisins de 3, on se rend compte que N_{3_B} et N_{3_C} ne peuvent pas être correctement assemblés si l'ordre cyclique parmi les nœuds doit être préservé.

Facteur de normalisation M

Le facteur de normalisation M correspond au nombre des appareillages *non vides* établis parmi les *voisinages de projection* (S_{j_0}, S_{i_0}). Ceci peut être plus formellement énoncé en définissant,

$$M = \text{Card}(\{j \mid j \neq d \text{ ou } f(j) \neq d\}) \tag{6.15}$$

dont f est la fonction de projection $f : S_{j_0} \rightarrow S_{i_0}$, et j un nœud de S_{j_0}.



$N_{3_A} = \{2_A, 4_A, 1_A\}$ $N_{3_B} = \{2_B, 4_B, 1_B\}$ $N_{3_C} = \{2_C, 1_C, 4_C\}$

FIG. 6.11 – Influence du mouvement relatif des objets dans la topologie du voisinage.

Facteur de pondération $W_{j_0 j}^{i_0 i}$

L'importance d'une relation de voisinage est très différente pour deux nœuds qui se touchent par un seul pixel, par rapport à deux nœuds qui ont une grand partie de leur contour en commun. Ceci est pris en compte par le facteur de pondération $W_{j_0 j}^{i_0 i}$ en assignant un poids à chaque relation de voisinage :

$$W_{j_0 j}^{i_0 i} = \frac{C_{j_0 j} + C_{i_0 i}}{C_{j_0} + C_{i_0}} \tag{6.16}$$

où C_{j_0} est la longueur du contour de la région j_0, et $C_{j_0 j}$ est la longueur de la partie du contour que les régions j_0 et j ont en commun (resp. C_{i_0} et $C_{i_0 i}$).

6.3.3 Processus récursif de relaxation

Etant donné que le changement dans la confiance d'une des assignations peut influencer d'autres décisions encore douteuses, nous ferons une mise à jour des valeurs de la matrice des permutations à l'intérieur d'une boucle de relaxation. Nous reprendrons ici la nomenclature classique établie par Hummel et Zucker dans [46], selon laquelle :

$$P^{(s)}(j_0 \to i_0) = \frac{P^{(s-1)}(j_0 \to i_0) Q^{(s-1)}(j_0 \to i_0)}{\sum P^{(s-1)}(j \to i) Q^{(s-1)}(j \to i)} \tag{6.17}$$

dont la *fonction de support* Q introduit dans la formule un facteur de pondération qui dépend de la confidence du voisinage. La valeur du dénominateur est un facteur de normalisation par files de la matrice des permutations.

Dans le cadre de notre approche nous avons introduit deux variantes pour rendre l'algorithme plus performant et accélérer la prise de décision. Nous allons,

– inclure la possibilité de faire une assignation ferme par seuillage à la fin de chaque itération du processus de relaxation, et

– recalculer la confidence seulement sur les appariements des nœuds (j, i) ayant une probabilité dans la matrice de permutations dans l'intervalle :

$$(1 - P_{min}) < P(j \to i) < P_{min} \qquad P_{min} > 0.5 \tag{6.18}$$

car en pratique ceux-ci sont les seuls appariements qui peuvent faire basculer la probabilité au-delà du seuil.

Tenant compte de ces modifications, la valeur de $P(j_0 \to i_0)$ est actualisée dans $P^{(s)}(j_0 \to i_0)$, en suivant les règles suivantes :

$$\begin{aligned} s = 0, \quad & P^{(0)}(j_0 \to i_0) = P(j_0 \to i_0) \\ s > 0, \quad & P^{(s)}(j_0 \to i_0) = P^{(s-1)}(j_0 \to i_0) Q^{(s-1)}(j_0 \to i_0) \end{aligned} \tag{6.19}$$

Notons que pour pouvoir appliquer le seuil nous travaillons sur une matrice de permutations qui n'est pas normalisée. Par ailleurs, nous avons imposé à la fonction de support Q de satisfaire certaines conditions. Ainsi nous voudrions,

1) $Q^{(s)}(j_0 \to i_0) = 1$,
si la cohérence du voisinage n'a pas changé au cours de la dernière itération, ou
si j_0 et i_0 n'ont pas de voisins en commun ;

2) $Q^{(s)}(j_0 \to i_0) > 1$,
si la cohérence du voisinage a augmenté dans la dernière itération ;

3) $Q^{(s)}(j_0 \to i_0) < 1$,
si la cohérence du voisinage a diminué.

Tenant compte de ceci nous avons défini la fonction de support Q comme suit,

$$Q^{(s)}(j_0 \to i_0) = \frac{1 + P^{(s)}(N_{j_0} \to N_{i_0})}{1 + P^{(s-1)}(N_{j_0} \to N_{i_0})} \tag{6.20}$$

en prenant

$$Q^{(0)}(j_0 \to i_0) = \frac{1 + P(N_{j_0} \to N_{i_0})}{1 + P(j_0 \to i_0)} \tag{6.21}$$

Grâce à la formule de relaxation, la cohérence concernant la topologie des deux graphes va propager sa certitude aux assignations douteuses. A chaque itération,

- si la valeur de $P(j_0 \rightarrow i)$ augmente pour un certain nœud i_0, il se peut que par seuillage cette assignation soit rendue ferme avant d'atteindre la stabilisation du processus.
- si le maximum de $P(j_0 \rightarrow i)$ diminue, tenant compte du fait que $P(j_0 \rightarrow d) = 1 - \max_{\forall i \in V_{B'}} P(j_0 \rightarrow i)$, il se peut que par seuillage l'assignation avec un nœud fictif soit rendue ferme.

Le processus finit lorsque tous les nœuds ont trouvé leur correspondant ou lorsque aucun changement n'a lieu. Les nœuds non assignés seront appariés à un nœud fictif.

6.4 Analyse des résultats

Dans cette section nous allons suivre en détail le processus de mise en correspondance avec des partitions obtenues à partir des images réelles. Pour cela, nous proposons deux exemples : celui de la Figure 6.13 (page 123), pour lequel les deux images font partie d'une même séquence ; et celui de la Figure 6.15 (page 128), où l'algorithme s'applique à deux images fortement différentes. Notons que dans les deux cas, les bords de l'image ont été traités comme des régions afin d'augmenter le nombre de points d'ancrage dans le processus de mise en correspondance.

Mise en correspondance de deux images très proches

Préalablement au calcul de la mise en correspondance, l'algorithme mène à terme la segmentation multiéchelle des images d'entrée afin d'extraire les partitions A et B et leurs graphes de voisinage. Sur la Figure 6.13 on peut observer comment, même si les images originales sont assez proches, les processus de segmentation respectifs nous ont menés à des résultats bien différents. Ainsi, l'objet assigné au nœud 13 du graphe A est occulté dans B. De même, à cause du mouvement de la caméra le nœud 9 dans A ne touche plus le cadre de l'image. Ce type de différences est inhérent aux changements produits au long d'une séquence. Néanmoins, il y a aussi des objets communs aux deux partitions qui ont été segmentés différemment. Ceci est le cas du nœud 5 dans A, qui ne peut être strictement mis en correspondance avec aucun des nœuds de B. Sur cet exemple il est facile de se rendre compte jusqu'à quel point de telles différences peuvent altérer les structures des graphes.

Par la suite, l'algorithme d'édition des partitions permet de minimiser la plupart des différences causées par la segmentation. Effectivement, on peut observer comment le nœud 5 dans A a donné lieu aux nœuds 9 et 14 dans A', qui se rapprochent du découpage de la scène proposé par B. De façon équivalente, le nœud 4 dans B a été aussi resegmenté à la recherche des contours de la chemise de A. Nous incitons le lecteur à chercher l'origine des différences sur les autres nœuds. Il devient alors évident que les partitions A' et B' sont beaucoup plus proches l'une de l'autre que ne l'étaient les originales.

Ensuite, la mise en correspondance de graphes est calculée parmi les graphes A' et B'. Pour cette étape, l'évolution des valeurs de la matrice de permutations est représentée sur les Tableaux 6.3 – 6.6, qui correspondent à chacune des étapes de calcul et seuillage de la matrice des permutations détaillées dans la Section 6.3.1. Par rapport à cette représentation, chaque ligne de la matrice garde les mesures de similarité d'un nœud j du graphe A' contre tous les nœuds i du graphe B', et à l'inverse pour les colonnes. Dans le coin supérieur gauche la matrice a été visualisée avec une palette de couleurs, allant du bleu clair pour les valeurs peu probables, jusqu'à la couleur violette pour les assignations fermes. Ainsi,

- Le Tableau 6.3 contient les valeurs initiales de la matrice de permutations calculées à partir de la ressemblance intrinsèque des attributs des régions. Les valeurs qui dépassent le seuil apparaissent en gras. Le fait que presque toutes ces valeurs se trouvent sur la diagonale de la matrice est fortuit et répond à l'ordre de labellisation des partitions.

- Le Tableau 6.4 contient la matrice de permutations après seuillage. De façon expérimentale, le seuil a été fixé à $P_{min} = 0.6$, le critère étant de séparer les appariements visuellement évidents, de ceux qui peuvent créer des doutes. La valeur de P_{min} détermine l'estricticité du processus de mise en correspondance, celui-ci étant le seul paramètre réglable du système. L'application du seuil peut être suivie en détail sur la Figure 6.12. Dans la plupart des cas il n'existe pas d'ambiguïté dans l'assignation, étant donné que le seuillage ne s'applique que si le maximum de probabilité coïncide pour $(j \rightarrow i)$ et $(i \rightarrow j)$. Afin de pouvoir appliquer le seuil, la distribution de probabilités n'a pas été normalisée. A ce point, il est intéressant de noter comment la plupart des nœuds peuvent être appariés sans avoir besoin de la confirmation de leur voisinage.

- Le Tableau 6.5 contient la matrice de permutations après calcul des probabilités de mise en correspondance d'un nœud réel avec un nœud fictif d. Rappeler que ces valeurs se calculent comme 1 moins la probabilité maximale de la ligne\colonne de la matrice. Ainsi, l'inclusion de nombreux zéros dans l'étape précédente peut rendre certaines assignations plus sûres.

- Le Tableau 6.6 contient les valeurs de la matrice de permutations après la dernière étape de seuillage. Ceci nous permet de rendre ferme l'apparition ou disparition de certains nœuds, or les différences parmi les caractéristiques des régions sont si importantes, que la ressemblance des voisinages ne pourrait en aucun cas les compenser(voir les nœuds j_3 et i_3).

A ce point, seule l'assignation des nœuds j_7, j_{11} et j_{12} du graphe A' et des nœuds i_7 et i_{12} du graphe B' reste douteuse. En attendant que l'information de voisinage puisse faire pencher la balance d'un côté ou de l'autre, l'algorithme de relaxation s'applique aux appariements ayant une probabilité parmi $(1 - 0.6) < P < 0.6$:

Itération 1

- $P^{(0)}(j_7 \rightarrow i_7) = 0.522$
  $P(N_{j_7} \rightarrow N_{i_7}) = 0.835$
  $P^{(1)}(j_7 \rightarrow i_7) = P^{(0)} \cdot Q^{(0)} = 0.522 \cdot 1.206 = 0.630 \quad \rightarrow \quad$ *assignation ferme*

- $P^{(0)}(j_{11} \rightarrow i_{12}) = 0.486$
  $P(N_{j_{11}} \rightarrow N_{i_{12}}) = 0.726$
  $P^{(1)}(j_{11} \rightarrow i_{12}) = P^{(0)} \cdot Q^{(0)} = 0.486 \cdot 1.161 = 0.565$

Pour ce cas-là, une seule itération a été nécessaire pour stabiliser les résultats. En rendant ferme l'assignation $(j_7 \rightarrow i_7)$, la probabilité $P(j_{12} \rightarrow d) = 0.89$ a aussi dépassé le seuil. Par la suite, aucune autre assignation pourra changer la $P(j_{11} \rightarrow i_{12})$, raison pour laquelle ces deux nœuds ne seront pas appariés. Nous avons résumé ci-dessous le résultat de la mise en correspondance établie parmi les nœuds des graphes A' et B' :

Projection des labels de A' vers B'

$j \in A'$	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	j_{17}
$i \in B'$	i_1	i_2	d	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	d	d	i_{11}	i_{13}	i_{15}	i_{14}	i_{16}

Projection des labels de B' vers A'

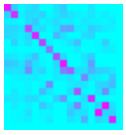
$i \in B'$	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}
$j \in A'$	j_1	j_2	d	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{13}	d	j_{14}	j_{16}	j_{15}	j_{17}

TAB. 6.2 – Appariements établis par la mise en correspondance.

Les projections résultantes peuvent être observées dans la Figure 6.13 (d1) et (d2). Ici, deux type de régions différentes ont été fusionnées :

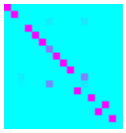
– régions créées par resegmentation qui n'ont pas trouvé de correspondante. Ceci est le cas de la région 12 dans B', qui a été obtenue par re-segmentation de la région 1 de la partition B. Ces sous-régions vont fusionner à nouveau avec la région de laquelle elles proviennent ;
– régions qui font partie d'un objet plus large dans la partition de départ. Ce cas-là est illustré par les nœuds 10 et 4 dans B'. Les deux ont trouvé leur correspondant dans A', cependant ils ne sont que des morceaux de la région 8 de A. Par conséquent, le processus d'édition va les regrouper sous un unique label dans la partition finale.

Pour rehausser les différences, sur les projections finales les nouveaux labels ont été marqués par une couleur verte.



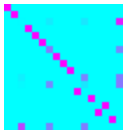
	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	d
j_1	**.92**	.20	.03	.11	.17	.00	.00	.12	.21	.00	.00	.01	.12	.00	.00	.00	–
j_2	.19	**.83**	.05	.15	.12	.17	.02	.10	.18	.11	.08	.05	.11	.04	.08	.03	–
j_3	.07	.05	.02	.14	.22	.04	.08	.15	.08	.10	.03	.10	.08	.05	.03	.09	–
j_4	.16	.16	.10	**.72**	.13	.14	.03	.11	.36	.38	.08	.08	.20	.15	.09	.06	–
j_5	.16	.12	.06	.16	**.88**	.00	.00	.29	.13	.00	.00	.13	.16	.00	.00	.00	–
j_6	.00	.19	.05	.14	.00	**.73**	.05	.00	.05	.12	.25	.03	.01	.04	.21	.04	–
j_7	.00	.02	.01	.02	.00	.04	.52	.00	.00	.02	.08	.00	.00	.02	.08	.07	–
j_8	.11	.10	.04	.12	.26	.01	.00	**.71**	.16	.00	.00	.11	.20	.00	.00	.00	–
j_9	.22	.19	.11	.32	.13	.04	.00	.16	**.91**	.00	.00	.09	.39	.00	.00	.00	–
j_{10}	.12	.11	.11	.44	.10	.18	.02	.10	.51	**.63**	.14	.12	.30	.19	.17	.10	–
j_{11}	.01	.05	.07	.07	.08	.02	.01	.09	.07	.12	.02	.49	.11	.06	.04	.22	–
j_{12}	.00	.03	.02	.06	.00	.05	.46	.00	.00	.07	.11	.11	.00	.16	.17	.27	–
j_{13}	.00	.09	.02	.07	.00	.27	.09	.00	.00	.12	**.68**	.04	.00	.12	.44	.07	–
j_{14}	.11	.11	.06	.15	.14	.03	.00	.19	.40	.00	.00	.14	**.84**	.00	.00	.00	–
j_{15}	.00	.08	.03	.07	.00	.23	.11	.00	.00	.13	.49	.07	.00	.17	**.94**	.12	–
j_{16}	.00	.03	.03	.14	.00	.04	.05	.00	.00	.21	.12	.13	.00	**.94**	.17	.17	–
j_{17}	.00	.02	.05	.07	.00	.04	.09	.00	.00	.13	.06	.33	.00	.17	.12	**.90**	–
d	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

TAB. 6.3 – Matrice des permutations : initialisation. En gras, les valeurs qui dépassent le seuil.



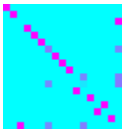
	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	d
j_1	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	–
j_2	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	–
j_3	0	0	.02	0	0	0	.08	0	0	0	0	.01	0	0	0	0	–
j_4	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	–
$j5$	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	–
j_6	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	–
j_7	0	0	.01	0	0	0	.52	0	0	0	0	0	0	0	0	0	–
j_8	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	–
j_9	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	–
j_{10}	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	–
j_{11}	0	0	.07	0	0	0	.01	0	0	0	0	.49	0	0	0	0	–
j_{12}	0	0	.02	0	0	0	.46	0	0	0	0	.11	0	0	0	0	–
j_{13}	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	–
j_{14}	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	–
j_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	–
j_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	–
j_{17}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	–
d	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

TAB. 6.4 – Matrice des permutations : première étape de seuillage. Les valeurs 0 et 1 dénotent les assignations fermes.



	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	d
j_1	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j_2	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j_3	0	0	.02	0	0	.08	0	0	0	0	.01	0	0	0	0	0	**.92**
j_4	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
$j5$	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
j_6	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
j_7	0	0	.01	0	0	0	.52	0	0	0	0	0	0	0	0	0	.48
j_8	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
j_9	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
j_{10}	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
j_{11}	0	0	.07	0	0	0	.01	0	0	0	0	.49	0	0	0	0	.51
j_{12}	0	0	.02	0	0	0	.46	0	0	0	0	.11	0	0	0	0	.54
j_{13}	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
j_{14}	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0
j_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0
j_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
j_{17}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
d	0	0	.93	0	0	0	.48	0	0	0	0	.51	0	0	0	0	–

TAB. 6.5 – Matrice des permutations : inclusion de nœuds fictifs.



	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	d
j_1	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j_2	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_4	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
$j5$	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
j_6	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
j_7	0	0	.01	0	0	0	.52	0	0	0	0	0	0	0	0	0	.48
j_8	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
j_9	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
j_{10}	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
j_{11}	0	0	.07	0	0	0	.01	0	0	0	0	.49	0	0	0	0	.51
j_{12}	0	0	.02	0	0	0	.46	0	0	0	0	.11	0	0	0	0	.54
j_{13}	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
j_{14}	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0
j_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0
j_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
j_{17}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
d	0	0	**1**	0	0	0	.48	0	0	0	0	.51	0	0	0	0	–

TAB. 6.6 – Matrice des permutations : deuxième étape de seuillage.

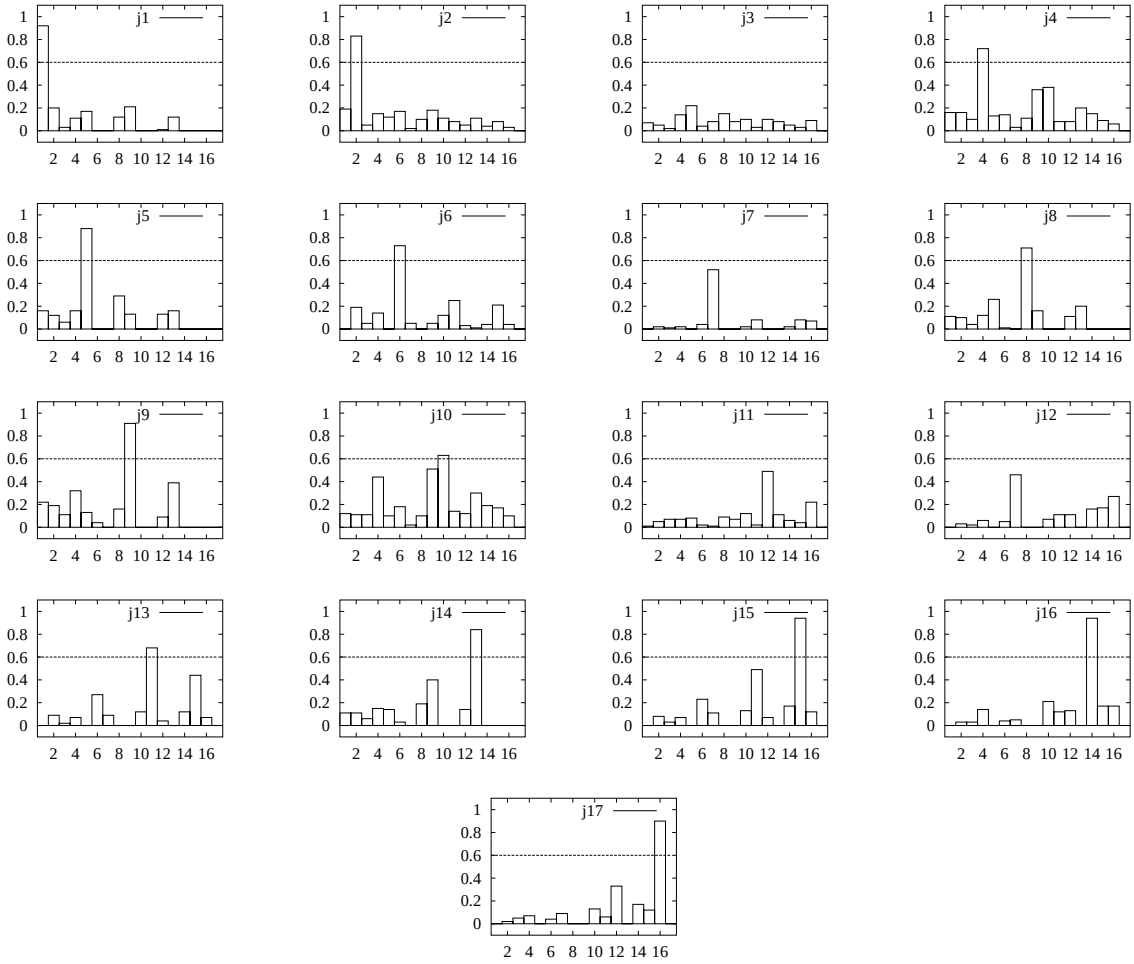
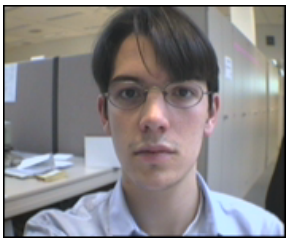
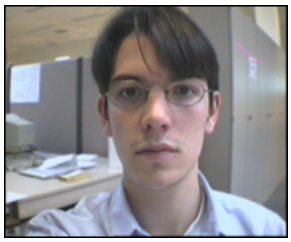


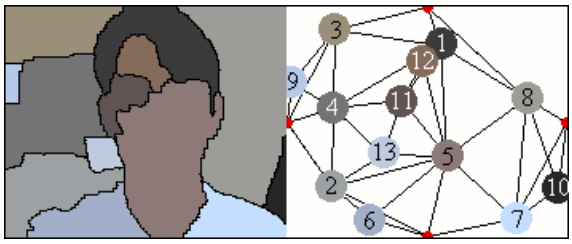
FIG. 6.12 – Première étape de seuillage de la matrice des permutations. Chacun des graphiques correspond à une des files de la matrice des permutations du Tableau 6.3.



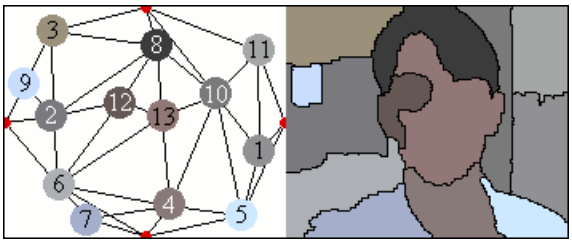
(a1) Image A



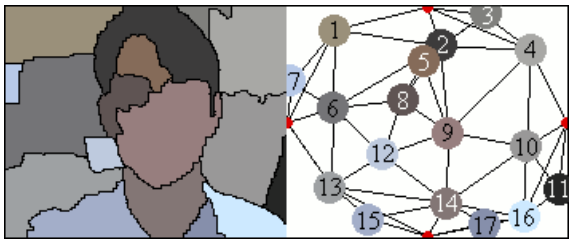
(a2) Image B



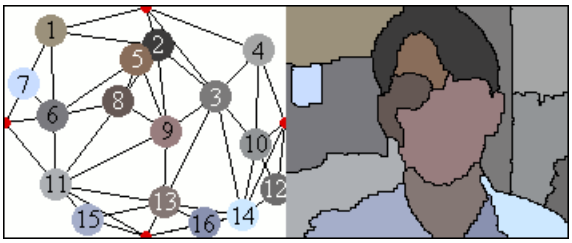
(b1) Partition A - Graphe A



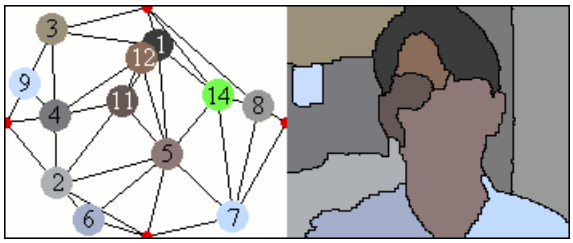
(b2) Graphe B - Partition B



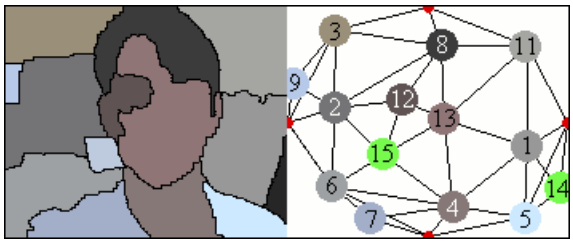
(c1) Partition A' - Graphe A'



(c2) Graphe B' - Partition B'



(d1) Projection de A sur B



(d2) Projection de B sur A

FIG. 6.13 – Résultat de la mise en correspondance de partitions sur deux images d'une même séquence. Il existe une cohérence parmi les labels de (b1)–(d1) et (b2)–(d2).

Mise en correspondance de deux images très différentes

Par opposition à l'exemple précédent, où le contenu des images était très proche, nous avons fourni dans la Figure 6.15 un deuxième exemple pour lequel les images à comparer différent fortement. L'intérêt ici est de voir comment se comporte l'algorithme lorsqu'un tel changement se produit.

Pour ces images, l'algorithme d'édition n'arrive pas à faire apparaître des objets communs, mais seulement à faire se rapprocher les deux découpages de la scène. Observer la resegmentation de la région 5 du graphe A, et des régions 2, 4, 9 et 10 du graphe B.

Etant donné les fortes différences parmi les graphes A' et B', les valeurs des probabilités chutent fortement dans le Tableau 6.7 par rapport à ceux obtenus lorsque les images se ressemblent. Cependant, ces assignations sont aussi indubitables lors de l'inclusion des nœuds fictifs dans le Tableau 6.10.

Suite au seuillage, seulement l'assignation des nœuds j_1, j_9, j_{15} et j_{16} du graphe A' et des nœuds i_2, i_5, i_{11} et i_{13} du graphe B' reste douteuse, car il existe effectivement une certaine ressemblance dans leur forme et position.

A l'égal de l'exemple précédent, une seule itération du processus de relaxation sera nécessaire pour atteindre la stabilité dans les assignations :

Itération 1

- $P^{(0)}(j_1 \to i_2) = 0.441$
  $P(N_{j_1} \to N_{i_2}) = 0.352$
  $P^{(1)}(j_1 \to i_2) = P^{(0)} \cdot Q^{(0)} = 0.441 \cdot 0.938 = 0.414$

- $P^{(0)}(j_9 \to i_5) = 0.590$
  $P(N_{j_9} \to N_{i_5}) = 0.042$
  $P^{(1)}(j_9 \to i_5) = P^{(0)} \cdot Q^{(0)} = 0.590 \cdot 0.655 = 0.387$
  $P^{(1)}(j_9 \to d) = 0.613 \quad \to \quad$ *assignation ferme*

- $P^{(0)}(j_{15} \to i_{11}) = 0.446$
  $P(N_{j_{15}} \to N_{i_{11}}) = 0.046$
  $P^{(1)}(j_{15} \to i_{11}) = P^{(0)} \cdot Q^{(0)} = 0.446 \cdot 0.723 = 0.323$
  $P^{(1)}(j_{15} \to d) = 0.677 \quad \to \quad$ *assignation ferme*

- $P^{(0)}(j_{16} \to i_{13}) = 0.506$
  $P(N_{j_{16}} \to N_{i_{13}}) = 0.327$
  $P^{(1)}(j_{16} \to i_{13}) = P^{(0)} \cdot Q^{(0)} = 0.506 \cdot 0.881 = 0.446$

Tenant compte des mesures de similarité établies, aucun voisinage n'a donné de soutien aux assignations, et par conséquent, les assignations vers des nœuds fictifs sont préférées.

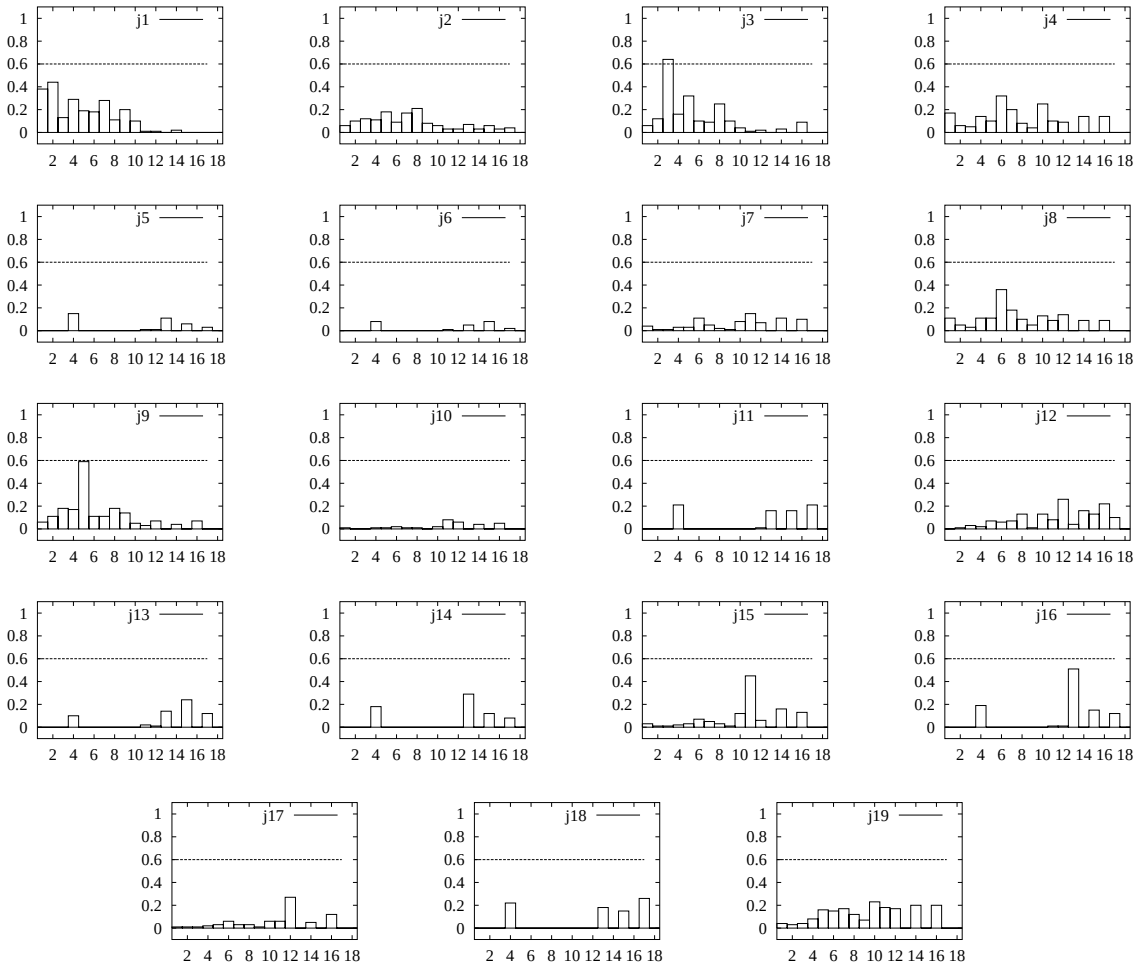
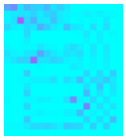
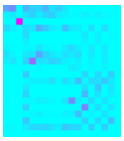


FIG. 6.14 – Première étape de seuillage de la matrice des permutations. Chacune des graphiques correspond à une des files de la matrice des permutations du Tableau 6.7.



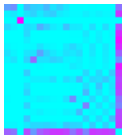
	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	i_{17}	d
j_1	.38	.44	.13	.29	.19	.18	.28	.11	.20	.10	.01	.01	.00	.02	.00	.00	.00	–
j_2	.06	.10	.12	.11	.18	.09	.17	.21	.08	.06	.03	.03	.07	.03	.06	.03	.04	–
j_3	.06	.12	**.64**	.16	.32	.10	.09	.25	.10	.04	.01	.02	.00	.03	.00	.09	.00	–
j_4	.17	.06	.05	.14	.10	.32	.20	.08	.04	.25	.10	.09	.00	.14	.00	.14	.00	–
j_5	.00	.00	.00	.15	.00	.00	.00	.00	.00	.00	.01	.01	.11	.00	.06	.00	.03	–
j_6	.00	.00	.00	.08	.00	.00	.00	.00	.00	.00	.01	.00	.05	.00	.08	.00	.02	–
j_7	.04	.01	.01	.03	.03	.11	.05	.02	.01	.08	.15	.07	.00	.11	.00	.10	.00	–
j_8	.11	.05	.03	.11	.11	.36	.18	.10	.05	.13	.09	.14	.00	.09	.00	.09	.00	–
j_9	.06	.11	.18	.17	.59	.11	.11	.18	.14	.05	.03	.07	.00	.04	.00	.07	.00	–
j_{10}	.01	.00	.00	.01	.01	.02	.01	.01	.00	.02	.08	.06	.00	.04	.00	.05	.00	–
j_{11}	.00	.00	.00	.21	.00	.00	.00	.00	.00	.00	.00	.01	.16	.00	.16	.00	.21	–
j_{12}	.00	.01	.03	.02	.07	.06	.07	.13	.01	.13	.08	.26	.04	.16	.13	.22	.10	–
j_{13}	.00	.00	.00	.10	.00	.00	.00	.00	.00	.00	.02	.01	.14	.00	.24	.00	.12	–
j_{14}	.00	.00	.00	.18	.00	.00	.00	.00	.00	.00	.00	.00	.29	.00	.12	.00	.08	–
j_{15}	.03	.01	.01	.02	.03	.07	.05	.03	.01	.12	.45	.06	.00	.16	.00	.13	.00	–
j_{16}	.00	.00	.00	.19	.00	.00	.00	.00	.00	.00	.01	.01	.51	.00	.15	.00	.12	–
j_{17}	.01	.01	.01	.02	.03	.06	.03	.03	.01	.06	.06	.27	.00	.05	.00	.12	.00	–
j_{18}	.00	.00	.00	.22	.00	.00	.00	.00	.00	.00	.00	.00	.18	.00	.15	.00	.26	–
j_{19}	.04	.03	.04	.08	.16	.15	.17	.12	.07	.23	.18	.17	.00	.20	.00	.20	.00	–
d	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

TAB. 6.7 – Matrice des permutations : initialisation. En gras, les valeurs qui dépassent le seuil.



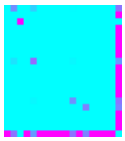
	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	i_{17}	d
j_1	.38	.44	0	.29	.19	.18	.28	.11	.20	.10	.01	.01	.00	.02	.00	.00	.00	–
j_2	.06	.10	0	.11	.18	.09	.17	.21	.08	.06	.03	.03	.07	.03	.06	.03	.04	–
j_3	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	–
j_4	.17	.06	0	.14	.10	.32	.20	.08	.04	.25	.10	.09	.00	.14	.00	.14	.00	–
j_5	.00	.00	0	.15	.00	.00	.00	.00	.00	.00	.01	.01	.11	.00	.06	.00	.03	–
j_6	.00	.00	0	.08	.00	.00	.00	.00	.00	.00	.01	.00	.05	.00	.08	.00	.02	–
j_7	.04	.01	0	.03	.03	.11	.05	.02	.01	.08	.15	.07	.00	.11	.00	.10	.00	–
j_8	.11	.05	0	.11	.11	.36	.18	.10	.05	.13	.09	.14	.00	.09	.00	.09	.00	–
j_9	.06	.11	0	.17	.59	.11	.11	.18	.14	.05	.03	.07	.00	.04	.00	.07	.00	–
j_{10}	.01	.00	0	.01	.01	.02	.01	.01	.00	.02	.08	.06	.00	.04	.00	.05	.00	–
j_{11}	.00	.00	0	.21	.00	.00	.00	.00	.00	.00	.00	.01	.16	.00	.16	.00	.21	–
j_{12}	.00	.01	0	.02	.07	.06	.07	.13	.01	.13	.08	.26	.04	.16	.13	.22	.10	–
j_{13}	.00	.00	0	.10	.00	.00	.00	.00	.00	.00	.02	.01	.14	.00	.24	.00	.12	–
j_{14}	.00	.00	0	.18	.00	.00	.00	.00	.00	.00	.00	.00	.29	.00	.12	.00	.08	–
j_{15}	.03	.01	0	.02	.03	.07	.05	.03	.01	.12	.45	.06	.00	.16	.00	.13	.00	–
j_{16}	.00	.00	0	.19	.00	.00	.00	.00	.00	.00	.01	.01	.51	.00	.15	.00	.12	–
j_{17}	.01	.01	0	.02	.03	.06	.03	.03	.01	.06	.06	.27	.00	.05	.00	.12	.00	–
j_{18}	.00	.00	0	.22	.00	.00	.00	.00	.00	.00	.00	.00	.18	.00	.15	.00	.26	–
j_{19}	.04	.03	0	.08	.16	.15	.17	.12	.07	.23	.18	.17	.00	.20	.00	.20	.00	–
d	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

TAB. 6.8 – Matrice des permutations : première étape de seuillage. Les valeurs 0 et 1 dénotent les assignations fermes.



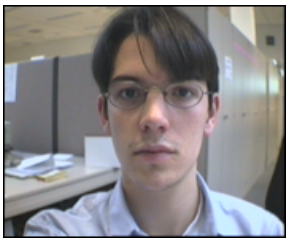
	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	i_{17}	d
j_1	.38	.44	0	.29	.19	.18	.28	.11	.20	.10	.01	.01	.00	.02	.00	.00	.00	.56
j_2	.06	.10	0	.11	.18	.09	.17	.21	.08	.06	.03	.03	.07	.03	.06	.03	.04	**.79**
j_3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j_4	.17	.06	0	.14	.10	.32	.20	.08	.04	.25	.10	.09	.00	.14	.00	.14	.00	**.68**
j_5	.00	.00	0	.15	.00	.00	.00	.00	.00	.00	.01	.01	.11	.00	.06	.00	.03	**.85**
j_6	.00	.00	0	.08	.00	.00	.00	.00	.00	.00	.01	.00	.05	.00	.08	.00	.02	**.92**
j_7	.04	.01	0	.03	.03	.11	.05	.02	.01	.08	.15	.07	.00	.11	.00	.10	.00	**.85**
j_8	.11	.05	0	.11	.11	.36	.18	.10	.05	.13	.09	.14	.00	.09	.00	.09	.00	**.64**
j_9	.06	.11	0	.17	.59	.11	.11	.18	.14	.05	.03	.07	.00	.04	.00	.07	.00	.41
j_{10}	.01	.00	0	.01	.01	.02	.01	.01	.00	.02	.08	.06	.00	.04	.00	.05	.00	**.92**
j_{11}	.00	.00	0	.21	.00	.00	.00	.00	.00	.00	.00	.01	.16	.00	.16	.00	.21	**.79**
j_{12}	.00	.01	0	.02	.07	.06	.07	.13	.01	.13	.08	.26	.04	.16	.13	.22	.10	**.74**
j_{13}	.00	.00	0	.10	.00	.00	.00	.00	.00	.00	.02	.01	.14	.00	.24	.00	.12	**.76**
j_{14}	.00	.00	0	.18	.00	.00	.00	.00	.00	.00	.00	.00	.29	.00	.12	.00	.08	**.71**
j_{15}	.03	.01	0	.02	.03	.07	.05	.03	.01	.12	.45	.06	.00	.16	.00	.13	.00	.55
j_{16}	.00	.00	0	.19	.00	.00	.00	.00	.00	.00	.01	.01	.51	.00	.15	.00	.12	.49
j_{17}	.01	.01	0	.02	.03	.06	.03	.03	.01	.06	.06	.27	.00	.05	.00	.12	.00	**.73**
j_{18}	.00	.00	0	.22	.00	.00	.00	.00	.00	.00	.00	.00	.18	.00	.15	.00	.26	**.74**
j_{19}	.04	.03	0	.08	.16	.15	.17	.12	.07	.23	.18	.17	.00	.20	.00	.20	.00	**.77**
d	**.62**	.56	0	**.71**	.41	.64	**.72**	**.79**	**.80**	**.75**	.55	**.73**	.49	**.80**	**.76**	**.78**	**.74**	–

TAB. 6.9 – Matrice des permutations : inclusion des nœuds fictifs.



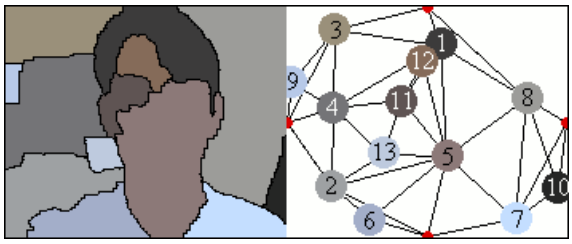
	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	i_{17}	d
j_1	.00	.44	0	.00	.19	.00	.00	.00	.00	.00	.01	.00	.00	.00	.00	.00	.00	.56
j_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_3	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_9	.00	.11	0	.00	.59	.00	.00	.00	.00	.00	.03	.00	.00	.00	.00	.00	.00	.41
j_{10}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_{11}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_{15}	.00	.01	0	.00	.03	.00	.00	.00	.00	.00	.45	.00	.00	.00	.00	.00	.00	.55
j_{16}	.00	.00	0	.00	.00	.00	.00	.00	.00	.00	.01	.00	.51	.00	.00	.00	.00	.49
j_{17}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_{18}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
j_{19}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
d 1	.56	.00	**1**	.41	**1**	**1**	**1**	**1**	**1**	.55	**1**	.49	**1**	**1**	**1**	**1**	**1**	–

TAB. 6.10 – Matrice des permutations : deuxième étape de seuillage.

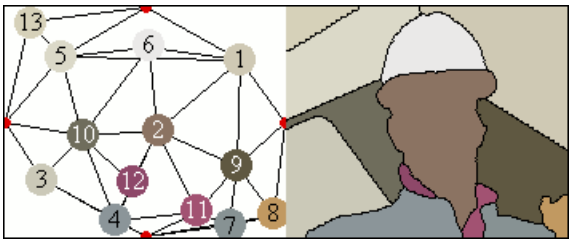


(a1) Image A

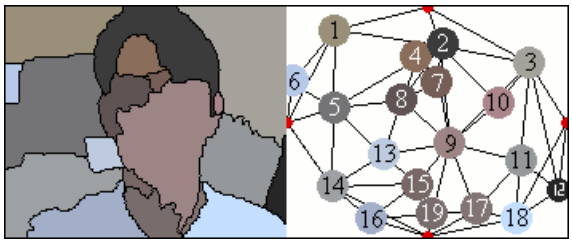
(a2) Image B



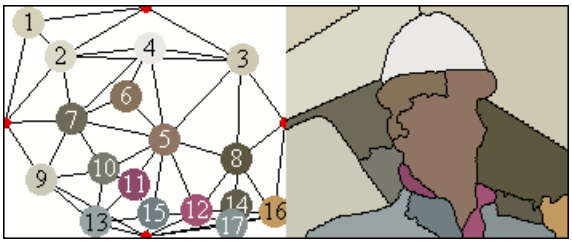
(b1) Partition A - Graphe A



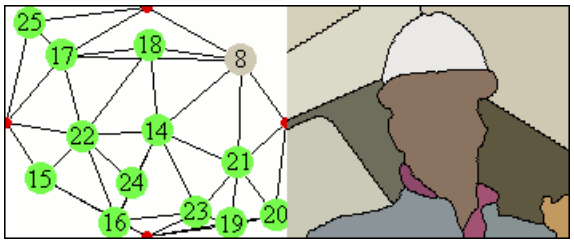
(b2) Graphe B - Partition B



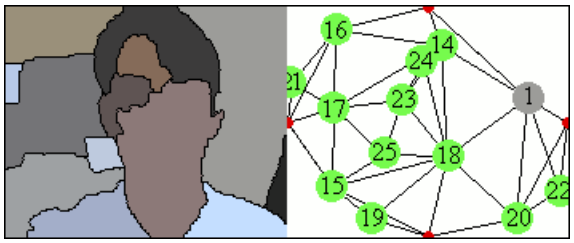
(c1) Partition A' - Graphe A'



(c2) Graphe B' - Partition B'



(d1) Projection de A sur B



(d2) Projection de B sur A

FIG. 6.15 – Résultat de la mise en correspondance de partitions lorsque le contenu des images diffère. Il existe une cohérence parmi les labels de (b1)–(d1) et (b2)–(d2).

Dans le résultat final, une seule région a cru être identifiée, tout le reste est correctement labellisé comme étant de nouveaux objets. Les assignations finales restent donc comme suit,

Projection des labels de A' vers B'

$j \in A'$	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}	j_{12}	j_{13}	j_{14}	j_{15}	j_{16}	j_{17}	j_{18}	j_{19}
$i \in B'$	d	d	i_3	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d

Projection des labels de B' vers A'

$i \in B'$	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	i_{17}
$j \in A'$	d	d	j_3	d	d	d	d	d	d	d	d	d	d	d	d	d	d

TAB. 6.11 – Appariements établis par la mise en correspondance.

6.5 Conclusions

Dans ce chapitre nous avons traité le problème de la mise en correspondance de partitions sous une nouvelle approche qui combine des algorithmes de mise en correspondance de graphes avec les techniques de segmentation multiéchelle présentées dans le chapitre précédent. C'est la technique de Segmentation et Appariement Conjoint (SAC). Conscients des inconvénients classiquement attribués aux algorithmes de mise en correspondance de graphes, mais motivés par le large potentiel qu'une telle représentation de la scène peut offrir,nous avons affronté le problème dans le but de :

- faire se ressembler les partitions au maximum en dehors du processus de mise en correspondance de graphes ;
- accélérer la prise de décision lors d'assignations fortement probables.

Ainsi, l'inclusion de deux étapes d'édition profitant de la décomposition hiérarchique des partitions a donné lieu à une technique de mise en correspondance très robuste, capable de surmonter quelques-uns des inconvénients les plus contraignants concernant l'appariement de partitions. Nous avons présenté cette technique comme étant générique car,

- du point de vue des techniques de *mise en correspondance de graphes*, le schéma de traitement proposant d'éditer les graphes de façon préalable à leur appariement peut être adopté par d'autres techniques et étendu à d'autres domaines d'application ;
- du point de vue des techniques de *mise en correspondance de partitions*, l'approche que nous avons développé peut être appliquée avec de faibles variations au problème du suivi d'un objet mais aussi au problème de la recherche d'un objet dans une base de données.

Bien que le sujet principal de cette thèse concerne la mise en correspondance de partitions en vue du suivi d'objets, nos algorithmes ne seront particularisés à ce type d'applications que dans le chapitre suivant. Par ailleurs, comme nous l'avons déjà avancé, l'extension de la méthode au domaine de la recherche d'objets dans des bases de données serait aussi envisageable, ceci constitue une des pistes ouvertes pour une poursuite éventuelle de ce travail.

Chapitre 7

Mise en Correspondance d'une Séquence de Partitions

Un long chemin a été parcouru pour aller du traitement des images fixes jusqu'à rendre possible l'analyse de séquences. Nous avons bâti des algorithmes pour construire la partition d'une seule image, ensuite pour apparier deux partitions, pour enfin aborder le problème de la mise en correspondance de toute une séquence de partitions.

Nous rappelons d'abord deux approches classiques : celles qui traitent la séquence comme un volume 3D spatio-temporel, et celles qui font appel à l'analyse de mouvement pour donner de la cohérence temporelle à la segmentation 2D. Bien que ces deux approches se soient montrées très performantes dans des nombreuses applications, elles n'ont pas résolu quelques-uns des principaux problèmes causés par la présence de mouvement.

En prenant ceux-ci comme un défi, nous allons proposer une nouvelle approche basée sur la méthode de mise en correspondance de partitions présentée dans le chapitre précédent. La principale innovation concerne l'utilisation du graphe en guise de mémoire temporelle, nous permettant de reconnaître un objet s'il réapparaît après occultation. Ceci ouvre tout un éventail de nouvelles applications par lesquelles nous conclurons cette deuxième partie du document.

7.1 Les séquences vidéo

Une séquence vidéo peut être définie comme une succession d'images constituant un échantillonnage spatial et temporel d'une scène qui se déroule dans le monde réel. Chacune de ces images reçoit le nom de *trame*.

Nous avons déjà présenté le processus d'acquisition des images numériques dans le Chapitre 2, lors de l'introduction à la première partie de cette thèse. Ici, lors du passage des images fixes aux séquences, un nouveau facteur entre en jeu : il s'agit de la *cadence d'enregistrement*.

Comme son nom l'indique, la cadence d'enregistrement fixe le nombre d'images acquises

par seconde. Afin de minimiser la mémoire requise pour stocker une séquence, ainsi que la bande passante nécessaire pour la transmettre, on fait en sorte qu'elle soit la plus faible possible. Celle-ci a été établie expérimentalement de façon à s'assurer que lors de la visualisation de la séquence, le mouvement et la luminance soient aperçus en continuité.

Continuité de mouvement : on a observé que les mouvements rapides ont besoin d'un minimum de 24 img/sec pour être aperçus en continuité, tandis que les mouvements lents peuvent se représenter à partir de 10 img/sec. Ainsi, on a fixé la cadence d'acquisition à 12-15 img/sec pour les bandes dessinées, allant jusqu'à 24-30 img/sec pour le cinéma et la TV [97].

Continuité de luminance : lors d'expériences on s'est rendu compte qu'en représentant la succession d'images à une fréquence trop faible, l'œil humain détecte un clignotement de la luminance lié à la fréquence de radiation de la source lumineuse. De même, la cadence d'échantillonnage exigée ici est beaucoup plus élevée que celle imposée par la continuité de mouvement. Face à cette contrainte, deux stratégies différentes permettent de doubler le nombre d'images montrées à l'écran sans augmenter pourtant la cadence d'enregistrement : on peut montrer deux fois la même image (cinéma), ou émettre les images entrelacées de façon à créer deux trames à partir d'une seule image enregistrée (TV).

La caractéristique la plus importante qui entre ici en scène est, sans doute, le *mouvement*. L'information liée au mouvement a des usages très différents en fonction de l'application :

- les systèmes de codage font de l'estimation de mouvement un outil qui permet d'extraire de l'information redondante parmi deux trames consécutives de la séquence.

- l'analyse de séquences s'appuie sur l'estimation de mouvement pour traiter les différences parmi les trames, car les deux éléments se trouvent étroitement imbriqués.

Nous allons nous focaliser sur ce dernier point, celui-ci étant d'un intérêt particulier vis-à-vis des applications de suivi d'objets. Plus en détail, les différences causées par le mouvement entre deux trames successives d'une séquence répondent au :

Déplacement : un objet qui se déplace provoque un décalage dans la position des régions sur deux images consécutives, tout en décrivant une trajectoire dans l'axe temporel de la séquence. L'importance de ce décalage est intimement liée à la vitesse et à la taille de l'objet. Les déplacements peuvent changer les conditions de voisinage d'une région dans le plan de l'image, et même provoquer une perte de connexité dans l'axe temporel.

Apparition\disparition : nous dirons qu'un nouvel objet apparaît quand il entre pour la première fois dans la scène ; nous dirons qu'un objet a disparu s'il est sorti du cadre de

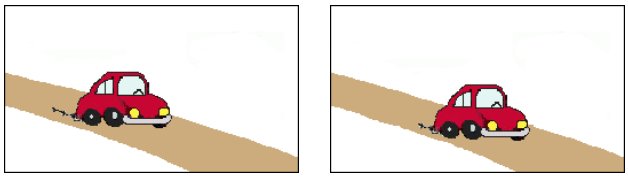


FIG. 7.1 – Déplacements.

l'image. Soit à cause du mouvement de la caméra, soit à cause des déplacements locaux, l'apparition\disparition d'objets est un événement courant dans les séquences vidéo. Et pourtant, ceci pose de nombreux problèmes à l'estimation du mouvement, car il n'y a pas de correspondance pour les zones dans l'arrière plan qui viennent d'être découvertes ou pour l'objet qui vient d'apparaître.

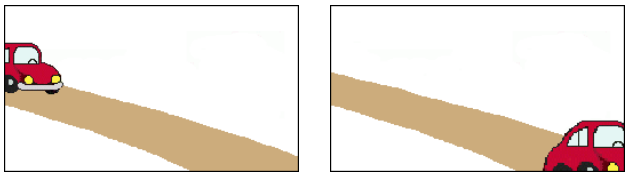


FIG. 7.2 – Apparition\disparition.

Occultation partielles\totales : nous dirons qu'un objet est partiellement occulté s'il est en partie visible derrière un autre objet présent dans la scène ; nous dirons que l'occultation est totale si l'objet n'est plus visible de façon passagère. Ces deux événements vont fortement troubler les algorithmes de mise en correspondance : un objet à moitié couvert ou à moitié découvert ne présente pas les mêmes attributs, donc il peut être facilement confondu avec un autre ; un objet qui disparaît temporellement ne peut pas être suivi sans mettre en jeu la notion de mémoire.

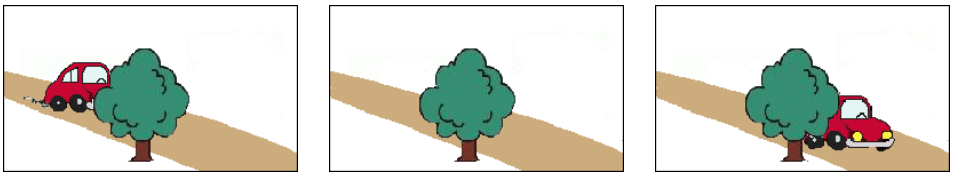


FIG. 7.3 – Occultations partielles\totales.

Jusqu'ici nous avons utilisé la notion de mouvement pour expliquer les différences parmi des trames consécutives d'une séquence. Quelques exemples nous ont servi pour présenter la problématique du suivi d'objets. De nombreuses applications sont concernées,

- surveillance d'activités humaines,
- surveillance du trafic routier,
- analyse de gestes,

– analyse de scènes pour la vision par ordinateur,
– stéréovision, ...

Par la suite, nous verrons comment les techniques de segmentation se sont adaptées au traitement des séquences d'images, où l'objectif est de créer une séquence de partitions tout en préservant la cohérence temporelle des labels.

Par rapport aux approches les plus classiques introduites dans la Section 7.2, qui nous serviront de point de repère, nous présenterons dans la Section 7.3 une nouvelle méthode obtenue par extension de la technique de mise en correspondance de partitions exposée au cours du chapitre précédent.

7.2 Approches classiques concernant la segmentation de séquences

Dans cette section nous allons aborder la problématique de la segmentation de séquences par l'intermédiaire de deux techniques fortement répandues dans ce domaine : celles qui font une segmentation 3D, en considérant la séquence comme un volume spatio-temporel ; et celles qui incorporent de l'information de mouvement à la segmentation 2D pour donner de la cohérence temporelle. La connaissance de leurs atouts et leurs limites nous donnera ensuite un point de repère pour mieux évaluer la performance de nos algorithmes.

7.2.1 Segmentation 3D

Ces techniques ont vu le jour dans le domaine biomédical pour traiter des données 3D [27], mais rapidement elles ont été étendues au traitement de séquences [73].

La procédure de calcul correspond à celle de la segmentation 2D, en tenant compte cette fois-ci d'une connexité 3D. C'est-à-dire, en considérant des voisinages volumiques au lieu des voisinages planaires, englobant des pixels de plusieurs images.

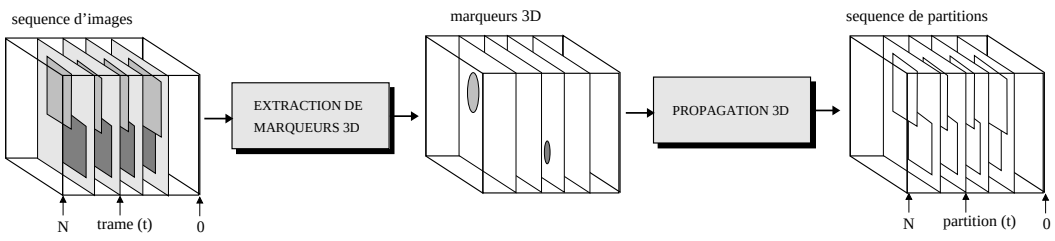


FIG. 7.4 – Schéma de traitement d'une séquence par segmentation 3D.

Une telle façon de procéder résout la plupart des problèmes d'instabilité liés au calcul isolé des partitions, car en segmentant toutes les trames au même temps la cohérence temporelle se voit renforcée. De plus, aucun algorithme de mise en correspondance proprement dit n'est nécessaire, étant donné que toute la séquence de partitions est créée à partir d'un même

ensemble de marqueurs. Néanmoins, d'autres inconvénients apparaissent qui vont limiter la performance de cette technique :

Changement de labels : la propagation correcte d'un label pendant la segmentation va dépendre de l'existence d'une connexité 3D parmi les régions qui représentent un même objet au cours des trames. C'est-à-dire, seul le recouvrement temporel des régions assure un chemin de propagation aux marqueurs dans le volume. Ainsi, les objets trop rapides ou trop petits risquent fortement de changer de label au long de la séquence.

Inclusion d'un retard temporel : par leur nature tridimensionnelle, ces algorithmes sont non-causals. Ceci signifie que le résultat de la segmentation au temps (t) dépend aussi bien de l'ensemble des trames qui la précédent, allant de 0 à $(t-1)$, que de celles qui la suivent dans l'ordre de la séquence, allant de $(t+1)$ jusqu'à N. Par ailleurs, la mémoire requise est très importante, car toute de la séquence est traitée en même temps.

Pour adapter la connexité 3D au mouvement, certaines approches ont inclus une compensation de mouvement précédant la propagation des labels par inondation. Face aux besoins de mémoire, des approximations basées sur des *fenêtres glissantes* ont été proposées. Il s'agit de traiter la séquence par de petits blocs décalés sur l'axe temporel. Finalement, parmi les extensions plus récentes dans le domaine de la segmentation 3D, nous pouvons mentionner les travaux de Zanoguera dans [109], concernant la création de hiérarchies de partitions volumiques.

7.2.2 Segmentation 2D + Analyse de mouvement

Afin de pouvoir appliquer les techniques de segmentation 2D au traitement de séquences, on a recours classiquement à l'analyse de mouvement pour donner de la cohérence temporelle à l'ensemble des partitions.

Dans la plupart des cas, ces approches procèdent de façon récursive en projetant l'information de la partition résultante à $(t-1)$ sur l'image à (t) (voir Figure 7.5). Etant donné que les objets ont pu se déplacer pendant cet intervalle de temps, une compensation de mouvement est nécessaire. Ainsi, les vecteurs de mouvement vont marquer la direction de la projection de la partition à $(t-1)$, qui sert à initialiser la segmentation 2D de la trame à (t). Ceci permet au processus de segmentation d'assurer une continuité dans la labellisation.

La récursivité de ces algorithmes permet de minimiser les besoins de mémoire. De plus, aucun retard n'est introduit dans la chaîne de traitement, car l'ensemble des trames est traité par paires.

Du fait de ces bonnes qualités, ces techniques se sont appliquées avec succès à de nombreuses reprises au suivi d'objets. Nous pouvons mentionner les travaux de Pardas [74], ainsi que ceux de Marcotegui [50], pour des applications de suivi d'objets en vue du codage. Et pourtant quelques limitations sont à prendre en compte :

Influence des erreurs de mouvement : la robustesse de la segmentation va dépendre fortement de la qualité de l'estimation de mouvement, car les vecteurs erronés peuvent causer des anomalies dans l'étape de projection. Ainsi, il se peut qu'une région de la partition soit fragmentée suite à la compensation de mouvement, ou qu'elle soit en conflit avec une autre région projetée sur les mêmes coordonnées.

Changement de labels : la disparition d'un objet, temporaire ou non, entraîne la perte définitive de son label. La récursivité de ces algorithmes n'assure la cohérence temporale de labellisation que parmi des trames consécutives. Ce problème est équivalent à celui de la dépendance par rapport à la connexité temporelle dans les algorithmes de segmentation 3D.

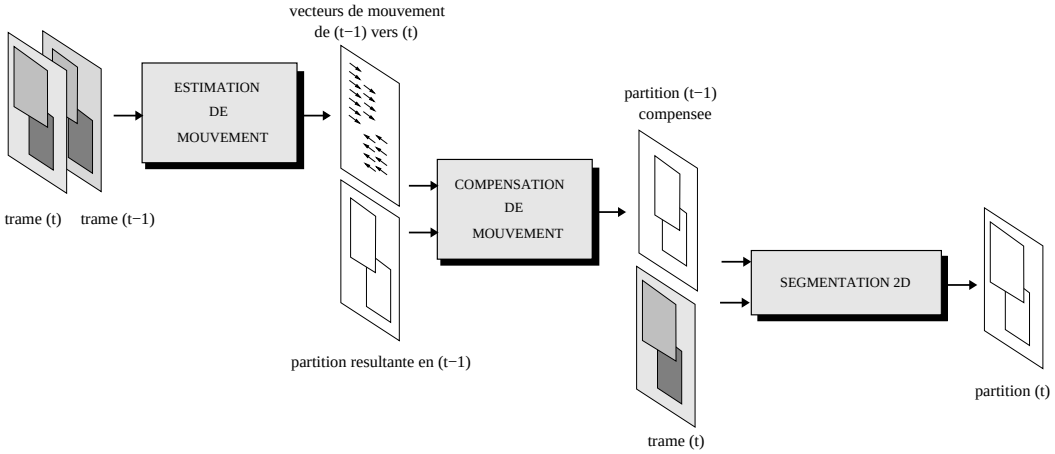


FIG. 7.5 – Schéma de traitement récursif par analyse de mouvement.

Afin d'augmenter la robustesse des algorithmes face aux vecteurs de mouvement erronés, certaines approches, comme celle de Marqués et Llach [55], incluent une étape de gestion de conflits suite à la compensation de la partition. La perte de labels n'est dans aucun cas résolue, et reste un défi pour ce qui concerne les applications de suivi d'objets.

7.3 Approche nouvelle basée sur la mise en correspondance de graphes

Nous allons présenter ici une nouvelle approche concernant la création d'une séquence de partitions. Le corps de l'algorithme est une extension de la technique de mise en correspondance présentée au cours du chapitre précédent, considérant que dorénavant toutes les paires d'images sous analyse vont correspondre à des trames consécutives d'une même séquence vidéo.

L'objectif final de l'algorithme est de créer une séquence de partitions de façon à ce que, sur chaque trame, un même objet soit également segmenté et également labellisé.

Etant donné que l'algorithme procède de façon récursive, une étape d'initialisation est nécessaire. Nous allons donc segmenter la première image de la séquence au moyen d'un algorithme purement 2D, dans le but de créer une *partition de référence* sur laquelle tous les objets d'intérêt soient présents. Cette première segmentation peut être fournie par l'utilisateur, en définissant le découpage de la scène qui s'adapte le mieux à ses intérêts, ou choisie automatiquement par le système lorsqu'on fixe le nombre de régions de la partition.

L'algorithme procède ensuite de façon récursive en projetant la partition résultante à $(t-1)$ vers celle obtenue à (t), et ainsi successivement jusqu'à la fin de la séquence. La Figure 7.6 illustre la récursivité de ce schéma de traitement, les trois étapes principales étant détaillées dans les sections qui suivent.

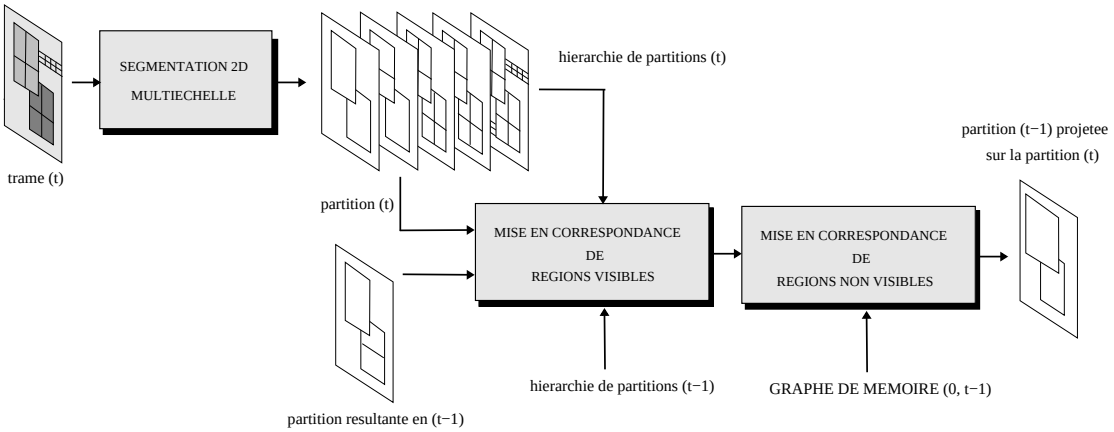


FIG. 7.6 – Création d'une séquence de partitions par mise en correspondance de graphes.

7.3.1 Segmentation 2D multiéchelle

Le système procède en premier à la segmentation de la trame au temps (t) en suivant l'approche multiéchelle présentée dans le Chapitre 4. Rappelons brièvement ici les différentes étapes sur lesquelles repose ce processus :

1.- *Etape de pré-traitement :* l'enchaînement d'un étirement de contraste plus une étape de filtrage par nivellement a pour but de rendre une image plus lisse sur les zones peu texturées et plus abrupte sur les contours des objets. L'intensité du filtrage peut être définie comme paramètre ajustable afin de mieux s'adapter aux conditions de la séquence.

2.- *Calcul de l'image gradient :* ensuite le calcul du gradient transforme cette image simplifiée en une image de contours qui servira de support de calcul au processus d'inondation.

3.- *Processus d'inondation :* en associant un label a chaque minimum de l'image gradient, l'inondation multiéchelle cherche à construire un historique des absorptions permettant

de classer l'ensemble des régions dans une hiérarchie.

Le résultat de la segmentation multiéchelle s'exprime sous la forme d'une hiérarchie de partitions représentant le contenu de l'image à différents niveaux de finesse. Nous verrons dans la section suivante comment cette hiérarchie de partitions fournira un support aux processus d'édition lors du calcul de la mise en correspondance.

Mais par ailleurs, la hiérarchie de partitions devra aussi fournir la partition au temps (t) qui sera appariée avec la partition résultante à $(t-1)$. Sans avoir aucune connaissance à priori des changements qui peuvent avoir eu lieu pendant cet intervalle de temps, le choix de telle ou telle autre partition à l'intérieur de la hiérarchie reste très ouvert. A grands traits, deux critères sont possibles :

Nombre de régions constant : il s'agit d'établir au début de la séquence la finesse à laquelle l'algorithme va travailler. Cette valeur conditionne la sensibilité du système face aux objets qui pourront être détectés.

Nombre de régions variable selon la complexité de la scène : il est également possible de contrôler la finesse de la partition de façon à calculer pour chaque trame de la séquence le nombre de régions en fonction de la complexité de la scène à segmenter.

Au cours de nos expériences nous avons commencé en fixant le nombre de régions de la partition au début du traitement de la séquence. Ce critère étant celui qui entraîne le moins de calculs, il s'est montré valable pour la plupart des séquences ayant une complexité régulière.

Nous avons aussi étudié la possibilité de rendre le nombre de régions variable en fonction de l'évolution de la complexité de la scène. Un critère simple, qui n'entraîne aucun calcul additionnel, consiste à rendre le nombre de régions de la partition au temps (t) égal au nombre de régions présentes dans la partition à $(t-1)$.

Ceci devrait permettre à l'algorithme d'augmenter le nombre de régions à chaque fois qu'un nouvel objet est détecté, mais aussi de diminuer le nombre de régions pour chaque objet qui disparaît. Néanmoins, sur les résultats obtenus on a observé comment, dans certains cas, la finesse de la séquence de partitions commençait à augmenter ou à diminuer sans arrêt.

En effet, il suffit qu'une région erronée apparaisse sur la partition résultante à (t) pour qu'une nouvelle région soit demandée lors de la segmentation à $(t+1)$. Sans que la complexité de la scène n'ait augmentée, cette nouvelle région a de très fortes probabilités de ne pas être appariée. Par conséquent, une nouvelle région sera encore demandée à $(t+2)$, et ainsi successivement. L'effet contraire peut aussi se produire dès qu'une région disparaît par erreur.

De cette première approche pour adapter la sensibilité du système à l'évolution de la complexité de la séquence, nous en conclurons que toute variation dans le nombre de régions peut rendre le système de mise en correspondance instable si elle n'est pas fortement maîtrisée. D'autres pistes restent à explorer.

7.3.2 Mise en correspondance des régions visibles

Au cœur du système, un algorithme de mise en correspondance de partitions a en charge d'établir l'appariement entre les régions de la partition à $(t-1)$ et celles qui viennent d'être segmentées en (t). Ces régions constituent ce que nous avons nommé l'ensemble des *régions visibles*, par opposition à l'ensemble des *régions occultées*, qui groupe l'ensemble des régions qui ont apparu dans l'intervalle $[0, t-1)$ mais qui ne sont pas visibles au temps (t).

L'algorithme que nous avons implémenté ici est une application directe de la technique générale de mise en correspondance de partitions présentée dans le Chapitre 6. Trois étapes se succèdent :

1.- *Edition des partitions par resegmentation des régions :* cette première étape a pour but de faire se rassembler au maximum le contenu des partitions présentes à l'entrée. A l'aide des hiérarchies de partitions obtenues auparavant, nous allons faire apparaître tout objet absent sur les partitions, mais visible sur les images originales.

2.- *Mise en correspondance de graphes :* suite à l'étape de resegmentation, l'appariement des graphes de voisinage est beaucoup plus simple, bien que restent à résoudre les différences causées par le mouvement des objets (déplacements, apparitions, disparitions ou occultations).

3.- *Edition des partitions par fusion des régions :* une fois que les structures des graphes ont été recalées, cette dernière étape d'édition décide du sort des régions qui n'ont pas trouvé de correspondant : quelques-unes vont fusionner avec des régions appariées, d'autres vont prendre un nouveau label.

A la sortie, l'algorithme rend la projection de la partition résultante à $(t-1)$ sur l'image courante. Cette projection comprend l'édition et la rellabelisation de la partition au temps (t) que nous avons choisie originalement dans hiérarchie.

Du point de vue des algorithmes de mise en correspondance, le suivi d'un objet au long d'une séquence revient à lui assigner le même label sur toute partition où il est visible. Dans ce contexte, on doit gérer quatre situations différentes,

Projection d'un label : un label est projeté pour chaque appariement établi entre une région à $(t-1)$ et une région en (t).

Disparition d'un label : un label disparaît pour chaque région de $(t-1)$ qui n'a pas trouvé de correspondante en (t),

Apparition d'un nouveau label : un label apparaît pour chaque région de (t) qui n'a pas trouvé de correspondante sur la partition résultante à $(t-1)$.

Réapparition d'un label : un label réapparaît pour chaque région de (t) qui correspond à une région disparue dans l'intervalle $[0, t-1)$.

La difficulté la plus importante étant de différencier les objets qui apparaissent pour la première fois de ceux qui viennent de réapparaître après occultation. Ainsi, dans le but précis d'établir le lien temporel qui assure la continuité dans cette labellisation, une deuxième étape de mise en correspondance est mise en œuvre. La nouveauté la plus importante concerne la gestion d'une mémoire temporelle.

7.3.3 La mémoire temporelle

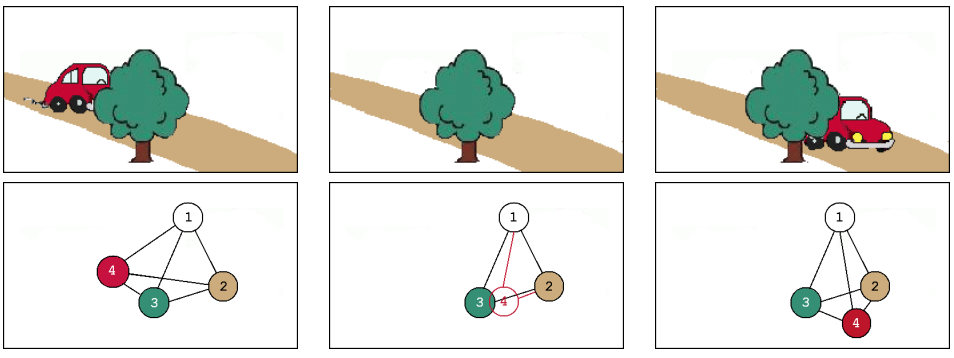
La reconnaissance d'un objet qui réapparaît après occultation est sans doute l'un des défis les plus ambitieux dans le domaine du suivi d'objets. Pour la plupart des systèmes de suivi, dès qu'un objet disparaît du champ visuel, il est irrémédiablement perdu. Exception prêt des occlusions courtes si un modélisation de la trajectoire est effectué.

Aucun des algorithmes classiques utilisés pour créer des séquences de partitions n'est capable de réassigner un label si une interruption se produit lors de la propagation. Ainsi, la segmentation 3D change le label d'un objet à chaque fois que la connexité temporelle est interrompue. De la même façon, la segmentation 2D récursive ne peut assurer la projection des labels que parmi de paires consécutives de trames. Dans tous les cas, l'absence d'un objet, même sur une seule image de la séquence, implique un changement de label.

7.3.3.1 Le graphe de mémoire

Dans le cadre de notre approche, la réapparition des labels est obtenue de façon naturelle lors que le graphe acquiert des attributs temporels. Une telle structure, nommée *graphe de mémoire*, va suivre l'évolution temporelle de tous les nœuds, qu'ils soient visibles ou occultés.

Sur la petite séquence synthétique qui suit nous avons illustré le rôle du graphe de mémoire dans la mise en correspondance de régions.



trame #0 trame #1 trame #2

FIG. 7.7 – Gestion des nœuds occultés dans le graphe de mémoire.

Au début de la séquence, le graphe de mémoire est initialisé à partir du graphe de voisinage de la partition. Cette équivalence entre les deux structures perdure jusqu'à ce qu'un nœud disparaisse.

Sur notre exemple, lorsque la voiture est occultée par l'arbre, elle ne fait plus partie de la partition. A ce moment-là, le graphe de mémoire garde tous les attributs associés à la voiture sur la dernière partition où elle était visible. Même si sur les trames suivantes l'objet est complètement occulté, le système continue à gérer l'ensemble de cette information dans le graphe de mémoire. En faisant ainsi, l'identification des objets qui réapparaissent devient possible.

7.3.3.2 La gestion du graphe de mémoire

Le graphe de mémoire garde l'évolution de tous les nœuds qui apparaissent au long de la séquence. Nous verrons par la suite les différentes étapes qui se succèdent lors de la gestion de cette information :

1.- Mise à jour des régions visibles

La première étape consiste à faire la mise à jour de l'ensemble des nœuds qui ont été correctement appariés lors du calcul de la mise en correspondance. Dans le graphe de mémoire tous ces nœuds ont un statut « *visible* », car ils représentent la totalité des objets visibles sur la partition au temps courant. Cependant, d'une image à la suivante, ces objets peuvent changer légèrement de couleur ou de forme, ou même de voisinage à cause du mouvement. Tous ces changements doivent être mis à jour, trame après trame, dans la mémoire.

$$\text{Cas 1 : nœud } \textit{visible} \text{ à } (t-1) \rightarrow \text{nœud } \textit{visible} \text{ à } (t)$$

2.- Gestion des régions occultées

Dans le graphe de mémoire, un nœud occulté garde les attributs de l'objet juste avant disparition, sauf en ce qui concerne la position. La trajectoire d'un objet disparu continue à être estimée jusqu'à ce qu'elle atteigne le bord de l'image, où elle reste figée. Nous faisons pourtant l'hypothèse que l'objet va réapparaître sur une trajectoire rectiligne. Néanmoins, avant de conclure qu'une région est encore occultée,

$$\text{Cas 2 : nœud } \textit{occulté} \text{ à } (t-1) \rightarrow \text{nœud } \textit{occulté} \text{ à } (t)$$

l'algorithme va essayer de la retrouver sur l'image courante. L'objectif étant d'assigner le même label aux objets avant et après occultation. Pour cela, nous ferons appel à une deuxième étape de mise en correspondance qui compare toute nouvelle région apparue au temps (t) avec l'ensemble des régions qui sont portées disparues au long de la séquence. Le résultat de ce processus peut entraîner un changement dans le statut du nœud.

$$\text{Cas 3 : nœud } \textit{occulté} \text{ à } (t-1) \rightarrow \text{nœud } \textit{visible} \text{ à } (t)$$

Par ailleurs, il se peut aussi qu'une région visible à $(t-1)$ ne le soit plus sur l'image à (t). Il s'agit d'objets qui, ou bien parce qu'ils viennent de disparaître ou bien parce qu'ils ont été différemment segmentés à (t), n'ont pas trouvé de correspondant sur la partition courante. Sur le graphe de mémoire ces nœuds vont prendre le statut « occulté ».

Cas 4 : nœud *visible* à $(t-1) \rightarrow$ nœud *occulté* à (t)

3.- Inclusion de nouvelles régions

La mise à jour du graphe de mémoire finit par l'inclusion des nouvelles régions apparues au temps courant. Ceci est faite seulement après avoir vérifié que ces nouvelles régions apparaissent pour la première fois dans la séquence. Elles seront incorporées au graphe de mémoire avec tous leurs attributs et ayant un statut « visible ».

Cas 5 : nœud *inexistant* à $(t-1) \rightarrow$ nœud *visible* à (t)

La puissance du *graphe de mémoire* sera mise en évidence dans la section suivante lors de la présentation des résultats sur des séquences réelles.

7.4 Application au suivi d'objets

Nous allons étudier ici la performance de nos algorithmes lorsqu'ils s'appliquent au suivi d'objets. Brièvement, rappelons d'abord les trois étapes qui vont se succéder dès l'arrivée de la séquence d'images jusqu'à la création de la séquence de masques :

1.- *Initialisation :* nous allons fournir au système la partition de la première trame de la séquence. Cette partition a été créée par édition manuelle de la hiérarchie de partitions (voir Section 4.3.2). Notre but ici est de fournir un découpage de la scène contenant les régions les plus représentatives, parmi lesquelles les objets qui seront à suivre.

2.- *Calcul de la mise en correspondance :* par la suite nous voudrions propager l'information fournie par la partition initiale à toute la séquence. Pour cela, nous ferons une mise en correspondance récursive, en prenant comme référence à l'instant (t) la partition résultante à $(t-1)$. Ceci jusqu'à la fin de la séquence.

3.- *Suivi d'objets :* une fois que la séquence de partitions a été créée, le suivi d'un objet revient à sélectionner l'ensemble des nœuds qui le représentent tout au long de la séquence.

Dans la Section 7.4.1 nous présenterons quelques-uns des résultats obtenus sur des séquences bien connues de la communauté scientifique. Ceci permettra au lecteur d'évaluer par lui-même la qualité et la robustesse du suivi. Les détails concernant le processus de suivi, ainsi qu'une discussion sur les performances de nos algorithmes feront le sujet de la Section 7.4.2.

7.4.1 Résultats en images

Pour illustrer la performance de nos algorithmes lorsqu'ils s'appliquent au suivi d'objets, nous allons montrer ici les résultats obtenus sur quatre séquences :

« Akiyo »

Cette séquence a été enregistrée à caméra fixe et suit l'évolution d'une présentatrice lors de l'émission d'un journal télévisé (voir Figure 7.8 (a)). La difficulté la plus importante concernant le suivi de ce personnage est liée à la segmentation des cheveux, le contraste avec le fond étant très faible.

La Figure 7.8 (b) montre un échantillonnage de la séquence de masques obtenue lorsque les nœuds qui représentent la présentatrice sont sélectionnés. Sur ces images on peut vérifier comment nos algorithmes arrivent à segmenter parfaitement la totalité de la séquence.

« Foreman »

La deuxième séquence que nous avons prise comme exemple rentre dans la catégorie des séquences de vidéophonie. Elle a été enregistrée avec une caméra tenue à la main et suit l'évolution d'un personnage qui parle face à la caméra (voir Figure 7.9 (a)). Bien que dans cette séquence il y ait aussi des régions à faible contraste (spécialement le casque), la différence la plus remarquable par rapport à l'exemple précédent concerne la présence de mouvements beaucoup plus importants. Plus concrètement, les mouvements du personnage sont forts et rapides, ce qui entraîne des changements significatifs dans le contenu de la scène, arrivant jusqu'à la modification de l'encadrement à la fin de la séquence.

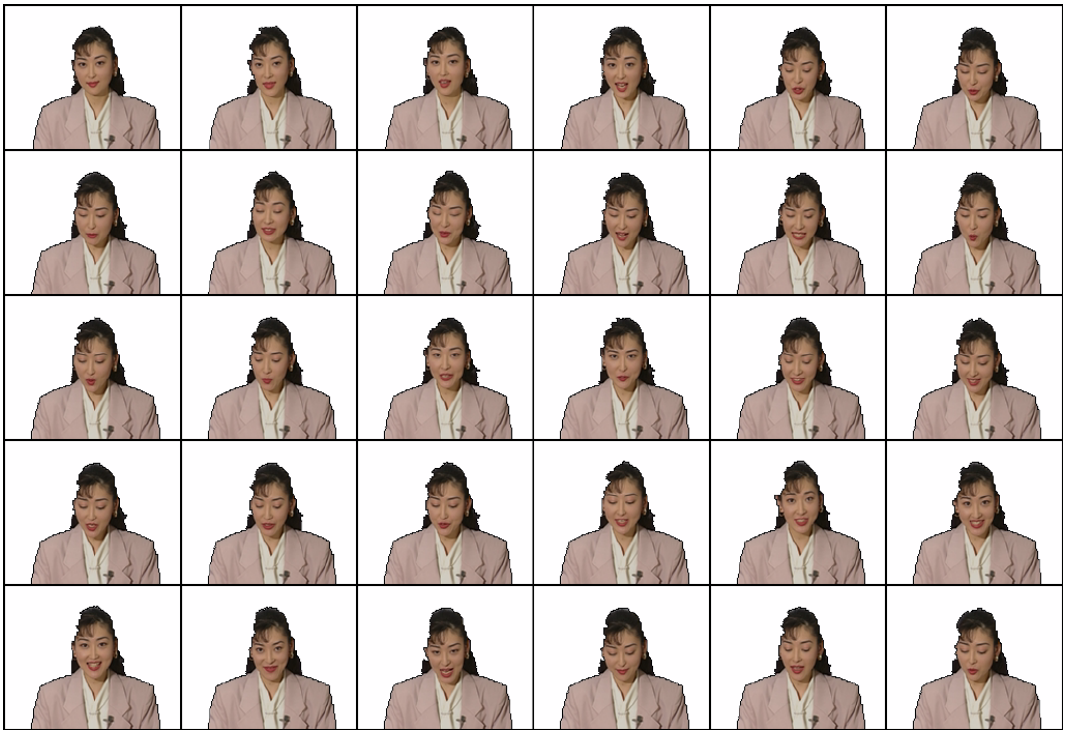
Les résultats du suivi du personnage se montrent sur la Figure 7.9 (b). On se rend compte ici comment le manque de contraste dégrade la qualité des contours sur certaines trames. Néanmoins, il est intéressant de remarquer que tous les déplacements des régions ont été suivis sans qu'aucune estimation de mouvement ne soit faite. On peut observer également sur ces images comment des régions qui disparaissent du champ visuel pendant un certain temps (épaules) sont reconnues lors de leur réapparition. Dans la section suivante nous verrons ceci plus en détail.

« Mother and daughter »

Ce troisième exemple correspond à une séquence enregistrée à caméra fixe avec deux personnages présents dans le premier plan (voir Figure 7.10 (a)). La particularité la plus importante ici est l'apparition du bras de la dame, car des régions significatives sont couvertes.

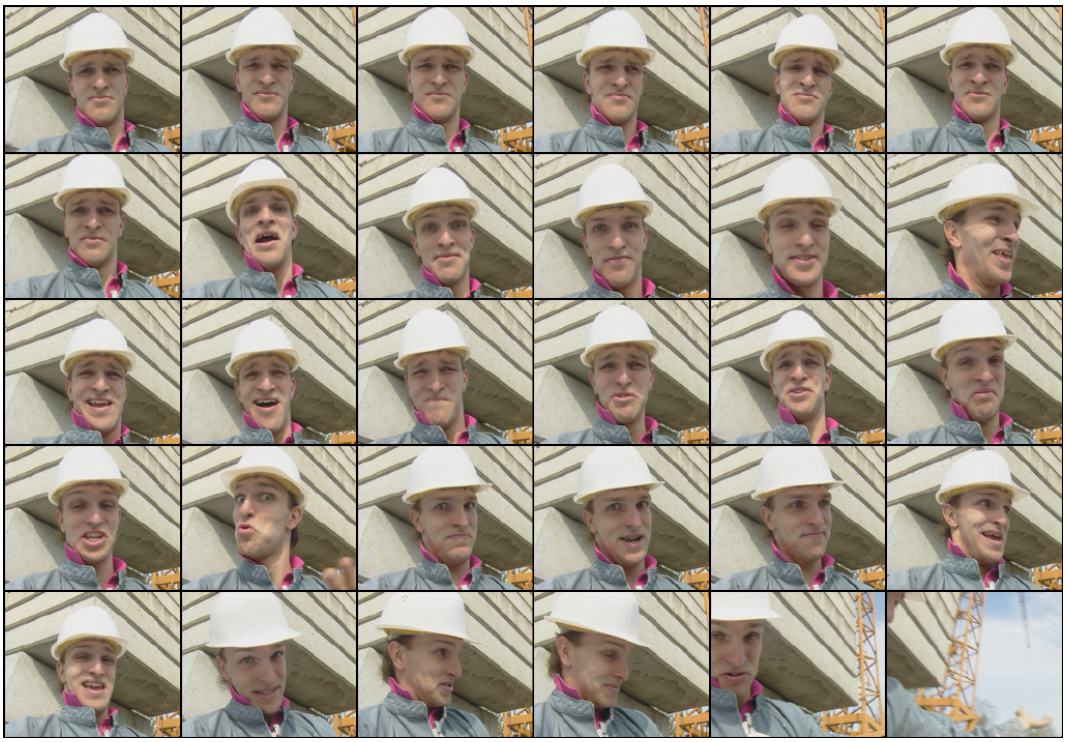
Afin d'évaluer le comportement de nos algorithmes face à l'apparition et disparition d'objets, nous allons suivre le visage et le bras de la dame (voir Figure 7.10 (b)). Bien que l'obtention de ces résultats soit étudiée en détail dans la section suivante, on peut déjà remarquer que la difficulté maximale est liée à la segmentation du bras, car son mouvement rapide créé des contours flous qui ont tendance à fusionner avec ceux du visage de l'enfant (voir trame 60).

(a) Trames : 0, 10, 20, 30,. . ., 290.



(b) Séquence des masques créés par sélection de nœuds.

FIG. 7.8 – Séquence « Akiyo ».

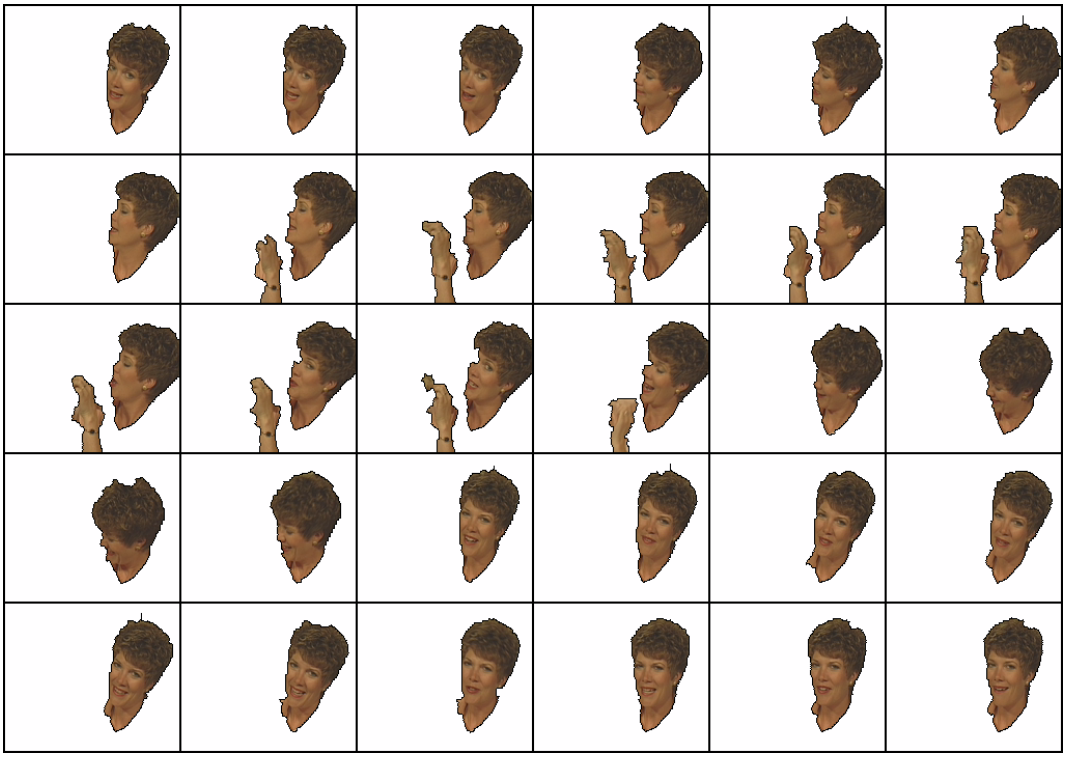


(a) Trames : 0, 10, 20, 30,. . ., 290.

(b) Séquence des masques créés par sélection de nœuds.

FIG. 7.9 – Séquence « Foreman ».

(a) Trames : 0, 10, 20, 30,. . ., 290.



(b) Séquence des masques créés par sélection de nœuds.

FIG. 7.10 – Séquence « Mother and daughter ».

(a) Trames : 0, 2, 4, 6,. . ., 60.

(b) Séquence des masques crées par sélection de nœuds.

Fig. 7.11 – Séquence « Ping-pong ».

« Ping-Pong »

La dernière séquence que nous avons prise comme exemple correspond à un extrait de l'enregistrement d'un match de ping-pong (voir Figure 7.11 (a)). Pendant les premières images de la séquence la caméra suit dans un plan rapproché l'évolution de la balle lors de la frappe. Ensuite, un éloignement progressif de la prise de vue découvre le corps du joueur. La difficulté du suivi dans cette séquence est bien connue, car :

- le mouvement extrêmement rapide de la balle fait qu'elle apparaît comme floue, parfois même déformée. Cet effet s'observe clairement sur plusieurs trames.
- la forme du bras varie fortement au long de la séquence, ce qui rend complexe la continuité du suivi.
- l'éloignement final de la prise de vue provoque une diminution de la taille des objets, ainsi que l'apparition des nouveaux éléments dans le champ de vue. L'union de ces deux effets compromet le suivi des objets petits comme la balle.

Tout ceci porte l'ensemble des techniques de suivi au limite de leurs performances. Effectivement, le manque de connexité temporelle rend la segmentation 3D inefficace ; tandis que la complexité du mouvement créé des erreurs insurmontables pour les systèmes qui en font usage pour projeter une segmentation 2D. Nos algorithmes sont aussi en difficulté, car comme l'illustre la Figure 7.11 (b), la balle n'est plus détectée dès que la complexité de la scène augmente. Les détails de la mise en correspondance seront donnés lors de l'étude du chronogramme des nœuds.

7.4.2 Discussion sur les performances

Pour chacune des séquences présentées auparavant, nous allons donner ici quelques détails concernant le processus de mise en correspondance à la base du suivi des objets. L'objectif de cette étude est double : d'un côté il s'agit de montrer les atouts de nos algorithmes lorsqu'ils surmontent quelques-unes des contraintes les plus importantes liées à la segmentation de séquences ; de l'autre côté il s'agit de montrer leur limites, afin de pouvoir proposer de futures améliorations et encadrer leur champ d'application.

Avant de particulariser nos commentaires à la problématique précise de chaque séquence, il est intéressant de souligner que l'ensemble des partitions initiales ont été composées par édition manuelle de la hiérarchie de partitions, dans le but d'obtenir une partition de référence sur laquelle,

- chacune des régions représente un objet, ou une partie d'un objet, sans créer de faux contours à cause des textures ou des ombres ;
- le nombre de régions reste faible dans le but de minimiser la complexité du traitement de la séquence.

« Akiyo »

En haut de la Figure 7.12 apparaît la partition de référence qui a été fournie pour initialiser la mise en correspondance. Cette partition contient dix régions, dont sept représentent l'objet d'intérêt et trois sont consacrées au fond. En bas, sur trois trames de la séquence, on peut comparer la topologie de la partition obtenue au temps (t) (entrée du bloc de mise en correspondance de régions visibles dans la Figure 7.6), avec le résultat de la projection de la partition à $(t-1)$ sur elle (sortie du bloc de gestion de la mémoire temporelle).

Au long de toute la séquence, la partition au temps (t) s'extrait sur le niveau à 13 régions de la hiérarchie, donnant un découpage proche (mais pas forcément égal) à celui de la partition de départ. On observe cependant que les cheveux ont toujours tendance à fusionner avec le fond, tandis que la veste apparaît parfois fragmentée. La réorganisation de l'ensemble de ces régions sur la partition finale est prise en compte par le processus de mise en correspondance.

Le chronogramme de la Figure 7.13 corrobore la robustesse du système lors de la création de cette séquence de partitions. Aucun nœud n'a été perdu, aucune itération du processus de relaxation n'a été nécessaire. Celui-ci étant un exemple du cas le plus favorable, nous verrons par la suite comment la complexité augmente dans des séquences ayant plus de mouvements.

« Foreman »

En appliquant le même type d'analyse, nous avons illustré dans la Figure 7.14 la partition originale fournie sur la première des trames, ainsi que quelques-unes des partitions de la séquence. Par la suite, nous allons étudier ces images en détail, car la variété des événements qui ont lieu dans cette séquence illustre parfaitement les atouts les plus importants de notre méthode.

On remarque rapidement sur la colonne de gauche que la segmentation des images d'entrée est beaucoup plus instable que celle obtenue sur la séquence précédente. Ceci s'explique par le fait que ces partitions ont été calculées au moyen d'une technique de segmentation hiérarchique purement 2D. Ainsi, plusieurs facteurs sont à l'origine de leurs différences, dont les mouvements rapides du personnage, le faible contraste existant sur certaines zones, les oscillations de la caméra, et par-dessus de tous ceux-ci, le changement de plan à la fin de la séquence. Cette variabilité contraste avec la topologie beaucoup plus régulière des graphes à la sortie du processus de mise en correspondance (voir colonne à droite). Bien que de petites fuites puissent se produire sur les contours faiblement contrastés, la robustesse des algorithmes face à l'instabilité de la segmentation devient évidente.

Néanmoins, l'atout le plus important de nos résultats concerne la continuité dans la labellisation des nœuds qui ont souffert d'une disparition temporelle. Comme exemple de ceci il suffit de suivre l'évolution de l'épaule droite du personnage :

– des trames 0 à 65 l'épaule est parfaitement visible, étant suivie sans problème par l'algorithme à partir du nœud 9 du graphe ;

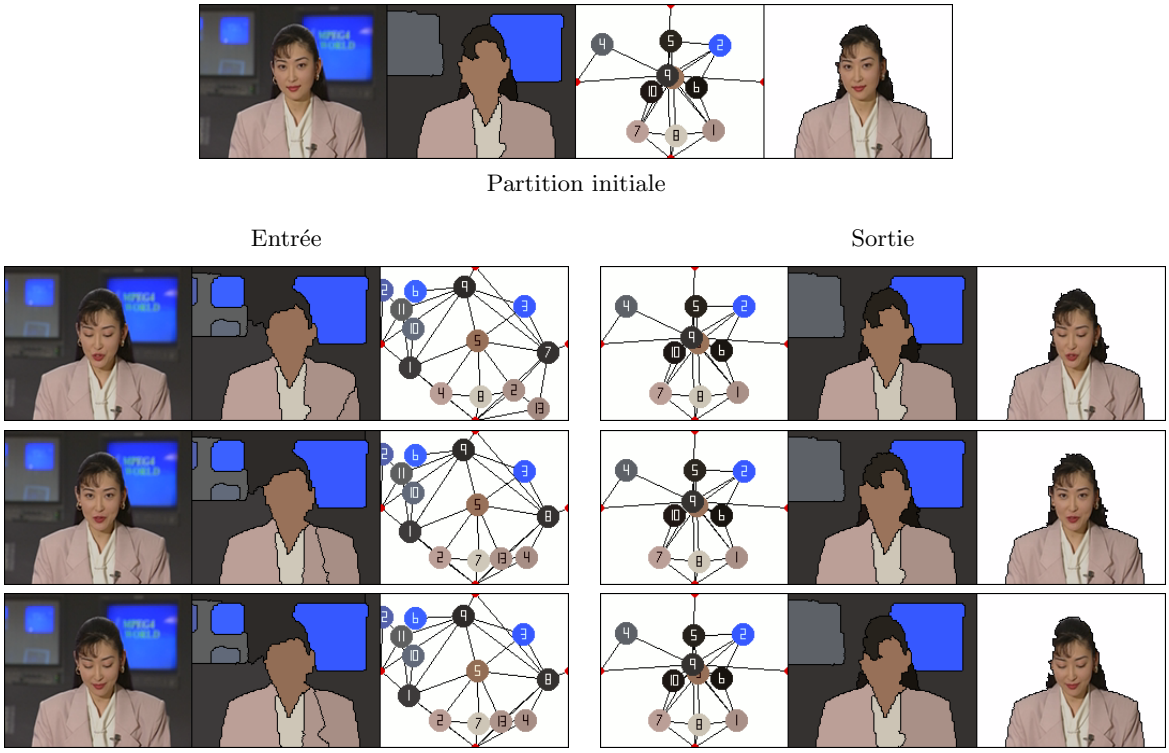


FIG. 7.12 – Séquence « Akiyo ». Détails concernant la mise en correspondance de graphes sur les trames : 40, 150 et 270.

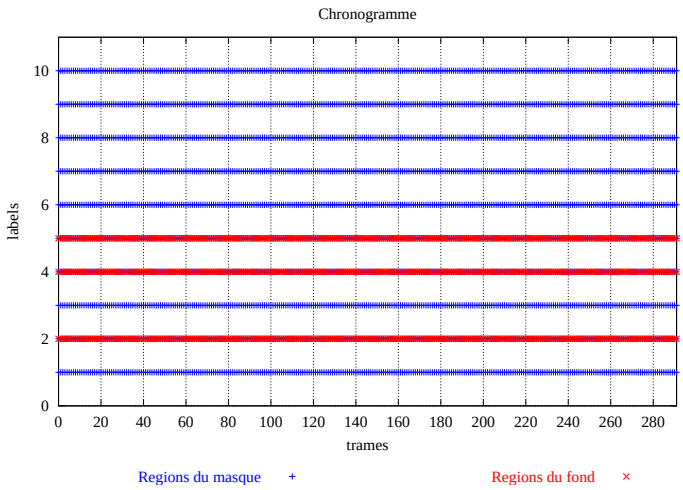


FIG. 7.13 – Séquence « Akiyo ». Chronogramme des nœuds. La séquence de masques de la Figure 7.8 (b) a été créée par suivi des labels : visage (3), cheveux (6, 9, 10), veste (1,7) et chemisier (8).

– des trames 66 à 80 elle sort progressivement du cadre de l'image, étant détectée pour la dernière fois sur la trame 76. A ce moment-là les attributs de l'épaule restent figés dans le graphe de mémoire sous un statut de région occultée ;

– des trames 81 à 130 l'épaule droite n'est plus visible sur un plan rapproché. Pendant cet intervalle de temps le nœud 9 n'apparaît pas sur le graphe de la partition mais reste en mémoire ;

– à partir de la trame 130 l'épaule réapparaît dans le champ visuel, étant reconnue sur la trame 134 grâce à l'information gardée dans le graphe de mémoire. Le nœud 9 passe au statut visible et réapparaît à son tour sur la partition.

L'évolution temporelle du nœud qui représente l'épaule, ainsi que celle de tous les nœuds qui ont fait partie du graphe de mémoire peut être suivie en détail sur les différentes graphiques de la Figure 7.15. Sur le chronogramme, chaque point (t, l) indique la présence du label l à la trame t. On a marqué en bleu l'ensemble des régions qui font partie du masque, traçant en rouge les régions du fond. Les deux autres graphiques montrent la variation du nombre de régions du graphe de mémoire, en détaillant les nouvelles apparues, les disparues, les visibles et les occultées à chaque instant.

L'apparition de beaucoup de nouvelles régions sur la fin de la séquence attire ici l'attention. Les seules régions nouvelles qui se projettent de façon stable correspondent au ciel (nœud 23) et à une grue (nœud 25). Du point de vue d'une application de suivi d'objets, cette apparition fugace de nœuds qui n'ont pas de continuité temporelle est dépourvue d'intérêt. Par contre, si on s'intéresse à l'analyse du contenu des séquences, ce manque de correspondance dans le fond pointe une transition dans l'encadrement.

Finalement, il est intéressant de remarquer le double rôle joué par le graphe de mémoire, qui permet de reconnaître non seulement les objets qui réapparaissent après occultation, mais aussi des régions perdues suite à une erreur dans la segmentation. On en trouvera un exemple dans le nœud 2 qui représente la partie supérieure gauche du bâtiment. Ce nœud prend un statut occulté pendant toutes les trames où cette zone apparaît fragmentée. Par ailleurs, dès qu'elle est segmentée à nouveau comme une seule région, le système arrive à réassigner le même label. Un tel comportement augmente la robustesse du système face aux erreurs de la segmentation.

« Mother and daughter »

A l'égal des exemples précédents, la Figure 7.16 donne un rapide aperçu de la complexité de la mise en correspondance sur cette séquence. Bien que la partition de départ ne contient que 10 régions, nous avons forcé le système à travailler à un niveau de résolution plus fin (16 régions), dans le but de pouvoir détecter correctement le bras lorsqu'il rentre dans la scène. Par ailleurs, le décalage existant entre le nombre de régions de la partition extraite de la hiérarchie, et le nombre de régions de la partition projetée, provoque l'apparition progressive de nouvelles régions.

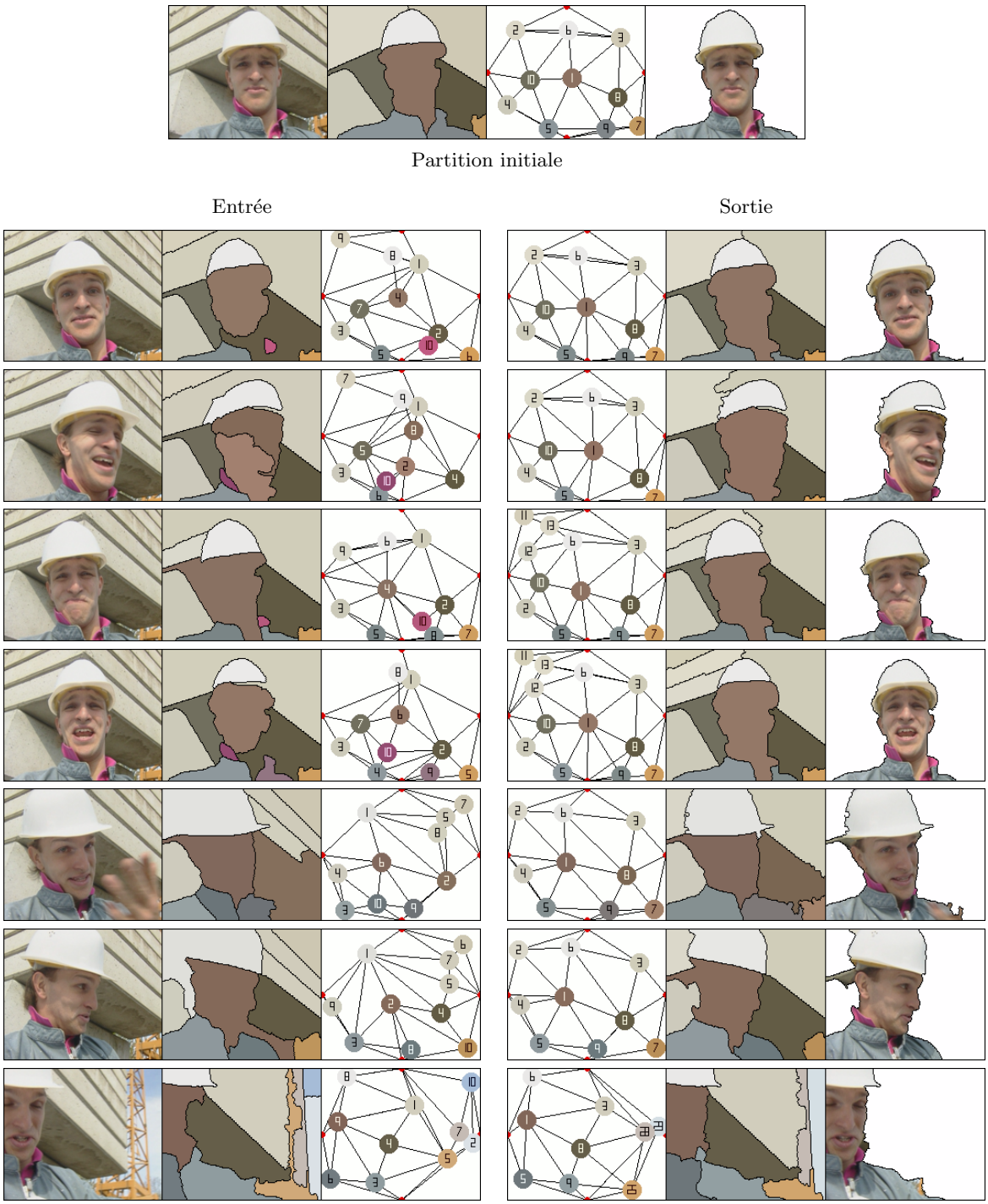


FIG. 7.14 – Séquence « Foreman ». Détails concernant la mise en correspondance de graphes sur les trames : 76, 113, 136, 156, 253, 263 et 282.

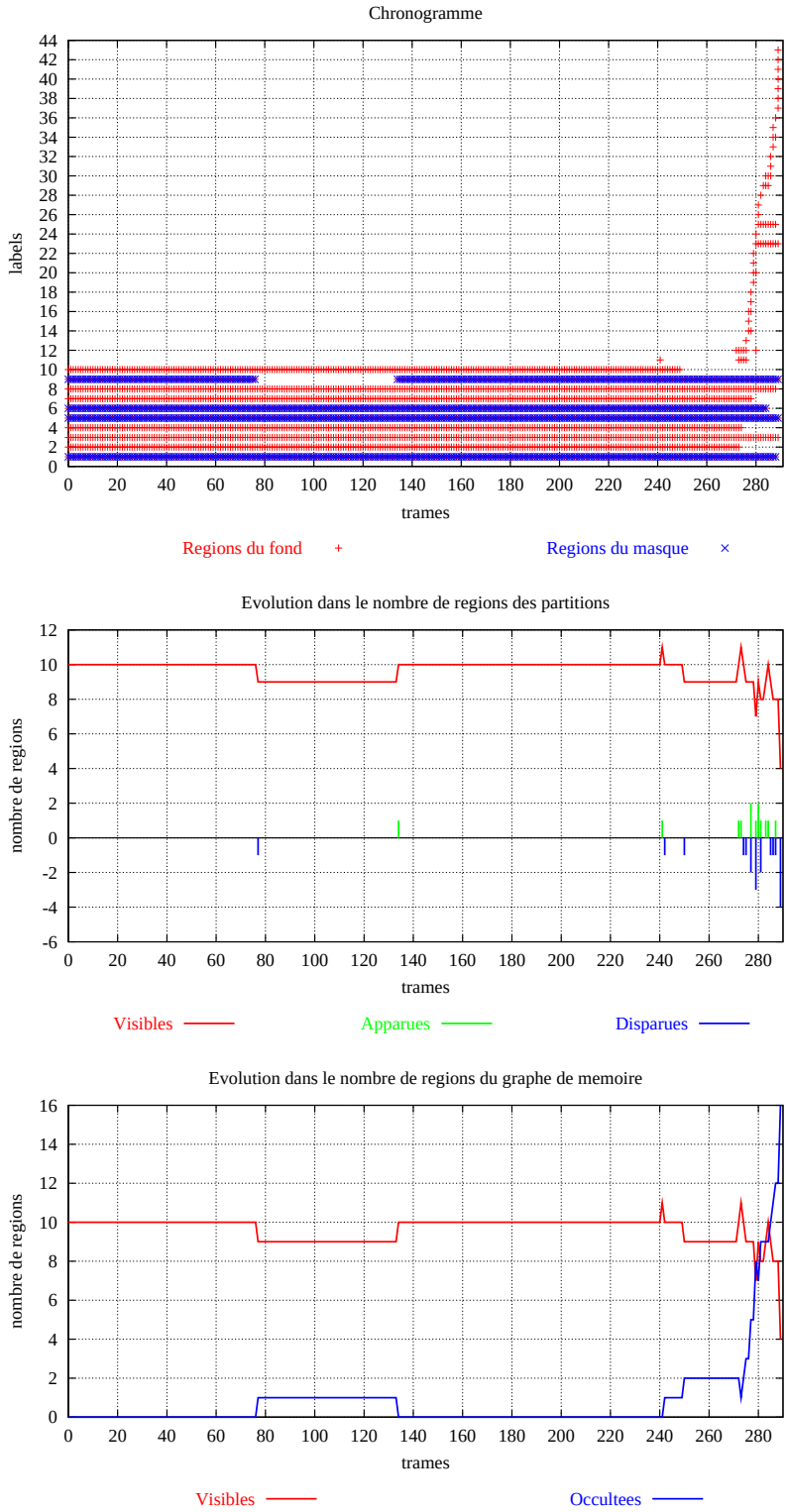


FIG. 7.15 – Séquence « Foreman ». Chronogrammes concernant l'évolution temporelle des nœuds dans le graphe de mémoire. La séquence de masques de la Figure 7.9 (b) a été créée par suivi des labels : visage (1), casque (6), épaules (5,9).

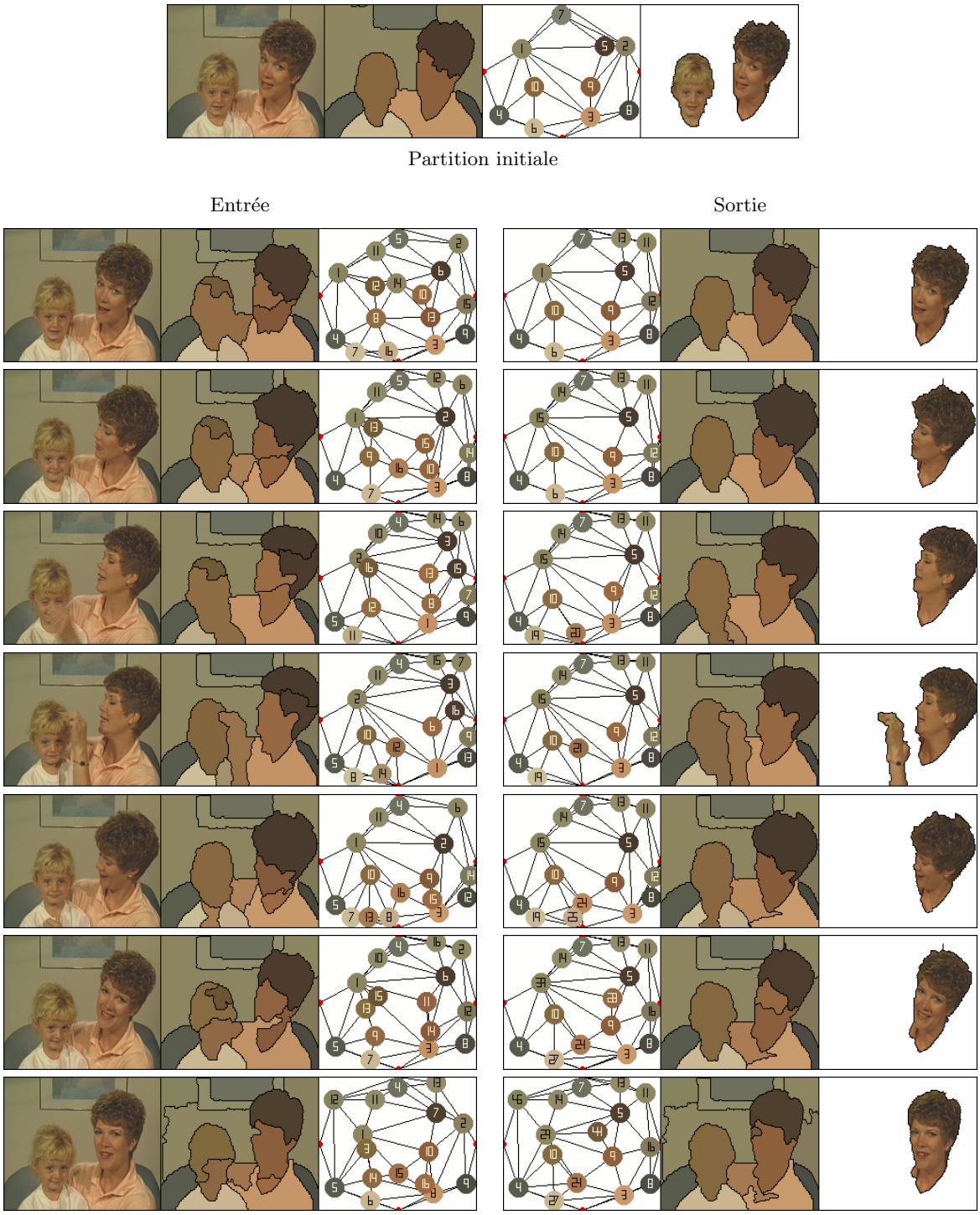


FIG. 7.16 – Séquence « Mother and daughter ». Détails concernant la mise en correspondance de graphes sur les trames : 20, 40, 60, 80, 240 et 280.

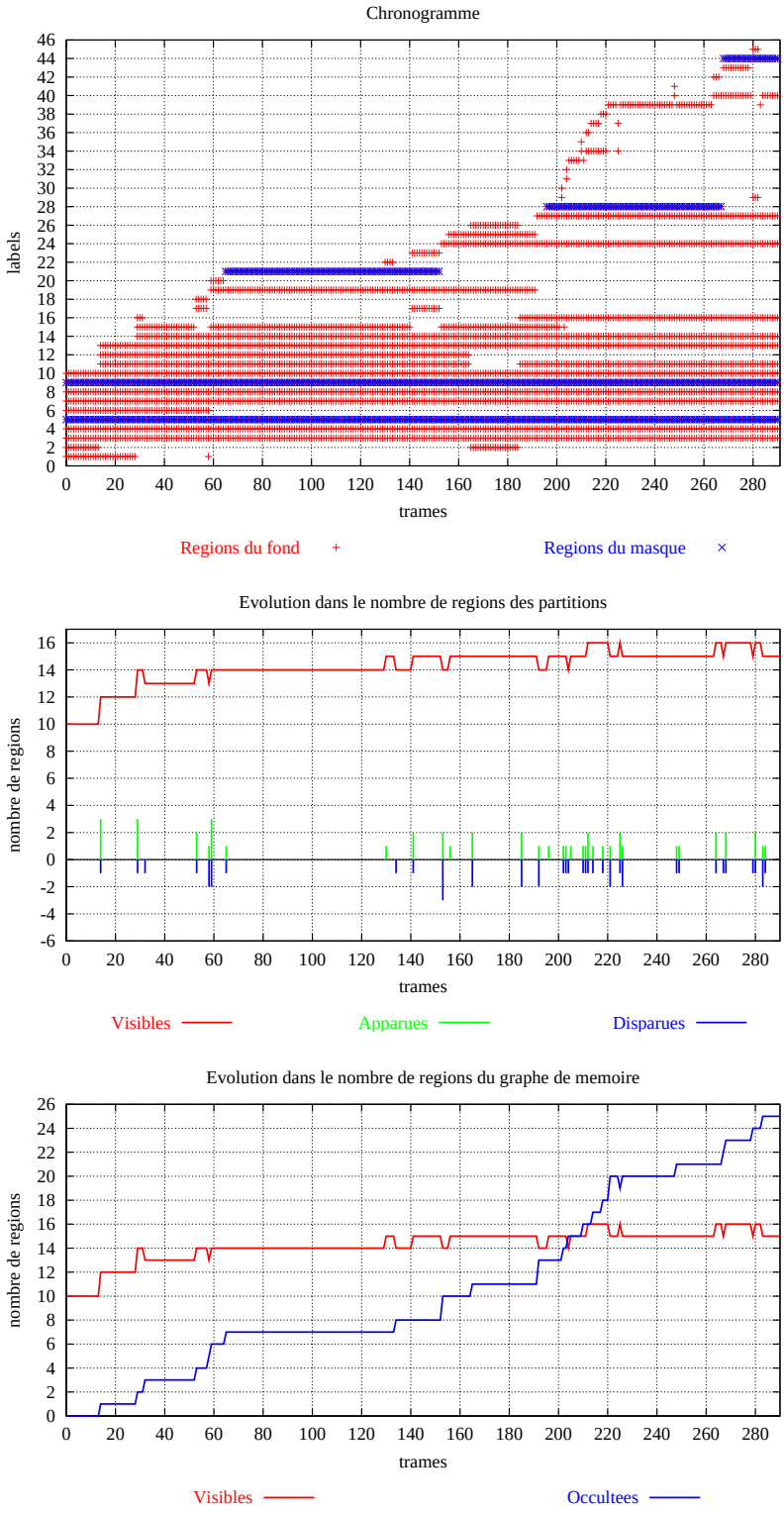


FIG. 7.17 – Séquence « Mother and daughter ». Chronogramme des nœuds. La séquence de masques de la Figure 7.10 (b) a été créée par suivi des labels : visage (9, 23, 44), cheveux (5) et bras (21).

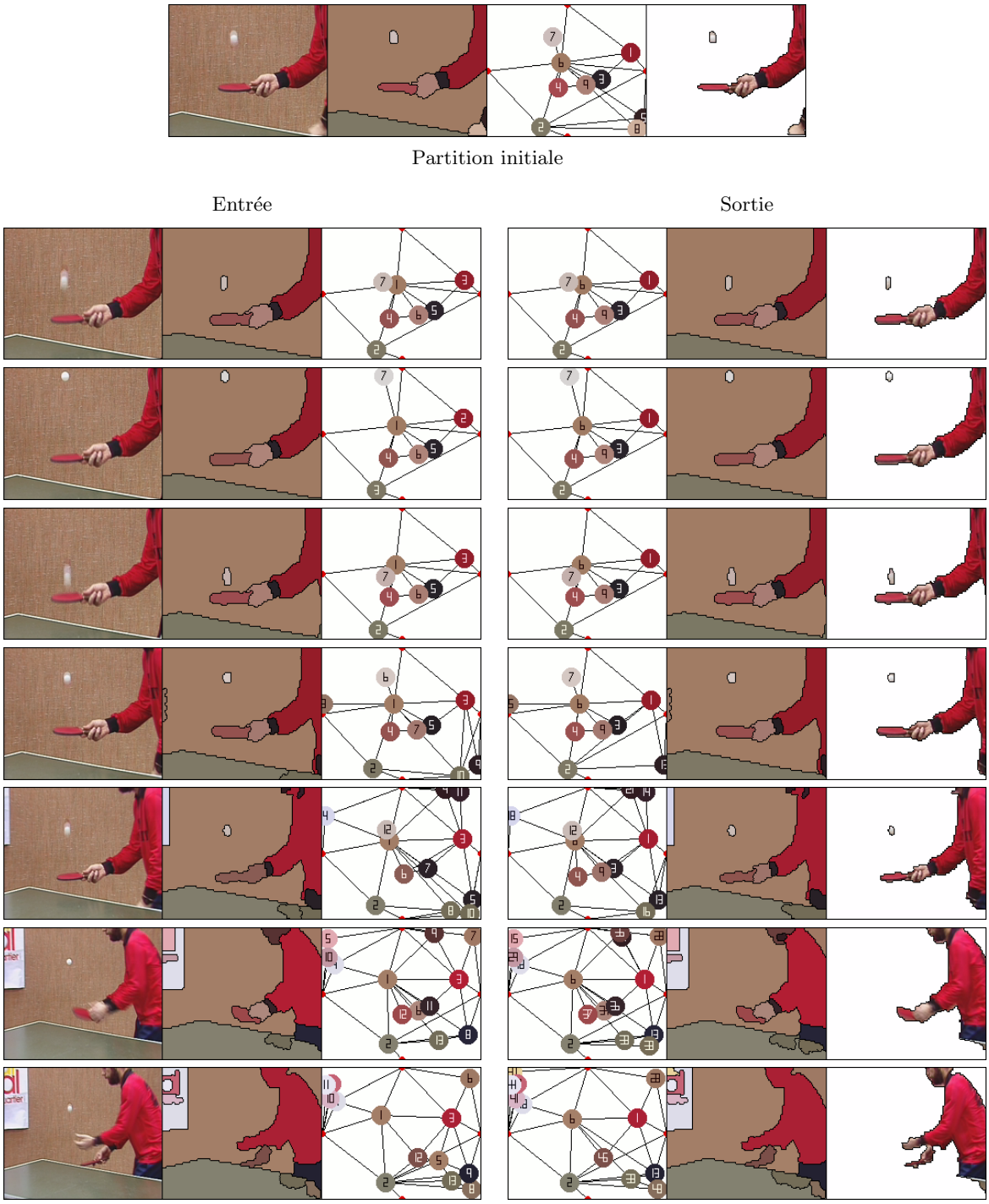


FIG. 7.18 – Séquence « Ping-Pong ». Détails concernant la mise en correspondance de graphes sur les trames : 9, 18, 25, 30, 37, 47 et 50.

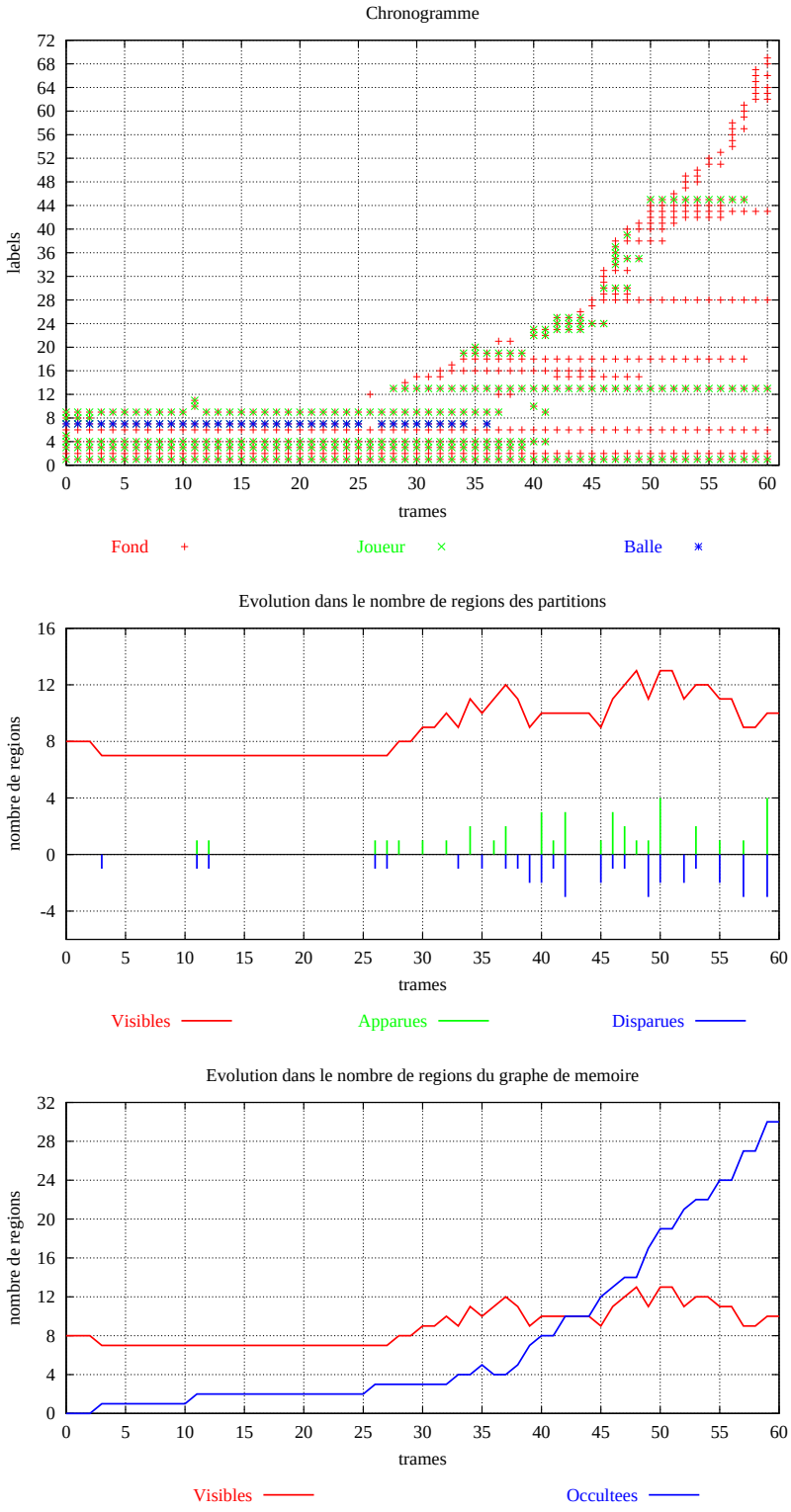


FIG. 7.19 – Séquence « Ping-Pong ». Chronogrammes concernant l'évolution temporelle des nœuds dans le graphe de mémoire.

On observe ainsi que la mise en correspondance fait évoluer la finesse de la séquence de partitions jusqu'à se stabiliser autour du nombre de régions imposé à la segmentation 2D. En particulier, plusieurs objets qui n'étaient pas présents sur la partition de départ vont ressortir lentement, comme par exemple le cadre du tableau (nœuds 13 et 14).

Cependant, les transitions les plus brusques dans le chronogramme de la séquence (voir Figure 7.17) sont en lien étroit avec les mouvements du bras :

- trame 60, le bras apparaît,
- trame 150, le bras se déplace vers le bas de l'image,
- trame 200, le bras disparaît du champ de vue.

Et pourtant la complexité de la mise en correspondance sur ces transitions n'est pas causée par l'intensité du mouvement, mais par le manque de contraste et les recouvrements qui se produisent. Observer par exemple l'évolution du bras sur le visage de l'enfant dans la trame 60 et les changements que ceci entraîne sur la topologie de la partition.

Sur la fin de la séquence nous avons détecté une certaine instabilité dans la segmentation du fond. Ceci s'explique par le fait que nous avons imposé une valeur constante dans le nombre de régions de la partition à l'entrée. Par conséquent, lorsque le nombre de régions exigé est supérieur au nombre d'objets significatifs dans l'image, l'algorithme de segmentation procède à une division des zones les plus larges. Les contours de ces nouvelles régions se placent sur des transitions texturées ou sur des ombres, difficilement appariées d'une image à la suivante. Même si cette augmentation dans le nombre de labels utilisés n'a pas de conséquences sur les objets suivis, nous estimons qu'elle pourrait être contrôlée en adaptant le nombre de régions de la segmentation à la complexité de la scène.

« Ping-Pong »

Nous allons étudier par la suite le comportement de nos algorithmes sur une des séquences les plus complexes pour les techniques de suivi d'objets. Les facteurs qui rendent le suivi difficile ont déjà été annoncés lors de la présentation de la séquence. Il s'agit par conséquent d'étudier comment les masques du joueur et de la balle ont été générés.

A l'égal des séquences précédentes, nous avons fourni dans la Figure 7.18 la partition initiale ainsi que quelques-unes des partitions qui font partie de la séquence finale. On doit remarquer la qualité du suivi sur les images du plan rapproché (trames 0 à 30), ce qui permet d'extraire des informations complémentaires. En guise d'exemple, la Figure 7.20 montre la trajectoire décrite par la balle, les coordonnées étant calculées à partir du déplacement de la région 7 de la séquence de partitions.

Par ailleurs, de forts changements se produisent dans la topologie du graphe à partir de la trame 30, où la prise de vue s'éloigne du joueur : de nouvelles régions apparaissent dans le cadre de l'image et d'autres disparaissent car leur importance relative diminue. Dès cet

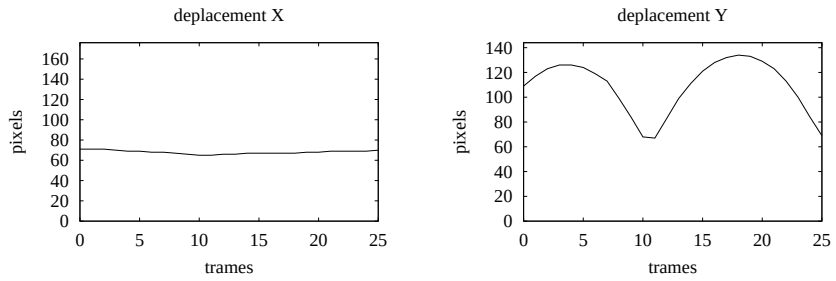


FIG. 7.20 – Evolution de la trajectoire de la balle sur les 25 premières trames de la séquence.

instant, le suivi de la balle devient de plus en plus fragile, car l'algorithme donne priorité à la segmentation des plus grandes régions qui apparaissent progressivement dans la prise de vue (le poster dans le fond et le corps du joueur). A ce moment-là, même le suivi du joueur devient complexe. A cause des mouvements rapides des bras et des changements dans la forme et les proportions du corps, beaucoup de régions ne sont pas appariées (voir le chronogramme de la Figure 7.19). De meilleurs résultats pourraient être obtenus en modélisant l'objet à suivre dans le calcul de la mise en correspondance. Ceci sera l'un des futurs travaux proposés à la fin de ce chapitre.

« Akiyo - Foreman - Akiyo »

Le but de ce dernier exemple n'est pas de créer une séquence de masques, comme nous l'avons fait auparavant, mais d'étudier le comportement du graphe de mémoire lors d'une transition brusque dans le contenu de la scène. Pour illustrer ceci nous avons créé une coupure artificielle en incluant deux images de la séquence « Foreman » parmi celles de la séquence « Akiyo ».

La Figure 7.21 contient l'ensemble de ces images ainsi que la séquence de partitions et graphes résultants. Sur ces résultats deux faits attirent fortement l'attention :

- d'un côté, la réapparition de la totalité des labels de la partition #1 sur la partition #4 suit à la coupure ;
- de l'autre côté, le lien établi entre le visage de la présentatrice et celui de l'ouvrier (voir la continuité du nœud 6 dans le chronogramme de la Figure 7.22).

Le premier de ces faits est un atout important du graphe de mémoire. Car il permet au système de porter à terme la mise en correspondance entre la totalité des régions de la partition #4 qui n'ont pas été appariées avec celles en #3 , et l'ensemble des régions qui ont disparu dans la partition #1. Néanmoins, on doit remarquer que le lien temporel ne sera établi que s'il existe une forte ressemblance entre la partition où les régions disparaissent (#1) et celle où les régions réapparaissent (#4). En supposant que le contenu de ces images est déjà proche, la proximité dans leurs partitions s'obtient facilement lorsqu'aucune réorganisation

manuelle des régions n'est forcée sur la partition initiale.

Le deuxième des faits mérite aussi quelques commentaires, car le niveau de ressemblance entre les deux visages peut paraître trop faible pour donner lieu à l'appariement de leurs nœuds. Par ailleurs, le comportement de l'algorithme s'explique ici facilement considérant que la mise en correspondance n'a pas été calculée sur les partitions montrées, mais sur deux versions rendues plus ressemblantes par resegmentation (étape d'édition de l'algorithme de mise en correspondance). Ainsi, le visage de la présentatrice rentre parfaitement avec une région dans la partie centrale du visage de l'ouvrier, toutes les autres parties non identifiées étant refusionnées à la sortie.

7.5 Conclusions et perspectives

Ce chapitre clôt la deuxième partie de cette thèse ayant pour sujet central la mise en correspondance de partitions. Nos apports dans ce domaine concernent la présentation d'une nouvelle méthode qui combine des algorithmes classiques de mise en correspondance par graphes avec de nouvelles techniques de segmentation morphologique. Les atouts les plus importants consistent en l'inclusion de

- deux étapes d'*édition des partitions*, faisant se rassembler au maximum le contenu des partitions à l'aide de la segmentation hiérarchique. Ceci permet de surmonter l'instabilité de la segmentation.

- un *graphe de mémoire* qui, en suivant l'évolution des nœuds occultés, offre la possibilité de reconnaître un objet s'il réapparaît après occultation.

La robustesse de ces algorithmes a été validée en deux étapes : d'abord lors de l'appariement d'images fixes (Chapitre 6), ensuite pour créer des séquences de partitions en vue du suivi d'objets (Chapitre 7). Dans ce domaine, nous arrivons à traiter une large variété de séquences sans avoir besoin de l'information de mouvement. Ceci rend nos résultats indépendants des erreurs qui apparaissent couramment lors de l'estimation des vecteurs de mouvement. Par ailleurs, en ce qui concerne le coût computationnel, le calcul de la mise en correspondance est abordable[1], même si une implémentation optimisée reste à faire.

Au vu des bons résultats obtenus, nous envisageons une suite dans l'usage de ces algorithmes. Raison pour laquelle il nous paraît intéressant de mentionner ici l'ensemble des perspectives qui s'ouvrent aux travaux futurs :

Incorporation d'un modèle d'objet vis-à-vis des applications de suivi.

L'algorithme de mise en correspondance que nous avons mis en œuvre créé une séquence de partitions en traitant à égalité toutes les régions qui y apparaissent. La séquence de masques d'un objet est composée *à posteriori* par sélection d'une ou plusieurs régions

[1] Le temps de calcul par image étant actuellement de l'ordre de quelques secondes sur un pentium à 266MHz.

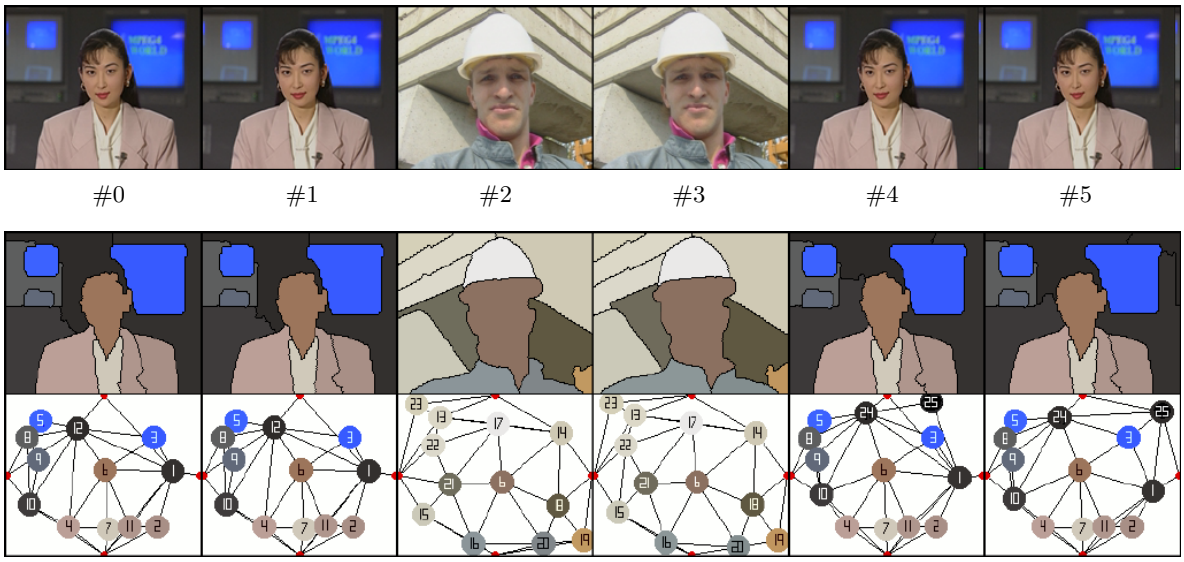


FIG. 7.21 – Détails de la mise en correspondance sur une coupure artificielle créée par composition d'images des séquences « Akiyo » et « Foreman ».

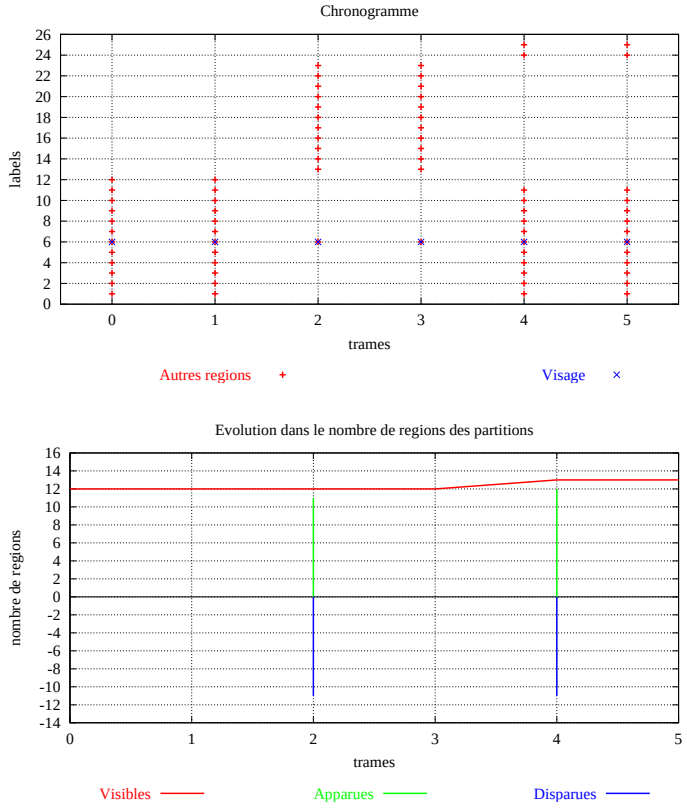


FIG. 7.22 – Séquence « Akiyo - Foreman - Akiyo ». Chronogrammes concernant l'évolution temporelle des nœuds dans le graphe de mémoire.

dans la séquence de partitions. Ainsi, nous croyons qu'une amélioration importante dans les résultats serait obtenue en incorporant un *graphe modèle* apportant de l'information *à priori* sur les objets à suivre. Ceci permettrait,

- aux filtres et à la segmentation de mieux s'adapter aux attributs intrinsèques des objets (la couleur, la taille, ...) ;
- au processus de mise en correspondance de mieux préserver leurs attributs contextuels (relations de voisinage, ...).

Exploitation d'autres attributs.

Un deuxième axe naturel de continuation concerne l'inclusion de nouveaux attributs dans la valuation des nœuds et des arêtes du graphe. Ceci permettrait d'augmenter la robustesse de la mise en correspondance, aussi bien des nœuds visibles que des nœuds occultés. Parmi les nombreuses options possibles, nous proposons d'inclure dans le graphe de mémoire des attributs sur,

- l'évolution temporelle des nœuds, permettant de donner plus de poids à l'appariement des nœuds présents dans la scène depuis plus longtemps.
- le mouvement et l'ordre de profondeur parmi les objets, de façon à ce que la combinaison de ces informations puisse donner un support à la détection et au suivi des objets occultés.
- l'ordre des régions dans la hiérarchie de partitions, dans le but de donner un support au calcul de la similarité parmi les nœuds.

Détection et correction automatique d'erreurs.

Une fois ces améliorations apportées, il est aussi envisageable d'inclure un contrôle d'erreurs agissant sur les nœuds de vie très courte. Car, dans la plupart des cas, ceux-ci ne sont que des erreurs de segmentation qui apparaissent comme de fausses apparitions d'objets. Face à eux, le système peut agir à fur et à mesure que la séquence est traitée, ou à posteriori par analyse des résultats obtenus, de façon à

- forcer leur disparition par fusion avec d'autres régions de projection stable ;
- établir des liens parmi les nouvelles régions et l'absence d'autres nœuds dans la séquence.

Contrôle de la taille du graphe de mémoire.

Il nous paraît aussi important de procéder au contrôle de la taille du graphe de mémoire par oubli progressif des nœuds occultés depuis plus longtemps. Ceci dans l'intérêt de

- maintenir constante la complexité de calcul, car l'augmentation progressive dans le nombre de nœuds occultés finit par ralentir le processus de la mise en correspondance ;

– éviter des réapparitions erronées, car à long terme de nouveaux objets pourraient être confondus avec d'anciens objets disparus.

Finalement, nous voudrions conclure cette énumération de perspectives en faisant référence au domaine d'application de nos algorithmes. Bien nous ayons porté nos efforts sur la création de séquences de masques en vue du suivi d'objets, il serait également envisageable d'utiliser la séquence de partitions comme point de départ d'une analyse de séquences. Car, beaucoup d'autres informations peuvent découler de l'étude détaillée de nos résultats. L'évolution du chronogramme donne une information sur les changements qui se produisent dans le contenu global de la scène, ou plus concrètement, sur l'apparition ou disparition d'un certain objet. L'étude de la séquence de partitions permet de suivre l'évolution temporelle de l'ensemble des caractéristiques des régions, comme la trajectoire, les changements dans la forme ou dans la taille, ...Par ailleurs, la topologie des graphes résultants permet l'étude des relations contextuelles parmi les régions.

Ceci clôt la description de l'ensemble des algorithmes que nous avons développés au cours de cette thèse en suivant une approche généraliste. Par opposition, le problème que nous allons aborder dans la dernière partie de ce document entraîne des contraintes bien précises, car il s'agit de mettre en œuvre une application de suivi d'objets en temps réel.

Troisième partie

Mise en œuvre d'une Application : le projet M4M

Chapitre 8

Introduction

La souplesse et la puissance de la norme MPEG4 ouvre les portes à des nombreuses applications qui vont du codage orienté objet à des nouvelles fonctionnalités multimédia. Cette complexité en hausse demande des algorithmes spécifiques. Ainsi, le projet européen M4M a été conçu pour donner des solutions techniques dans le domaine des communications mobiles en temps réel. Dans le cadre de ce projet, nos travaux vont de l'étude de viabilité à la réalisation d'un démonstrateur temps réel capable de segmenter et suivre un locuteur dans des séquences de vidéophonie ou vidéconférence. L'ensemble des algorithmes produits font actuellement l'objet d'une valorisation industrielle.

8.1 Description de l'application

Le projet M4M (MPEG fo(u)r mobiles) [58], auquel nous avons participé, a concentré pendant quatre ans les efforts des plus grandes compagnies européennes dans le domaine des télécommunications (Bosch, Philips, Thomson, STMicroelectronics, Siemens et Sican) vers la conception d'une nouvelle génération de terminaux mobiles, capables de recevoir et d'émettre des séquences vidéo encodées selon la nouvelle norme MPEG4.

Parmi les nouvelles fonctionnalités qui vont apparaître dans le domaine de la vidéophonie et la vidéoconférence, nos recherches feront possible le détourage automatique du locuteur. Ceci permettrait, par exemple, de regrouper divers locuteurs dans une même salle virtuelle, de préserver le caractère privé du domicile ou d'adapter le codage au différents objets de la scène.

séquence enregistrée | détourage automatique | séquence transmise

FIG. 8.1 – Intérêt de la segmentation pour un système de vidéoconférence.

8.1.1 Les Contraintes

Nous allons résumer ici le cahier des charges qui nous a été imposé sous la forme d'une série de contraintes. Ainsi, pour être mis en œuvre pour des communications mobiles, les algorithmes devront être capables de traiter les données en temps réel et en conditions d'enregistrement réalistes, permettant l'usage de caméras mobiles, des conditions d'éclairage médiocres, et un fond non figé avec des personnes qui peuvent se déplacer en arrière-plan. Tout ceci peut être assumé en trois points :

Traitement des données en temps réel : le système devra être capable de traiter une séquence de vidéophonie à une cadence de 10 images par seconde en format QSIF (160x120 pixels). Ceci nous oblige à concevoir des solutions efficaces les plus simples possibles.

Traitement des données en mode automatique : le système devra travailler de façon automatique, en demandant un minimum d'interaction à l'utilisateur, celle-ci étant toujours de caractère simple et intuitif.

Variabilité dans les conditions d'enregistrement : le système devra pouvoir s'adapter aux variations du contraste, balance des couleurs, bruit, ... C'est-à-dire, qu'il ne doit pas exister de fortes restrictions sur les conditions d'enregistrement.

8.1.2 Les Hypothèses

Sous ces contraintes, il est licite (même conseillé) d'inclure un maximum d'heuristique dans les algorithmes afin d'augmenter leur efficacité. Ainsi, toute connaissance a priori pouvant simplifier la tâche sera prise en compte. Ceci implique l'acceptation d'une série d'hypothèses logiques :

Particularités de la vidéoconférence : dans le cadre de cette application il existe de nombreuses particularités en ce qui concerne le type de scènes à traiter. Ainsi, lors d'une vidéoconférence, il est raisonnable d'espérer que le locuteur se tiendra droit face à la caméra sans faire de mouvements brusques. Par ailleurs, ni l'encadrement ni la distance à la caméra ne sont connus d'avance.

Particularités du personnage en tant que objet suivi : de la même façon, la détection du locuteur pourra s'appuyer sur une sorte de « portrait robot » , pour lequel on peut supposer connues le chromaticité de la peau, les proportions du visage, ...

8.2 Description du projet d'études

Nos contributions au projet M4M iront de l'étude de viabilité des algorithmes potentiels jusqu'à la mise en œuvre du système en étroite collaboration avec le groupe de développement. Ceci permettra de présenter à la fin du projet un démonstrateur logiciel tournant en temps réel sur un ordinateur portable (pentium III à 500 MHz). L'adjonction d'une caméra permettra ainsi de tester nos algorithmes dans des conditions d'enregistrement réalistes (cf. Chapitre 11).

Du point de vue algorithmique, le système doit être capable de produire une *séquence de masques* détourant le locuteur. Pour cela nous allons procéder en deux étapes : premièrement une phase d'initialisation aura en charge la détection du personnage et la création du masque initial ; ensuite un processus récursif fera le suivi tout au long de la séquence.

Sans rentrer dans les détails techniques, présentons d'abord les grandes lignes des algorithmes qui ont été implémentés.

8.2.1 La procédure d'initialisation

Nous allons intégrer dans le démonstrateur un algorithme d'initialisation ayant pour but la détection et segmentation du locuteur sur les premières images de la séquence (cf. Chapitre 9).

Les méthodes d'initialisation habituellement utilisées dans des applications de vidéoconférence sont très contraignantes, car on a besoin de procéder à une double acquisition : la première de l'image du fond sans le personnage, la deuxième avec lui. Nous avons préféré offrir une plus grande souplesse, en perturbant le moins possible l'enregistrement de la scène. Ainsi, notre approche repose sur une segmentation automatique qui démarre suite à un simple clic de souris sur la zone du visage à détecter. Cette interaction avec l'utilisateur nous permet d'adapter le système de détection aux conditions courantes d'éclairage.

8.2.2 La procédure de suivi

La procédure de suivi (cf. Chapitre 10) comporte la segmentation automatique du personnage, ainsi que la procédure interactive de contrôle d'erreurs.

La segmentation automatique du personnage

Nous abordons ici l'étude de viabilité de l'ensemble des outils de filtrage, segmentation et mise en correspondance développés dans un sens plus large au cours de cette thèse. La contrainte du temps réel exerce un lourd poids sur eux : le nombre d'opérations doit être minimisé, leur complexité aussi. Par ailleurs, la fiabilité des algorithmes doit aussi être éprouvée, car on ne pourra pas revenir en arrière en cas d'échec.

Ceci nous obligera à concevoir une chaîne de traitement simple mais solide, ayant recours à des techniques plus testées au détriment des techniques en cours de développement qui, bien que pouvant être plus performantes, ne peuvent pas tourner en temps réel.

Contrôle interactif de la qualité du suivi

Dans des conditions réelles d'enregistrement, il peuvent arriver toutes sortes d'accidents : quelqu'un passe devant la caméra, la lumière change, ... Pour de multiples raisons, l'algorithme de suivi peut perdre le fil, ce qui va se traduire par l'obtention d'un masque erroné : il perd le locuteur en partie ou en totalité, ou bien au contraire il envahit une partie du fond. Pour conférer au système une capacité de prompt rétablissement lorsque ce genre d'accident se produit, nous donnerons à l'utilisateur la possibilité d'interaction tout au long de la séquence. Ainsi, à l'aide de la souris, il pourra corriger facilement des erreurs de segmentation sans que le système de suivi ne s'arrête.

8.3 Conclusions

Ce chapitre, introduisant la dernière partie de cette thèse, nous a servi pour présenter l'ensemble des travaux qui ont été développés dans le cadre du projet européen M4M, la réalisation d'un démonstrateur mettant en valeur la viabilité des techniques implémentées.

Nous verrons dans les chapitres qui suivent comment la mise en œuvre d'une application réelle oblige à prendre des choix parfois douloureux du point de vue du chercheur. Ainsi, des approches plus élégantes devront laisser leur place à des approches plus efficaces, celle-ci étant une nouvelle priorité.

Chapitre 9

Procédure d'Initialisation : la détection du locuteur

Dans ce chapitre nous allons présenter la procédure d'initialisation qui a été proposée pour détecter et segmenter automatiquement le locuteur au début d'une séquence de vidéophonie. Un bref survol des différentes techniques de segmentation, automatiques et semi-automatiques, nous permettra de faire le point sur leurs performances vis-à-vis de cette application. Parallèlement, nous procéderons à l'analyse détaillée de tous les indices qui, étant connus à priori, peuvent aider à la détection. Sur de solides bases, nous bâtirons ensuite une procédure d'initialisation établissant un lien étroit entre les algorithmes de segmentation automatiques et l'analyse heuristique. Finalement, l'étude des résultats obtenus permettra d'évaluer les performances obtenues ainsi que d'établir les limites de cette technique.

9.1 Différents scénarios : différentes techniques

En guise d'étude préliminaire, nous allons aborder le problème de l'initialisation en faisant un bref rappel des techniques de segmentation et de leurs domaines classiques d'application. Le problème de la segmentation étant de nature très variée, la conception d'un système universel et complètement automatique devient impossible. Ainsi, les systèmes purement automatiques ne pourront donner une solution qu'à des applications se déroulant dans des environnements contrôlés, l'interactivité étant de plus en plus nécessaire lorsque le domaine d'application s'élargit.

9.1.1 Segmentation automatique

Les méthodes de segmentation automatiques sont conçues pour donner des solutions à des problèmes très spécifiques, afin de pouvoir extraire par elles-mêmes des régions avec une valeur sémantique. Nous allons différencier celles qui se basent sur,

Modélisation de la scène : la plus simple des techniques de segmentation concerne le détourage d'un objet qui se trouve dans l'avant plan d'un fond connu (Marixal dans [52]). Ces approches sont généralistes pour ce qui concerne l'objet à segmenter, car la détection est faite par simple comparaison avec le fond de référence. Ceci s'utilise couramment pour composer les scènes télévisées au moyen d'un fond lisse de chromaticité distinctive sur lequel le présentateur évolue.

Analyse du mouvement : sous cette rubrique nous avons englobé l'ensemble des techniques qui détectent des objets à partir de l'analyse de mouvement. Certaines approches considèrent que toute région en mouvement capte l'intérêt du spectateur (Meier et Ngan dans [59]), d'autres plus sophistiquées se basent sur l'étude de l'ordre de la profondeur pour proposer comme régions d'intérêt celles qui se trouvent dans l'avant-plan (Bergen dans [8]).

Modélisation de l'objet : par dualité avec la méthode précédente, ces techniques vont baser la détection sur la connaissance de l'objet d'intérêt. Il s'agit donc de créer un modèle de l'objet le caractérisant au niveau des couleurs, textures, forme, taille, etc. Par la suite, la segmentation va extraire les régions s'adaptant au modèle défini. Notons donc que ces approches sont très spécifiques pour ce qui concerne les objets, mais généralistes pour ce qui concerne leur contexte. La modélisation est utilisée couramment dans le milieu biomédical [24] ainsi que pour la détection de cibles [95].

9.1.2 Segmentation semi-automatique

L'interactivité homme-machine se trouve à la base d'un groupe de techniques de segmentation dites *interactives* ou *semi-automatiques*. On les retrouve dans des boîtes à outils extrêmement performantes dans le cadre des applications multimédia, pour l'édition d'images fixes ou de séquences. En reprenant la classification faite par Marqués et al. dans [53], ces approches peuvent s'ordonner par rapport au niveau d'interaction qu'elles proposent à l'utilisateur. Ainsi, on retrouve l'interaction,

Au niveau des pixels : le processus de segmentation se déclenche dès qu'un ou plusieurs pixels sont sélectionnés, ce qui permet à l'algorithme d'extraire automatiquement un vecteur des caractéristiques locales autour de ces points (Chalom dans [19]). A ce stade, les systèmes les plus simples, limités strictement au niveau du pixel, proposent comme segmentation le résultat d'une classification binaire des autres pixels de l'image. Néanmoins, il existe aussi des systèmes mixtes, faisant la transition entre ceux qui sont restreints au niveau du pixel et ceux qui travaillent au niveau des régions, qui proposent comme masque la composante connexe qui inclue le point sélectionné au départ.

Au niveau des régions : nous retrouvons ici des algorithmes qui travaillent sur une partition de l'espace, ou même sur une hiérarchie de partitions comme font ceux de Zanoguera et al. dans [108]. Le système crée de façon automatique une partition de l'espace et attend ensuite que l'utilisateur compose l'objet d'intérêt par sélection des régions.

Au niveau des contours : finalement il existe un autre type de techniques agissant au niveau des contours des objets. Les plus connues sont les *contours actifs*, qui interpolent un contour à partir d'un ensemble des points sélectionnés par l'utilisateur. Parmi d'autres approches, mentionner celle de Falçao dans [25] qui adapte la tracée de l'utilisateur à la ligne de gradient la plus forte.

9.1.3 Choix d'une stratégie

Notre tâche ici revient à la conception d'une procédure d'initialisation capable de détecter et segmenter le locuteur au début d'une séquence de vidéoconférence.

A partir du bref survol que nous avons fait sur l'ensemble des techniques qui peuvent donner une solution à ce problème, le choix d'une stratégie s'impose. Néanmoins, nous devrons faire un choix entre deux solutions extrêmes :

- La segmentation automatique est la technique la moins perturbante du point de vue de l'utilisateur, étant donné qu'aucune interaction ne lui est demandée. La détection est faite à partir de l'information disponible a priori. Cependant, la contrainte du temps réel limite le nombre et la complexité des pistes qui peuvent s'exploiter, rendant les résultats peu fiables dans des environnements non contrôlés.

- La segmentation interactive a un taux d'erreur très faible. Dans la mesure où ces techniques confient la détection des objets à l'utilisateur, les seules erreurs ont lieu sur des régions où notre intuition perçoit un objet mais l'image au niveau des pixels ne permet pas de faire la différence. Comme seul inconvénient on doit remarquer la durée du temps d'interaction, qui peut varier fortement en fonction de l'habileté de l'utilisateur ou de la complexité des images à traiter.

Ainsi, il existe un compromis entre le degré d'automaticité, la robustesse de la procédure et le temps requis pour faire l'initialisation. Notre objectif immédiat est donc d'analyser les indices disponibles a priori avant de pouvoir choisir une procédure d'initialisation entièrement automatique, ou bien avec un certain degré d'interaction.

9.2 Analyse des indices connus a priori

L'analyse des indices connus a priori peut nous servir de point de repère dans un algorithme de segmentation automatique. Pour cela, nous avons entrepris deux études : l'une concerne la caractérisation chromatique de la peau, l'autre la caractérisation de la forme et les proportions du corps. Dans les deux cas, il s'agit d'étudier la validité d'un modèle, chromatique ou géométrique, comme élément de support à la détection du personnage.

9.2.1 Caractérisation chromatique

La couleur d'un objet est l'une des pistes les plus utilisées dans la vision par ordinateur, car elle est indépendante du point de vue. De même, afin de diminuer l'influence des conditions d'éclairage, l'étude de la couleur est restreinte normalement au plan chromatique. Un vecteur couleur tridimensionnel du type (r, g, b) est décomposé dans une composante de luminance plus un vecteur chromatique bidimensionnel. Notamment, l'espace couleur choisi pour faire une telle décomposition va déterminer les caractéristiques de la distribution spectrale, et par conséquent, la performance du modèle mathématique.

Etant donné que nos algorithmes font partie d'un système de codage, nous nous sommes intéressés à l'espace YUV, standard du vidéo digital. Bien qu'il puisse avoir des espaces plus discriminants, comme l'espace TSL proposé par Terrillon dans [96], c'est l'usage de l'espace YUV qui entraîne le moindre coût de calcul.

Pour cela, nous avons confectionné une base de données d'images représentant les quatre groupes ethniques principaux (blanc, asiatique, indien et noir). Dans leur ensemble, ces images présentent un large éventail des teintes de peau et des conditions d'éclairage que nous pouvons retrouver plus tard dans nos séquences. Sur ces images, nous avons étudié deux points : premièrement nous avons interrogé le caractère discriminante de la pigmentation entre les différentes ethnies, deuxièmement nous avons étudié les effets de l'éclairage sur la perception de la couleur.

9.2.1.1 La pigmentation de la peau

Les groupes ethniques se différencient les uns des autres par une série de caractères physiques héréditaires, et notamment par la pigmentation de leur peau. La question sous-jacente est donc de savoir quelle est l'influence de la pigmentation sur le spectre chromatique de la peau. Devions-nous concevoir un modèle adapté à chaque ethnie ? Est-il possible de les modéliser comme un seul ensemble ? La recherche d'une réponse à ces questions constitue l'objectif de cette étude.

De façon préalable, chaque image de la base de données a été segmentée manuellement, afin de créer un masque précis qui couvre la totalité des zones de peau. Puis, on a procédé au calcul de la distribution chromatique de l'ensemble des échantillons.

La Figure 9.1 illustre les résultats obtenus, en utilisant une image de la base de données pour symboliser chacun des quatre groupes ethniques considérés.

Sur la colonne centrale on peut observer la distribution des valeurs de luminance. Cette composante s'est montrée très sensible aux variations d'éclairage, ainsi qu'aux reflets et aux ombres qui apparaissent normalement sur les visages. La variance de ces distributions par rapport à une valeur moyenne commune est trop large pour être considéré comme un trait distinctif de la peau humaine. Par contre, la situation est bien différente pour ce qui concerne les distributions chromatiques qui se montrent sur la colonne de droite. On observe pour elles de fortes ressemblances : tous les spectres sont compacts, de forme plus ou moins elliptique et orientés vers la teinte rouge (voir détails sur le plan chromatique de l'espace YUV sur la Figure 2.4, page 12).

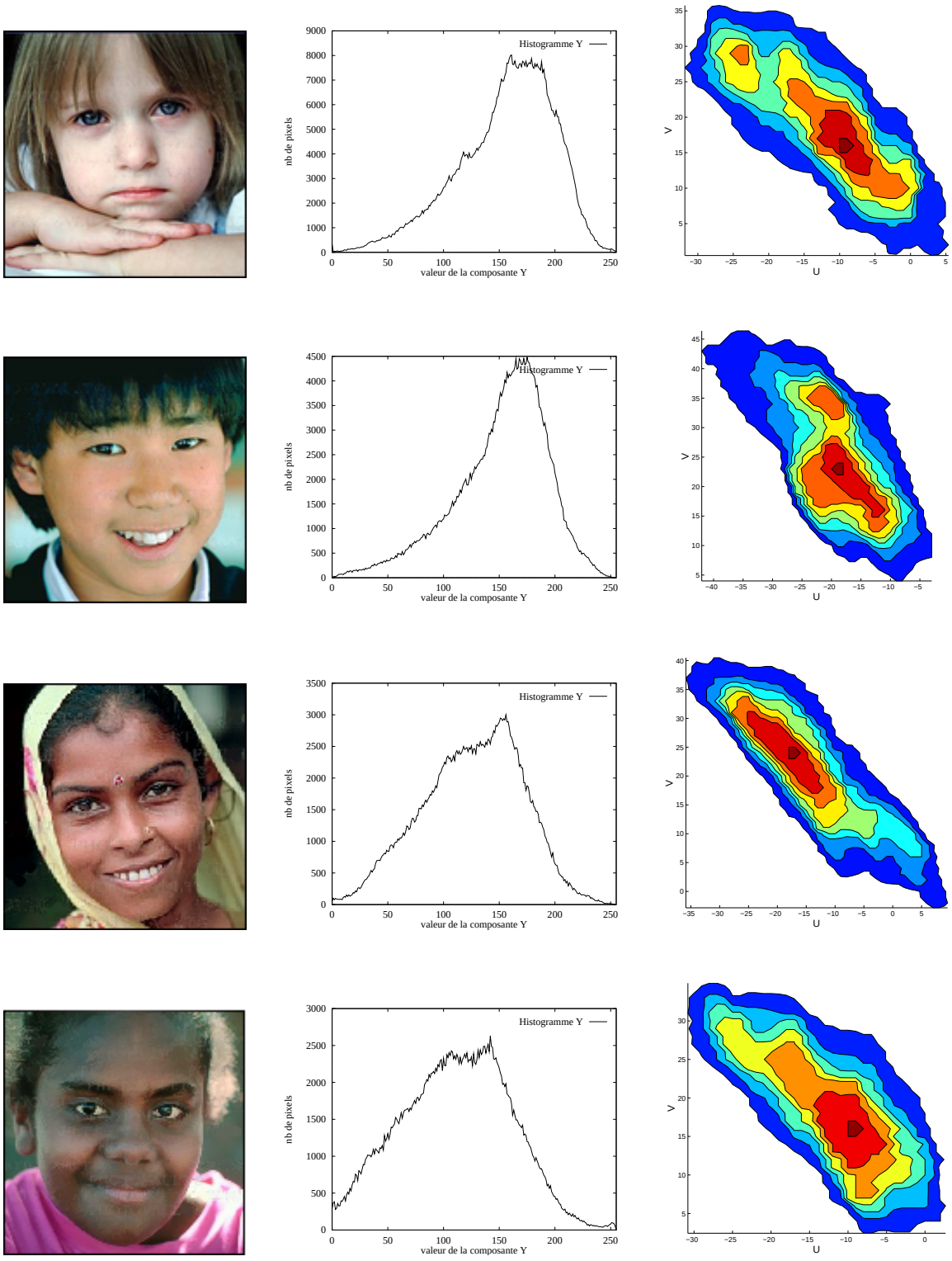


FIG. 9.1 – Caractérisation dans l'espace YUV de la couleur de la peau des différentes ethnies.

Les statistiques de premier ordre calculées à partir de ces distributions chromatiques sont indiquées dans le Tableau 9.1. On observe ici clairement comment le centre chromatique (μ_u, μ_v) pour les races noire et blanche est plus proche du zéro que celui des groupes asiatique et indienne. Cela s'explique puisque les individus de provenance asiatique et indien ont des pigmentations beaucoup plus saturées. Cependant, ces petits écarts ne constituent pas une barrière infranchissable pour la modélisation conjointe de tous les échantillons.

Race	# pixels	μ_u	μ_v	c_{uu}	c_{vv}	c_{uv}
blanche	315.618	$-12,20$	$19,77$	$65,40$	$75,74$	$-47,56$
asiatique	308.663	$-19,46$	$25,80$	$64,67$	$110,51$	$-57,19$
indienne	317.973	$-14,50$	$22,97$	$104,16$	$135,41$	$-92,83$
noire	315.610	$-12,61$	$17,11$	$139,23$	$182,78$	$-133,38$
	1.257.864	$-14,31$	$20,92$	$93,22$	$118,62$	$-77,91$

TAB. 9.1 – Statistiques de première ordre calculées sur des échantillons chromatiques des principaux groupes ethniques.

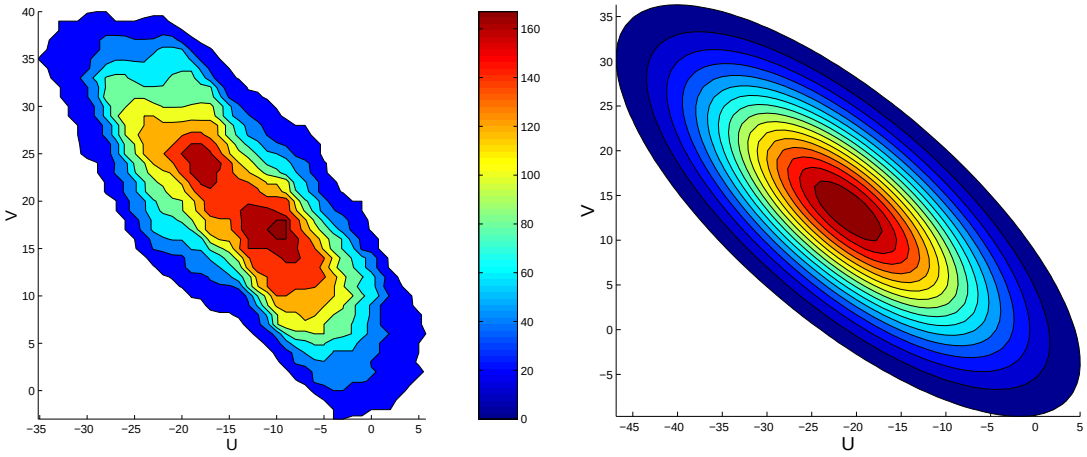
À la suite de ces résultats, nous ne considérerons qu'une seule classe statistique $\mathcal{S}$ pour modeler de façon conjointe la chromaticité de tous les échantillons de peau disponibles dans notre base de données. Ceci permet d'envisager la construction d'un filtre chromatique générique capable de détecter les régions de peau sans devoir spécifier préalablement l'ethnie. Le spectre chromatique ainsi généralisé est montré dans la Figure 9.2 (a). A partir de cet ensemble de données, deux modèles probabilistes sont valables :

Modélisation dans le plan UV

Comme il a été proposé dans [83], la distribution chromatique caractérisant la peau humaine peut être modelée par une fonction de distribution gaussienne bidimensionnelle définie par :

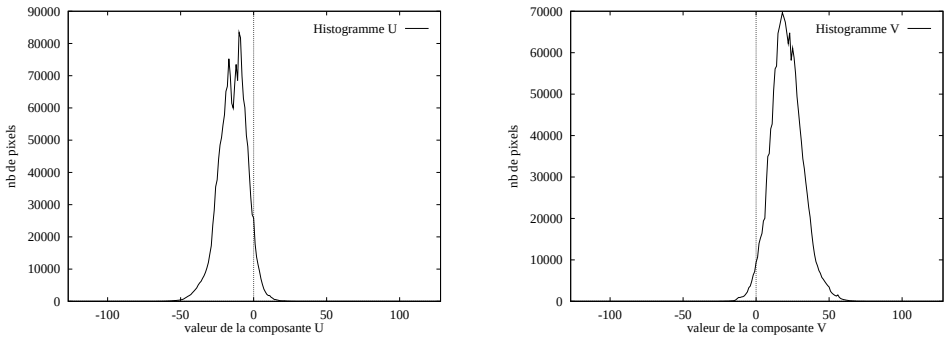
$$P(w|\mathcal{S}) = \frac{exp[\frac{-1}{2}(w-\mu_S)^T {C_S}^{-1}(w-\mu_S)]}{2\pi \mid C_S \mid^{\frac{1}{2}}} \tag{9.1}$$

où $P(w|\mathcal{S})$ représente la probabilité d'un vecteur chromatique $w = (w_u, w_v)$ d'être membre de la classe $\mathcal{S}$. Le vecteur de moyennes $\mu_S = (\mu_u, \mu_v)$ ainsi que les valeurs de la matrice de covariances C_S correspondent à ceux en bas du Tableau 9.1. Du point de vue géométrique, la fonction $P(w|\mathcal{S})$ génère des courbes de niveau de forme elliptique sur le plan chromatique. Le centre est déterminé par le vecteur de moyennes et l'élongation par la matrice de covariances (voir Figure 9.2 (b)). Car, tous les points appartenant à une même courbe sont équiprobables, il existera une plus grande sensibilité vers les déviations de teinte (sur le petit axe de l'ellipse), que vers celles de la saturation (parallèles au grand axe).

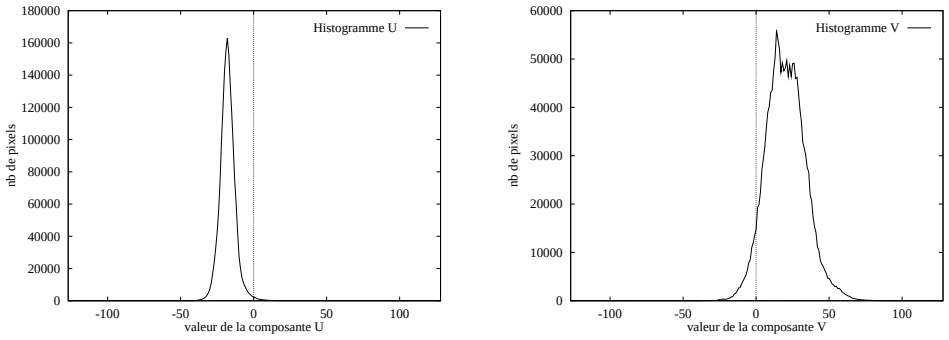


(a) (b)

FIG. 9.2 – Distribution chromatique de la peau humaine dans l'espace YUV : (a) histogramme bidimensionnel obtenu expérimentalement ; (b) modèle gaussien calculé au moyen d'une fonction de distribution de probabilité conjointe.



(a) Spectre original.



(b) Spectre orienté.

FIG. 9.3 – Distribution chromatique monodimensionnelle des composantes U et V.

Modélisation composante par composante

Dans le but de réduire la complexité des calculs, l'algorithme procéderait ici en deux étapes. Dans un premier temps il s'agit d'approximer les histogrammes de chacune des deux composantes chromatiques (voir Figure 9.3 (a)) par de fonctions de distribution gaussiennes. Ensuite l'estimation bidimensionnelle peut être approchée en prenant le minimum de ces deux probabilités,

$$P(w_u|\mathcal{S}) = \frac{1}{\sqrt{2\pi\sigma_u^2}} e^{\frac{-1}{2}\left(\frac{w_u-\mu_u}{\sigma_u}\right)^2} \qquad P(w_v|\mathcal{S}) = \frac{1}{\sqrt{2\pi\sigma_v^2}} e^{\frac{-1}{2}\left(\frac{w_v-\mu_v}{\sigma_v}\right)^2}$$

$$P(w|\mathcal{S}) = min\{\ P(w_u|\mathcal{S}),\ P(w_v|\mathcal{S})\ \} \tag{9.2}$$

Par ailleurs, il est possible d'améliorer la finesse du modèle en alignant par rotation les axes de l'ellipse avec les axes U et V. Sur la Figure 9.3 (b) on peut apprécier la différence entre les deux modélisations. Le nouveau centre du spectre se trouve sur $\mu_S = (-17.32, 24.06)$ et l'écart-type mesuré correspond à $(\sigma_u, \sigma_v) = (9.27, 21.27)$.

9.2.1.2 Effets de l'éclairage : les reflets et les ombres

L'existence d'un modèle chromatique caractérisant la peau étant montrée, étudions à présent les effets de l'éclairage. Car, la source de lumière peut influencer le spectre chromatique associé aux objets éclairés. Pour cela, nous avons pris une série d'images du même individu soumis à des conditions d'éclairage très variées. Ces images, ainsi que la distribution des couleurs sur les visages, sont inclusses dans la Figure 9.4. Visuellement, on aperçoit déjà des différences remarquables au niveau de la couleur, qui va des tonalités rosées jusqu'à celles clairement jaunes. À l'origine de ce phénomène, nous avons trouvé plusieurs facteurs :

Les reflets : la distribution spectrale de la source lumineuse va déterminer la chromaticité des reflets aperçus. La lumière blanche est la seule à préserver intact le spectre de la reflectance de l'objet, par opposition à des éclairages colorés. En fonction des conditions d'éclairage, la peau humaine peut présenter des tonalités très différentes : rosées, rouge-brunes, jaunes et même verdâtres ou bleuâtres. Ceci s'observe clairement sur les images (a,b), et nous pouvons le constater en même temps sur les variations de la moyenne des statistiques du Tableau 9.2. Sur une image saturée ceci devient encore plus critique, car les dérives chromatiques s'éloignent rapidement de la moyenne spectrale.

Les ombres : dans tous les cas où la lumière présente un angle d'incidence uniforme, le spectre chromatique forme un ensemble compact qui peut être modelé par une seule classe statistique. Par contre, lorsque les rayons éclairent obliquement le visage, il se créé deux zones très contrastées : l'une fortement saturée, l'autre complètement dans l'ombre (Figure 9.4 (e)). Cet effet va créer deux types de problèmes : d'un coté les

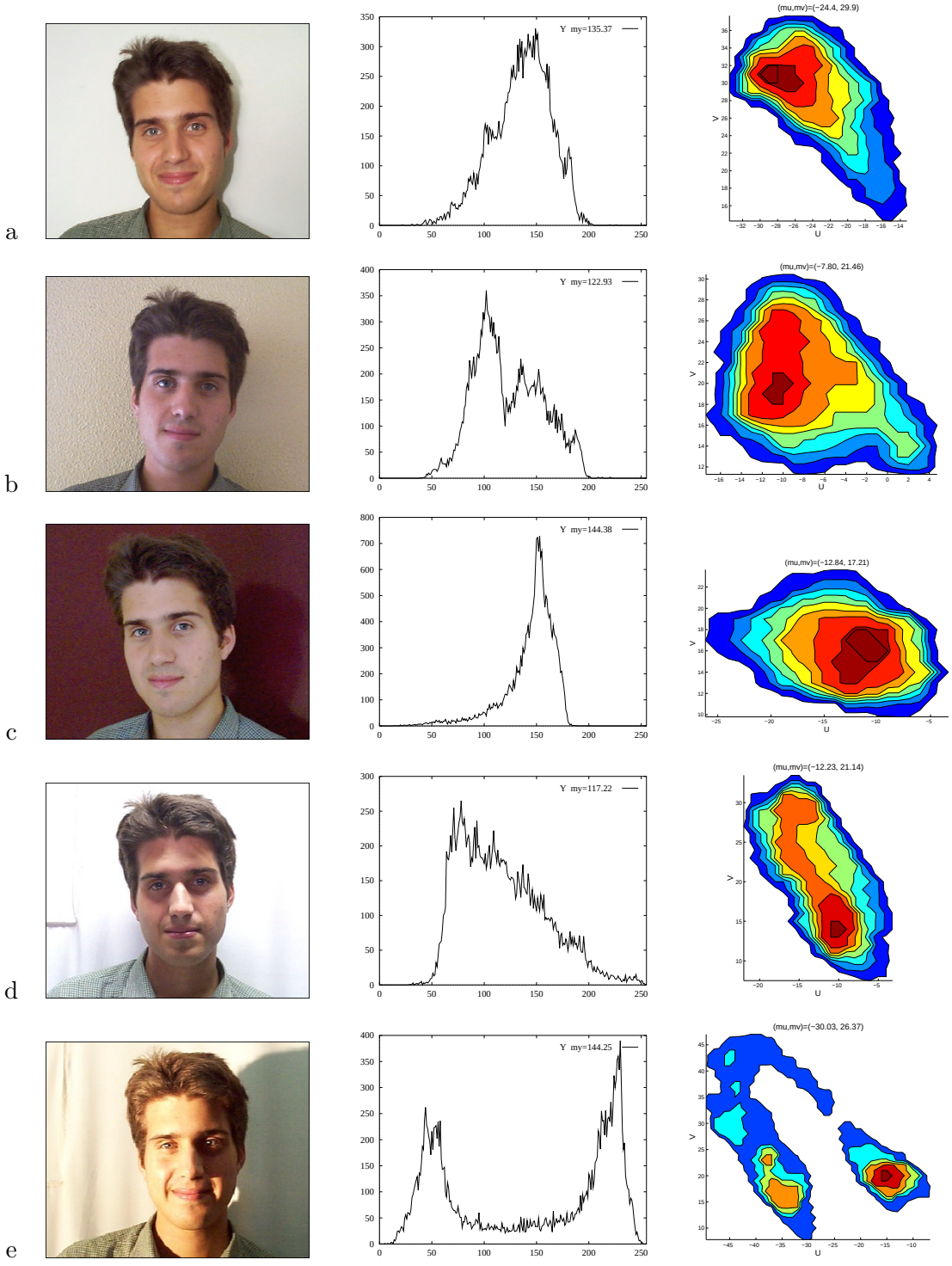


FIG. 9.4 – Effets de l'éclairage sur la perception de la couleur.

ombres provoquent des vecteurs couleurs très peu saturés, qui peuvent basculer facilement sur une autre tonalité à cause du bruit ; de l'autre côté, la modélisation du spectre par une seule classe ne va permettre de détecter qu'une des deux parties du visage, au lieu du visage entier comme on l'aurait souhaité.

Images	# pixels	μ_u	μ_v	c_{uu}	c_{vv}	c_{uv}
a	20.228	-24,43	29,97	18,38	20.86	-11,77
b	20.651	-7,80	21,46	21,44	18.25	-5,67
c	22.779	-12,84	17,21	17,94	7.94	-2,91
d	20.763	-12,23	21,14	15,68	41.28	-16,99
e	21.325	-30,03	26,37	144,01	94.92	-60,67

TAB. 9.2 – Statistiques de première ordre concernant la chromaticité de la peau lorsqu'elle est soumise à différentes conditions d'éclairage.

9.2.2 Caractérisation géométrique

Dans la section précédente nous avons vu comment la caractérisation chromatique de la peau peut aider à la localisation du visage. A présent, nous cherchons des indices équivalents qui peuvent faciliter la détection et la segmentation du corps du personnage. Dans ce sens, la piste de la couleur n'est plus utilisable car nous ne connaissons a priori ni la couleur des cheveux ni celle des vêtements. Le seul modèle générique qui peut être établi concerne l'estimation de leurs contours à partir des proportions du corps.

Le problème de la modélisation du corps est très complexe, d'autant plus que l'aspect physique peut varier fortement d'une personne à une autre : il y a par exemple des personnes grandes et petites, grosses et minces, des visages ronds et allongés, des cheveux longs ou courts, fins ou frisés. Dans le cadre des applications de vidéophonie, il est raisonnable de faire l'hypothèse que le locuteur est en vue frontale face à la caméra. Ceci permet de faire usage d'un modèle 2D au lieu d'avoir recours à des modèles 3D plus complexes et plus coûteux en temps de calcul.

Nous visons donc à approcher la forme de la silhouette. Plus tard, lors de la mise en œuvre de la procédure d'initialisation, les contours réels seront obtenus au moyen d'un algorithme de segmentation.

Modélisation de la silhouette

Nous proposons d'approcher la silhouette d'un personnage à partir des contours d'un modèle géométrique composé par 49 points de contrôle. La Figure 9.5 montre la distribution de ces points et le contour obtenu par des segments droits. Le modèle doit être de type corps entier pour ne pas imposer de contraintes sur la distance du personnage à la caméra. Afin d'avoir plus de précision sur les plans rapprochés, nous avons augmenté la densité des points de contrôle sur la région de la tête et des épaules.

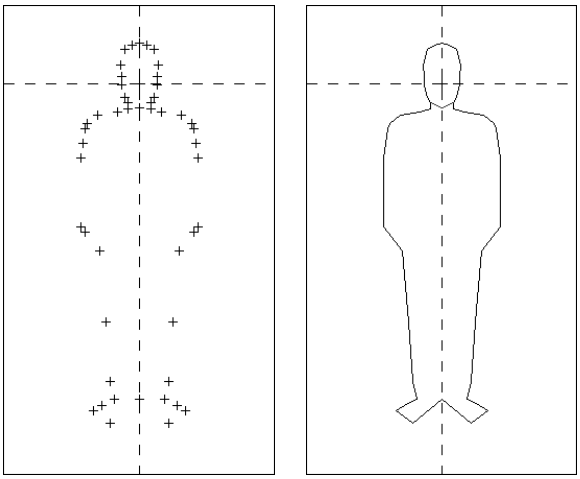


FIG. 9.5 – Modèle de la silhouette humaine construit à partir de 49 points de contrôle.

Pour adapter le modèle au personnage il suffit de connaître : les coordonnées du centre du visage (mc_x, mc_y), ses proportions (a,b)[1] et son angle d'inclination (θ). Ensuite, par *transformation affine* le modèle se place dans le cadre de l'image à l'échelle appropriée :

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x_m \\ y_m \end{pmatrix} + \begin{pmatrix} mc_x \\ mc_y \end{pmatrix}$$

La Figure 9.6 illustre un exemple d'adaptation. Pour chacune de ces images le centre et la largeur du visage ont été fournis manuellement.

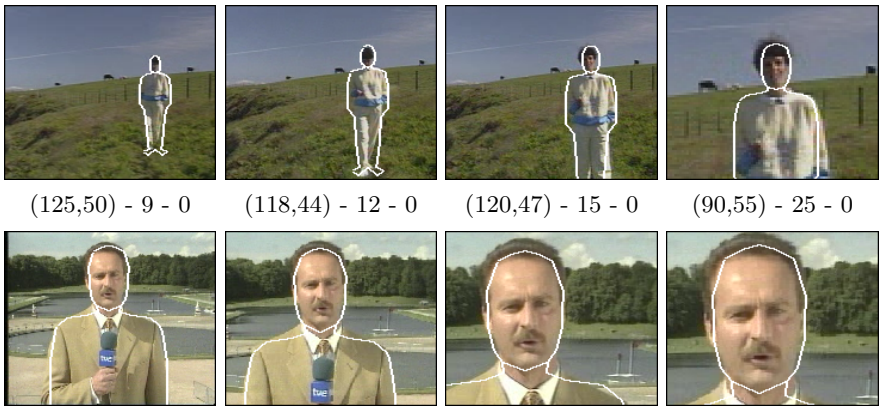


FIG. 9.6 – Adaptation du modèle au contour d'un personnage par transformation affine. Paramètres du modèle : centre, largeur et angle d'inclination du visage.

[1]Si seulement une des deux valeurs est connue, il est possible d'estimer l'autre à partir des proportions standard $(a \approx 0,7 \cdot b)$.

9.3 Mise en œuvre d'une approche automatique

Nous présentons ici la conception et mise en œuvre d'une procédure d'initialisation complètement automatique, les modèles de couleur et de forme étant à la base du processus de détection. La viabilité finale de l'ensemble des algorithmes reste conditionnée à l'existence d'un compromis entre la robustesse et le temps de calcul. Ainsi, si les résultats obtenus sont qualifiés de satisfaisants, l'importance et la complexité de chacune des opérations sera révisée en vue de leur implémentation dans le démonstrateur.

Le schéma global de la procédure d'initialisation est indiqué dans la Figure 9.7, où les images ajoutées illustrent les données disponibles à l'entré et à la sortie des blocs. La détection du locuteur sur la première image de la séquence est faite en deux étapes : la première vise à la *détection du visage*, la deuxième à la *segmentation du corps*. L'ensemble des opérations menées à terme sont détaillées dans les sections qui suivent.

9.3.1 Détection du visage

Le problème de la détection du visage a fait l'objet de nombreux papiers dans la littérature. Chellapa dans [20] fait un état de l'art détaillé des techniques existantes. Dans ce domaine, les études les plus récentes convergent vers l'usage généralisé des informations de couleur et de forme connues a priori. Dans la ligne de ces derniers travaux [56, 2, 1], nous proposons d'augmenter la robustesse de la détection,

- en incluant une étape d'adaptation des statistiques de la classe peau aux conditions locales ; et,
- en travaillant sur une hiérarchie de partitions de l'image de probabilités lors de la recherche de la région du visage.

Voyons ceci plus en détail.

Calcul des probabilités par adaptation des statistiques aux conditions locales

Les analyses précédentes ont montré qu'il est possible de modeler le sous-espace couleur qui caractérise la chromaticité de la peau. L'objectif final de ce calcul de probabilités est de nous permettre de séparer les régions qui correspondent à des zones de peau des autres. Dans des environnements contrôlés, l'extraction d'un masque binaire couvrant les zones de peau peut être fait par simple application d'un seuil. Néanmoins, dans des cas plus complexes où l'éclairage rend douteuse la classification des pixels, cette approche n'est pas suffisamment discriminante.

Pour faire face à ce problème, nous permettrons au système de s'adapter aux conditions locales de l'image par itération des étapes suivantes, jusqu'à la stabilisation des valeurs moyennes des statistiques :

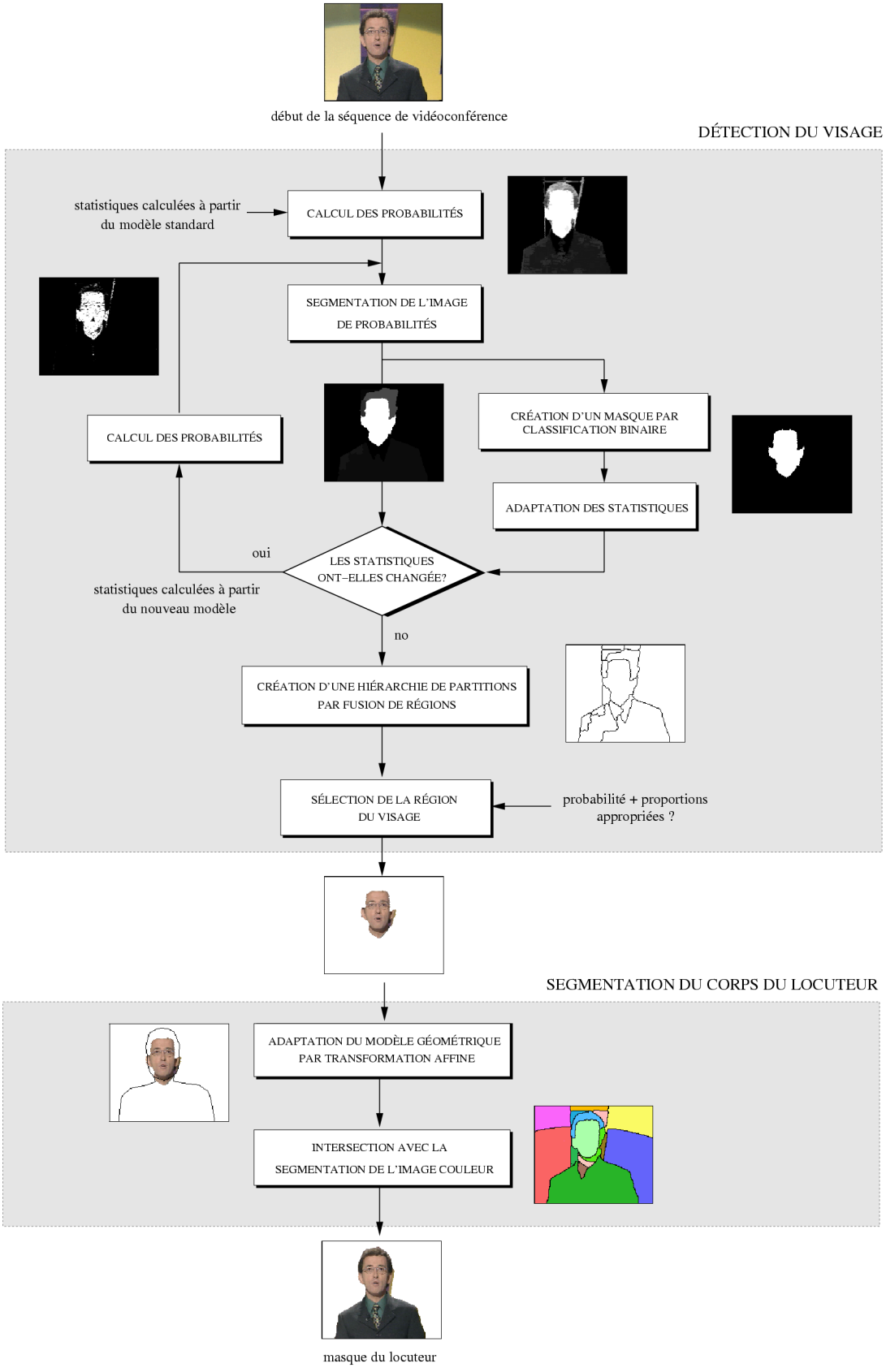


FIG. 9.7 – Schéma de la procédure d'initialisation automatique.

1.- Calcul des probabilités : dans un premier temps, l'algorithme estime la probabilité de chaque pixel d'appartenir à la classe statistique de la peau. Nous appliquerons ici le modèle de distribution gaussien bidimensionnel tel qu'il a été formulé dans la Section 9.2.1.1. Au cours de la première itération, en l'absence de connaissances a priori sur les conditions d'enregistrement, le système fait usage des statistiques calculées sur les images de la base de données.

2.- Segmentation de l'image de probabilités : afin de rendre l'analyse des probabilités plus robuste face au bruit et aux dégradations liées aux processus de codage, nous allons étendre le support d'analyse du pixel à la région. Pour cela, nous segmentons l'image de probabilités conformément à la méthodologie décrite dans le Chapitre 4 et rappelée brièvement ici :

- *Simplification*
- *Calcul du gradient morphologique*
- *Extraction de marqueurs à partir des minima du gradient*
- *Inondation de l'image gradient à partir des marqueurs*

Par contre, cette fois-ci la partition résultante n'est pas complètement satisfaisante : bien que les contours séparent des zones de probabilité différente, ceux-ci sont très irréguliers. Pour améliorer le comportement du système sur ce point, nous proposons d'extraire les marqueurs à partir des zones de probabilité homogène, et d'utiliser un gradient couleur pour l'inondation, celui-ci étant beaucoup plus robuste car il inclut de l'information de luminance. Notons que cette partition ne vise pas à contourner tous les objets de la scène, mais à les séparer conformément à leur probabilité d'être une partie de la peau.

3.- Création d'un masque par classification binaire des probabilités : nous procédons dans cette étape à la création d'un masque de peau par classification binaire de l'histogramme des probabilités moyennes des régions de la partition. Néanmoins, à cause des conditions d'éclairage changeantes, le seuil fixe qui permettrait de détecter la peau dans certaines images serait trop restrictif dans d'autres. Il parait donc essentiel de parvenir au calcul d'un seuil adapté au rang des valeurs de l'histogramme des probabilités. Compte tenu de ceci, nous allons placer le seuil entre les deux pics de l'histogramme qui ont la séparation la plus large.

4.- Adaptation des statistiques : sous des conditions d'éclairage particulières, il se peut que l'ensemble des tonalités de la peau que nous voudrions détecter ne soit pas complètement inclus dans le sous-espace modélisé. De ce fait, le masque dont nous disposons à présent ne contienne qu'une partie de la peau visible dans l'image. Pour faire face à cet inconvénient, nous allons calculer un nouveau modèle par adaptation des statistiques à l'image courante. Ceci est fait en trois étapes : premièrement on procède au calcul des nouvelles statistiques à partir des échantillons à l'intérieur du masque ; ensuite on obtient l'image de probabilités correspondante ; et finalement, on fait la moyenne entre cette nouvelle image et celle qui précède l'étape d'adaptation. Cette pondération permet de procéder à une adaptation progressive du

modèle générique.

L'ensemble de ces opérations est itéré jusqu'à ce que les moyennes statistiques se stabilisent. Cela va nous permettre de faire croître l'écart entre les régions du fond ayant une couleur proche à celle de la peau humaine, et celles qui correspondent vraiment à des régions de peau.

Afin de pouvoir évaluer les performances de cette méthode, nous allons suivre le processus d'adaptation sur deux exemples : dans la Figure 9.8 nous avons pris une image dont le fond présente une probabilité faible, mais pas nulle, d'être de la peau ; par comparaison, dans la Figure 9.9 nous avons pris une image dont l'éclairage sur le visage provoque des probabilités en dessous de la moyenne.

En haut des deux figures on peut observer l'image couleur originale. La suite des colonnes résume les différentes itérations du processus d'adaptation, et chaque ligne illustre une des étapes de l'algorithme :

(a) Images de probabilités calculées à l'aide des statistiques de la classe peau. Dans la première étape les statistiques correspondent à celles de la base de données, ensuite elles sont calculées par adaptation progressive au contenu de l'image.
(b) Segmentation de l'image de probabilités, sur laquelle on a fusionné toutes les régions inférieures à une certaine taille (90 pixels), n'ayant pas d'intérêt du point de vue de la détection du visage.
(c) Labellisation des régions de la partition par la valeur moyenne des probabilités calculées auparavant. Obtention d'une partition des probabilités qui respecte les contours réels de l'image couleur.
(d) Histogramme de la partition des probabilités. Le seuil de la classification binaire séparant les régions de peau des autres est marqué en ligne pointillée.
(e) Masque créé par une telle classification. Les statistiques du modèle vont s'adapter ensuite à cet échantillon de peau, en moyennant l'image de probabilités ainsi obtenue par celle de l'étape précédente.

Suite à l'adaptation des statistiques à la tonalité de la peau du personnage, la probabilité du visage augmente tant que la probabilité des autres objets diminue (voir la partition résultante dans l'étape #3 - c). Cela crée un contraste maximal entre les zones de peau et le fond, cet effet peut être facilement observé sur la série d'histogrammes (d).

Création d'une hiérarchie de partitions en vue de la détection du visage

Dans les étapes suivantes, l'algorithme doit procéder à la détection du visage à partir de l'information des probabilités disponible sur les régions de la partition. A ce stade, le visage ne serait détecté que s'il apparaissait comme une seule région, et ces cas sont rares. Dans la plupart des cas, il est habituel qu'il soit fractionné en plusieurs morceaux.

Dans le but de donner un meilleur support d'analyse, nous proposons de mener à terme la recherche de la région du visage sur une hiérarchie complète de partitions. Cette hiérarchie est

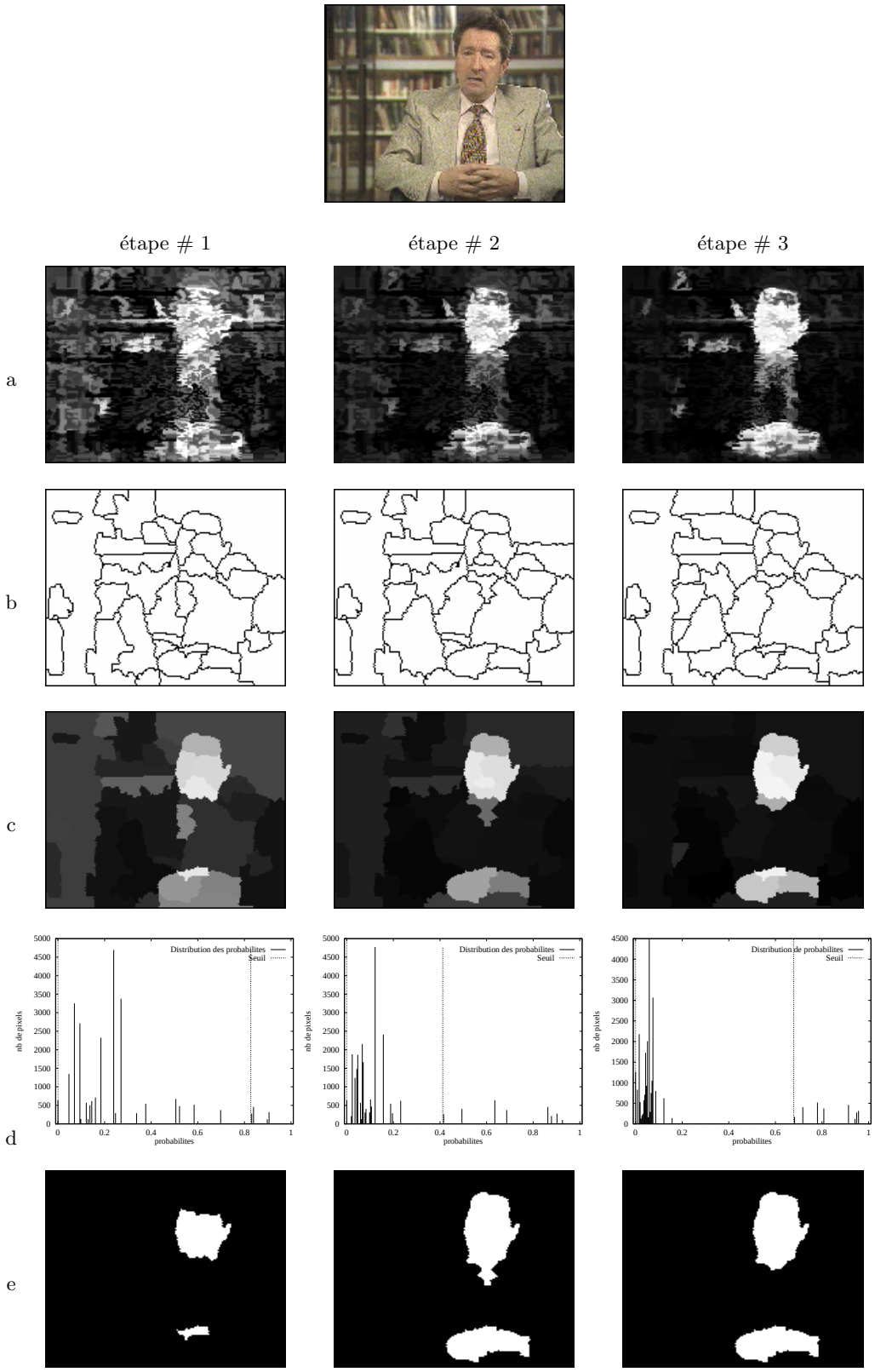


FIG. 9.8 – Processus d'adaptation des statistiques chromatiques sur une image ayant de hautes valeurs de probabilités au départ.

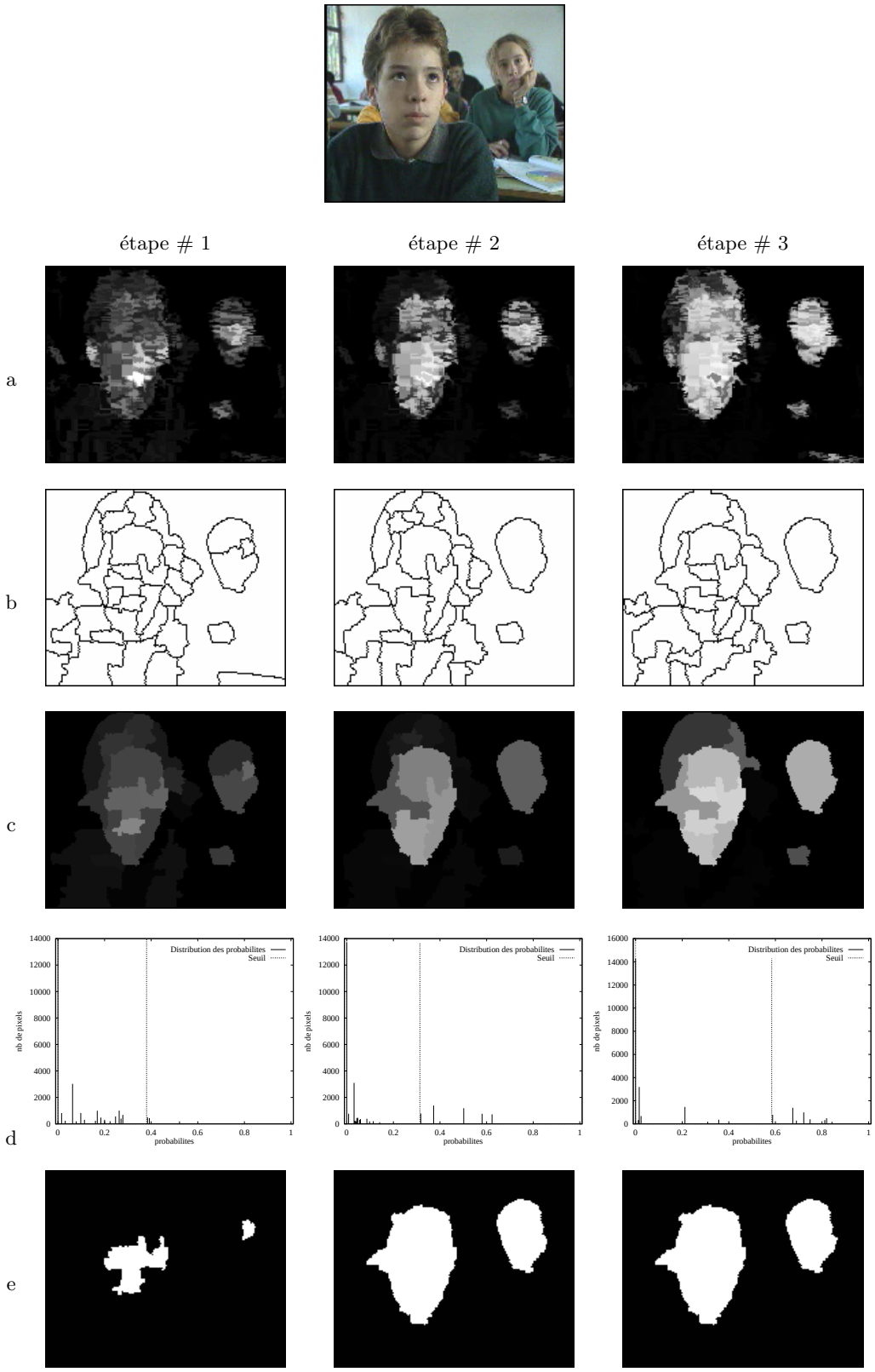


FIG. 9.9 – Processus d'adaptation des statistiques chromatiques sur une image ayant de faibles valeurs de probabilités au départ.

calculée par un processus itératif de fusion de régions. A la base nous retrouvons la partition obtenue lors de l'adaptation du modèle chromatique.

Ensuite, à chaque itération, deux régions fusionnent, donnant lieu à une série de partitions emboîtées de plus en plus grossières. Le critère de fusion est le suivant,

Critère de fusion : à chaque niveau de la hiérarchie disparaît la région de plus petit *volume*. Elle fusionne avec le voisin le plus *similaire* compte tenu de sa couleur et sa probabilité d'être de la peau.

Après chaque fusion, nous procédons à une réévaluation des attributs de la nouvelle région ainsi qu'à la mise à jour des mesures de similarité avec les régions voisines. Lorsqu'aucun critère d'arrêt n'est inclue, le processus finit quand toutes les régions de la partition initiale ont fusionné.

Il nous reste seulement à préciser la mesure de *similarité* entre deux régions et la notion du *volume* d'une région :

- *Calcul de la similarité entre deux régions :* dans le cadre de notre approche, la similarité entre deux régions voisines correspond au produit de deux distances : l'une calculé sur la couleur moyenne des régions, l'autre sur leurs valeurs de probabilité.

- *Calcul du volume d'une région :* dans un espace 3D, la mesure de similarité entre deux régions peut être vue en relation avec la hauteur d'un mur qui les sépare. Ceci permet d'interpréter le produit de la surface de la région par la hauteur du mur le plus bas qui l'entoure comme une mesure de volume.

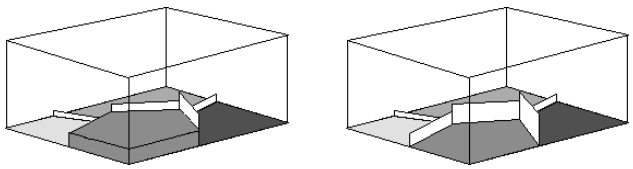


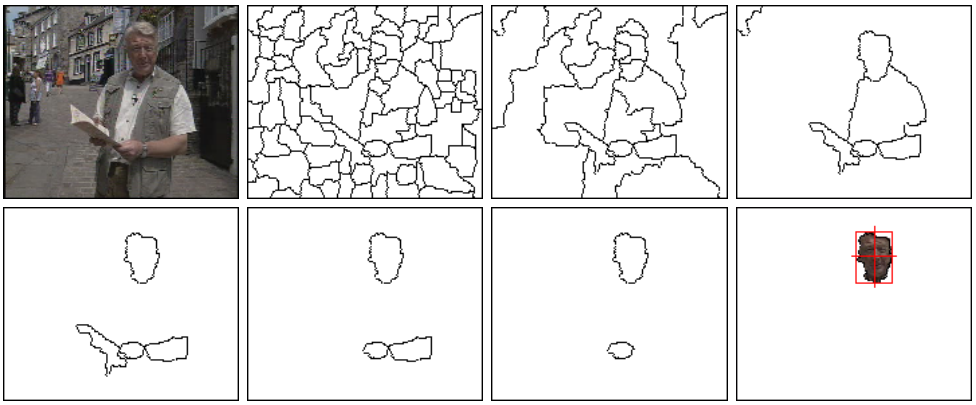
FIG. 9.10 – Calcul du volume d'une région.

En utilisant le volume comme valeur d'extinction, nous faisons disparaître dans les niveaux plus bas de la hiérarchie les petites régions de couleur et probabilité semblable à celles de leur entourage. Ainsi, en haut de la hiérarchie nous retrouvons, si elles existent, les régions de grande taille, avec une forte probabilité d'être de la peau et contrastées par rapport à leur entourage. Cette ordination fournit donc un support idéal pour détecter le visage comme une seule région.

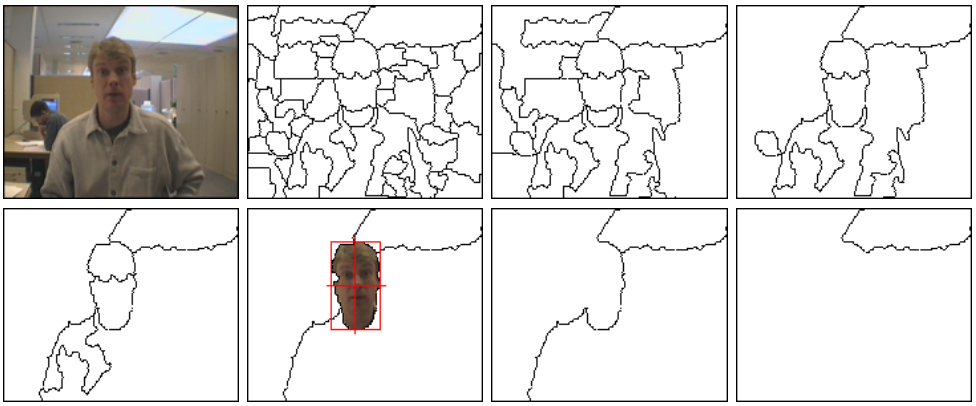
La Figure 9.11 montre deux exemples des hiérarchies ainsi obtenues. A côté des images originales nous avons placé la partition la plus fine qui se trouve à la base de la hiérarchie. Puis, quelques unes des partitions qui en font partie, jusqu'à atteindre celle qui ne contient que les deux dernières régions qui fusionnent.

L'exemple (a) montre un personnage qui n'est pas dans un plan rapproché, son visage apparaît petit et une grand partie de son corps rentre dans le cadre de l'image. Par ailleurs, le visage n'est pas la seule région de peau visible, le bras étant aussi présent. Dans la hiérarchie les petits détails dans le fond fusionnent rapidement. De même pour les différentes parties du visage. A la fin, seulement les régions ayant une forte probabilité d'être de la peau perdurent.

L'exemple en (b) illustre une situation bien différente. La plupart des objets présents dans l'image ont une couleur proche à celle de la peau, exception prêt de la lampe dans le plafond. Ceci fait que le visage fusionne ici avec son entourage sans atteindre le sommet de la hiérarchie. Bien qu'il a la plus forte valeur de probabilité, il est plus petit que d'autres régions et moins contrasté.



(a) Nombre de régions dans les partitions : 82, 22, 7, 5, 4, 3, 2.



(b) Nombre de régions dans les partitions : 55, 20, 10, 6, 4, 3, 2.

FIG. 9.11 – Hiérarchies de partitions créées à partir de l'image couleur et de l'image des probabilités.

Sélection de la région du visage

Pour procéder à l'extraction du visage, il suffit de parcourir la hiérarchie des partitions du haut vers le bas, en cherchant la région que s'adapte le mieux aux critères chromatiques et

géométriques qui caractérisent un visage. Ainsi, pour qu'une région soit classée comme visage elle devra avoir,

- *une valeur de probabilité d'être de la peau au-delà d'un certain seuil,*
- *les proportions d'un visage* ($largeur \approx 0,7 \cdot hauteur$),
- *une surface proche de celle de la plus grande l'ellipse incluse dans boîte minimale qui entoure la région,*
- *le centre de gravité coïncidant, ou presque, avec celui de la boîte minimale qui la contient.*

Dans le cadre de notre application, nous devons obtenir un seul visage comme résultat de la détection. Nous croisions donc à la fin la région candidate la plus large, en supposant que le locuteur est le personnage le plus proche de la caméra. Par ailleurs, certaines contraintes additionnelles peuvent être imposées pour aider le système à détecter le personnage dans des scènes complexes. Par exemple, on pourrait supposer que le personnage se tient droit face à la caméra ($\theta \approx 0$) ou qu'il se trouve dans un rang de distances connu qui permet d'accoter l'échelle d'analyse.

9.3.2 Segmentation du corps du locuteur

Dans le cadre d'une application de vidéoconférence, où le but est de segmenter le locuteur, le masque ne peut pas se restreindre à la zone du visage. Il doit inclure aussi toute autre partie du corps qui est visible. Cette tâche est confiée au deuxième bloc de notre schéma de traitement, ayant comme données à l'entrée l'image couleur originale et le masque du visage. L'algorithme que nous proposons procède en deux étapes :

Adaptation du modèle géométrique par transformation affine

Comme première approche, le système adapte le modèle générique de la silhouette d'un personnage (voir Section 9.2.2) au visage détecté dans l'image. Trois paramètres sont nécessaires pour mener à terme la transformation affine :

- *Proportions du visage :* cette information s'exprime sous la forme d'un vecteur (largeur, hauteur) calculé à partir de la boîte minimale qui englobe la région du visage. Ces valeurs vont déterminer l'échelle du modèle.
- *Coordonnées du centre de gravité :* elles se calculent à partir des moments d'inertie d'ordre zéro et un du masque binaire du visage. Aucune restriction n'est imposée sur la position du visage dans le cadre de l'image.

$$m_{ij} = \sum_x \sum_y x^i y^j f(x,y)$$

$$mc_x = \frac{m_{10}}{m_{00}} \qquad mc_y = \frac{m_{01}}{m_{00}}$$

– *Orientation de l'axe principal de symétrie :* cette valeur détermine l'inclinaison latérale du visage. Typiquement, elle est calculée à partir des moments d'inertie centrés d'ordre deux :

$$\theta = \frac{1}{2}\arctan\left(\frac{2\overline{m}_{11}}{\overline{m}_{20} - \overline{m}_{02}}\right)$$

Cependant, le résultat peut être faussé si le visage a des parties manquantes à cause d'une barbe, des reflets sur les lunettes, ... Pour surmonter cet inconvénient, nous proposons une méthode alternative. Il s'agit de calculer l'orientation du visage comme celle de la boîte de surface minimale qui englobe la région, en vérifiant de façon préalable que les centres de gravité coïncident. Dans le cas contraire, le masque est filtré jusqu'à ce que cette condition soit vérifiée.

L'objectif du modèle obtenu est seulement de trouver les contours approchés du personnage, un algorithme de segmentation étant en charge de retrouver les contours réels.

Intersection avec la segmentation de l'image couleur

Parallèlement à l'adaptation du modèle au masque du visage, le système procède à la segmentation de l'image couleur originale. Le nombre de régions de la partition se calcule automatiquement en fonction de la taille du visage : plus petit est le visage, plus il faudra de finesse sera nécessaire dans la partition pour segmenter les régions du corps, et vice-versa. L'intérêt de cette segmentation est de fournir un découpage de la scène tel que tous les contours du personnage soient présents.

Il nous reste alors à décider pour chacune des régions de la partition si elles font partie du masque du personnage M_p, ou si elles appartiennent au fond. Pour prendre cette décision nous allons créer un masque à partir de la silhouette modelée M_s et calculer l'intersection avec la partition. Ainsi,

– toute région R_i complètement à l'intérieur de M_s fera partie de M_p ;

– toute région R_i complètement à l'extérieur de M_s fera partie du fond ;

– toute région R_i partiellement incluse dans M_s fera partie de M_p si et seulement si le pourcentage de surface à l'intérieur du modèle est au-delà d'un certain seuil,

$$\frac{\text{Surf}(R_i \cap M_s)}{\text{Surf}(R_i)} > s_{min} \quad \Rightarrow \quad R_i \in M_p$$

Ceci étant la dernière étape de l'algorithme d'initialisation, le système rend automatiquement un masque complet du corps du locuteur. Nous étudierons la performance obtenue lors de l'analyse de résultats dans la Section 9.5. Auparavant, nous donnerons quelques détails concernant l'approche qui a été implémentée dans le démonstrateur du projet M4M.

9.4 Mise en œuvre d'une approche interactive

La robustesse des algorithmes étant satisfaisante, plusieurs modifications ont été nécessaires pour les rendre compatibles avec la contrainte de temps réel qui pèse sur cette application.

Certaines étapes ont pu être simplifiées, d'autres ont dû être supprimées à cause de leur temps de calcul. Ceci nous a obligé à changer de stratégie. Ainsi, nous avons transformé un système qui été complètement automatique en une procédure d'initialisation de nature interactive.

Un schéma analogue à celui que nous avons présenté pour l'approche automatique est fourni dans la Figure 9.12. Les nouveautés de cette deuxième procédure sont détaillées par la suite.

La première différence qu'on remarque concerne l'existence d'un bloc parallèle ayant en charge le calcul de la segmentation de l'image d'entrée. Dans cette version de l'algorithme, la même partition est utilisée tout au long du processus d'initialisation, la segmentation de l'image des probabilités étant ainsi supprimée. Pour ce qui concerne le premier des blocs de la procédure, plusieurs changements sont aussi à préciser.

Détection du visage

Cette partie du système a dû être fortement restructurée, donnant lieu à une procédure en six étapes qui s'exécutent de façon séquentielle,

1.- Sélection d'un point du visage : l'algorithme d'initialisation attend une action de l'utilisateur pour commencer, la seule interaction qui lui est demandée étant de cliquer avec la souris sur un point au milieu du visage.

2.- Adaptation des statistiques : une seule étape d'adaptation des statistiques est calculée, car, grâce à l'interaction, nous connaissons depuis le départ, sans possibilité d'erreur, l'endroit où se trouvent les échantillons de peau appartenant au visage. Pour créer le masque nécessaire à l'adaptation, nous allons prendre les régions de la partition qui se trouvent autour du point signalé par l'utilisateur. Ceci nous donne un support suffisamment large pour calculer l'adaptation.

3.- Création d'un masque de peau par classification binaire : l'image de probabilités n'est utilisée ici que pour extraire un masque binaire M séparant les zones de peau du reste. Compte tenu de ceci nous pouvons simplifier fortement les calculs. En traitant séparément chaque composante couleur (voir Section 9.2.1.1), nous considérerons qu'un pixel $w = (y, u, v)$ appartient à la classe de la peau si les valeurs de u et v rentrent dans l'écart-type des courbes gaussiennes respectives du modèle. L'inclusion d'une valeur constante c permet d'ajuster la sensibilité de la détection.

$$\forall w \in I, \quad si \left\{ \begin{array}{l} |u - \mu_u| < c\sigma_u \\ |v - \mu_v| < c\sigma_v \end{array} \right\} \Rightarrow w \in M$$

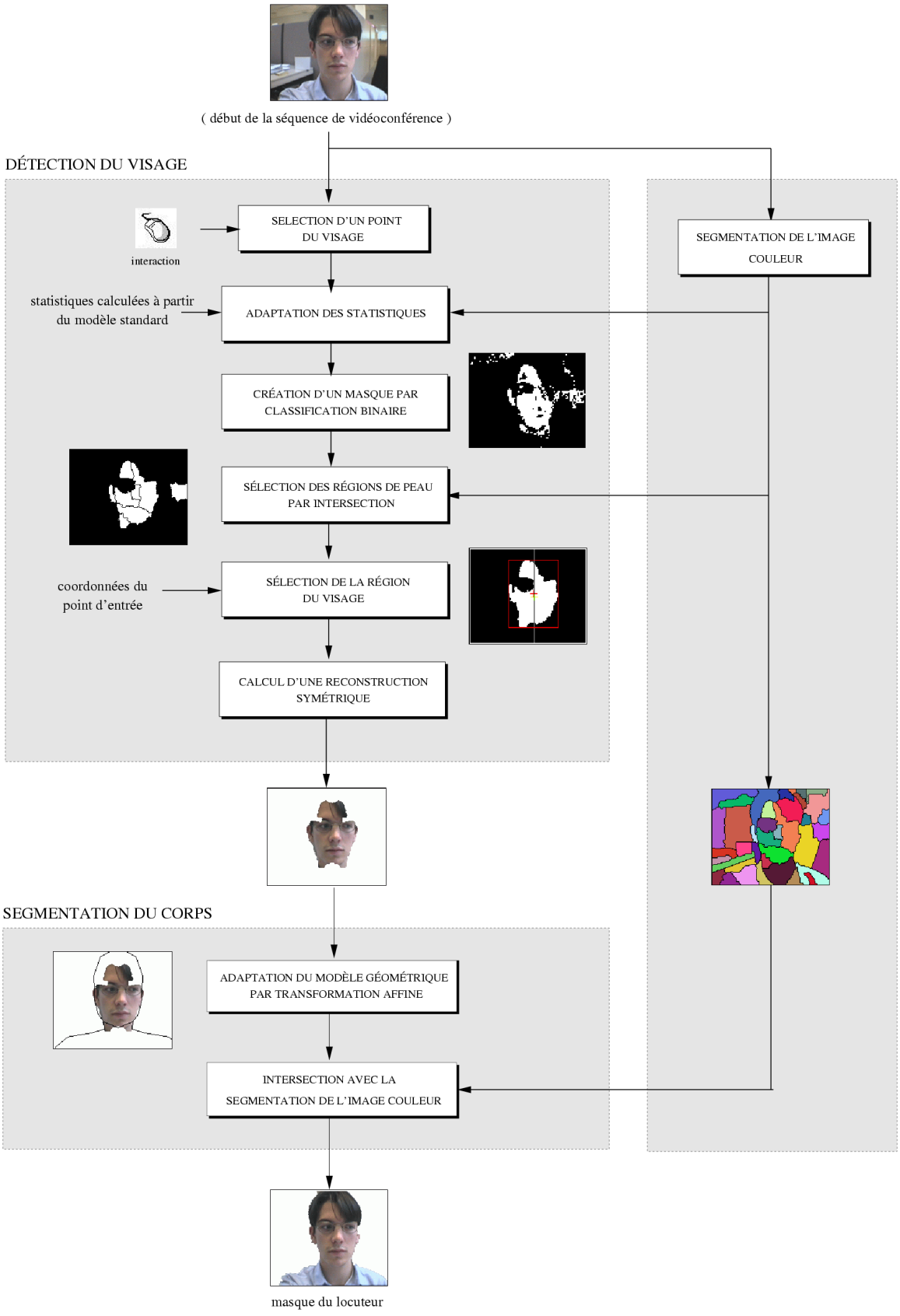


FIG. 9.12 – Schéma de la procédure d'initialisation interactive.

4.- Sélection des régions de peau : le masque créé par classification binaire des pixels est très sensible au bruit et aux petites variations dans la couleur. Couramment, on a des « faux positifs » pour des pixels qui sans être de la peau ont été inclus dans le masque et des « faux négatifs » pour des pixels de peau qui n'ont pas été détectés. Pour rendre l'analyse robuste face à ces petites erreurs, nous proposons de passer du masque au niveau des pixels à un masque au niveau de régions. Ceci est fait par calcul du pourcentage de la surface de chaque région qui est classée comme de la peau.

5.- Sélection de la région du visage : le masque créé par intersection avec la partition peut contenir plusieurs composantes connexes. L'algorithme sélectionne comme visage celle qui contient le point introduit par l'utilisateur.

6.- Calcul d'une reconstruction symétrique : finalement, nous avons inclus une nouvelle étape de complexité réduite, mais efficace dans la correction d'erreurs. Il s'agit de faire une reconstruction des valeurs du masque de chaque côté de l'axe de symétrie. Ceci est possible dans le cadre de notre application sous l'hypothèse que le locuteur est en vue frontale face à la caméra.

La procédure de segmentation du corps du locuteur n'a pas changé car, la partition de l'image couleur étant déjà disponible, il ne reste qu'à adapter le modèle géométrique et à calculer l'intersection.

9.5 Etude de la performance du système

Au cours de cette section nous montrerons un large éventail d'exemples, permettant aux lecteurs d'évaluer par eux-mêmes la performance achevée. Et pourtant, nous n'avons pas choisi des conditions simples. Tout au contraire, nous avons testé les algorithmes dans des conditions réelles et, dans certains cas, même extrêmes. Ceci nous permettra par la suite de formuler des conclusions sur le fonctionnement des algorithmes, où se trouvent leurs limites et quelles sont les améliorations à envisager.

Notons que toutes les images de résultats montrées ici ont été obtenues au moyen de la méthode automatique, qui ne peut qu'être amélioré pour toute forme d'interaction. Les résultats de la méthode interactive vont être toujours égaux ou meilleurs que ceux obtenus automatiquement : la détection du visage est plus aisée (nous disposons d'un point de repère connu sur le visage), et le comportement est le même pour ce qui concerne la segmentation du corps du locuteur.

9.5.1 Analyse de la robustesse : résultats en images

Dans les Figures 9.14 et 9.15 un ensemble d'images illustrent les résultats de la détection du personnage. Ceci concerne non seulement la détection des visages mais aussi la segmentation des autres parties du corps. On doit souligner d'avance qu'aucune de ces images ne fait partie de la base de données utilisée pour calculer le modèle statistique de la peau. Afin de

placer le système face à des situations adverses, nous avons choisi des images réelles, enregistrées dans des environnements non contrôlés et montrant une large variété

- des teintes de peau,
- des conditions d'éclairage,
- des distances et positions face à la caméra, et
- des objets et couleurs dans le fond.

Ces exemples confirment donc la robustesse de la segmentation dans des scènes complexes. Les erreurs qui apparaissent sous la forme de petits accrochages ou petites régions manquantes sont dûs normalement au faible contraste existant sur ces zones. Ceci est le cas de l'exemple (a.4) pour lequel le visage est parfaitement détecté et c'est ensuite la segmentation qui n'arrive pas à trouver le contour des cheveux ou du bras droit. Par ailleurs, si le personnage n'est pas en vue frontale, la probabilité d'avoir des fusions est plus importante car le modèle de silhouette ne s'y adapte pas comme prévu (voir exemple (d.3)).

De ces observations on conclut que les petites erreurs présentes dans le masque du locuteur sont en grand majorité dûes à la difficulté de la segmentation du corps, et peuvent apparaître même après une bonne détection du visage.

9.5.2 Analyse des situations extrêmes

Dans le cadre de notre application, les conditions d'enregistrement étant non contrôlées, nous ne pourrons pas assurer la segmentation correcte du personnage en toute circonstance. Nous estimons pourtant intéressant de dédier cette section à l'analyse des situation extrêmes à éviter pour assurer le bon fonctionnement des algorithmes. Plusieurs facteurs peuvent être à l'origine de problèmes, et même plusieurs d'entre eux peuvent se manifester à la fois :

Occultation partielle du visage : de toute évidence, la présence de lunettes, de barbes, ou d'objets quelconques qui cachent une partie du visage comporte une contrainte de plus à surmonter. Soit parce que le morceau de peau visible ne peut pas être reconnu par lui même comme étant un visage, soit parce qu'en identifiant une partie du visage comme étant un visage tout entier on fausse les véritables proportions du personnage. La Figure 9.13 illustre l'un de ces cas extrêmes où, même si la peau a été parfaitement détectée, l'adaptation du modèle n'arrive pas à se faire complètement. Dans des situations moins extrêmes, l'algorithme arrive quand même à segmenter le personnage correctement (revoir les résultats de la Figure 9.15)

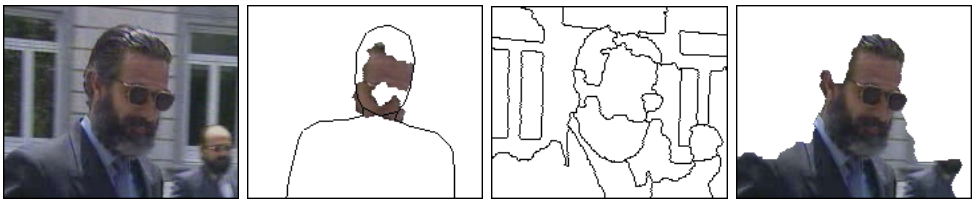


FIG. 9.13 – Exemple d'occultation partielle du visage.

FIG. 9.14 – Résultats : images originales.

FIG. 9.15 – Résultats : personnages segmentés automatiquement.

Présence de peau sur d'autres parties du corps : l'algorithme que nous avons développé identifie comme visage toute région de la hiérarchie de partitions ayant une couleur chair et les proportions d'un visage. A partir de cet ensemble de régions candidates, nous retenons à la fin celui de plus grande taille, sous l'hypothèse que le locuteur est le personnage le plus proche de la caméra. Nous avons vu dans les exemples de la section précédente que ce critère suffit normalement pour faire la différence entre le visage et les mains, ou des autres objets dans le fond. Cependant, il peut arriver que parmi toutes les régions identifiées comme visage il n'y ait pas un vrai visage. Ainsi, la présence d'autres parties du corps peut être à la base de détections erronées comme celle commise dans la Figure 9.16.

FIG. 9.16 – Exemple de présence de peau sur d'autres parties du corps.

Par ailleurs, s'il existe un autre objet dans le cadre de la scène dont la valeur de probabilité d'être de la peau est supérieure à celle du visage, la détection pourrait passer à côté du personnage.

Manque de contraste entre l'objet et le fond : plusieurs problèmes au niveau de la segmentation peuvent se produire si le fond ne se différencie pas de l'objet à contourner. Lorsqu'il n'existe pas un contour en proximité à celui du modèle de la silhouette, l'algorithme doit trancher pour ce qui concerne l'appartenance des régions au fond ou au personnage. Ainsi, le masque peut avoir un comportement envahissant et fusionner l'objet avec des zones du fond, ou dans l'extrême opposé, le masque peut avoir des régions manquantes.

La Figure 9.17 illustre un exemple de ce type de situation. Suite à une détection et modélisation correctes du personnage, le système n'a pas été capable de trouver le contour entre les épaules et le fond. Bien que sur cet exemple l'erreur finale soit important, si le manque de contraste se produit de façon locale, il n'est qu'à l'origine de petites fuites. Ceci est le cas pour quelques-uns des exemples de la Figures 9.15.

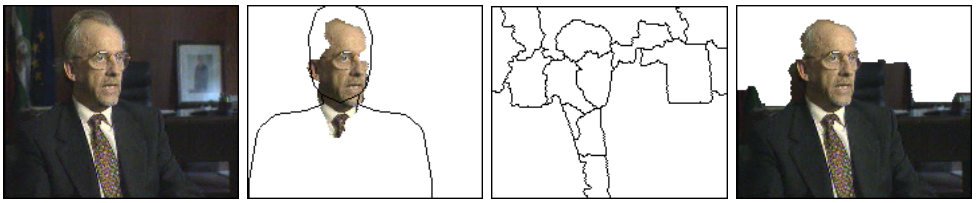


FIG. 9.17 – Exemple de la manque de contraste entre l'objet et le fond.

Présence de fortes ombres sur le visage : comme nous l'avons vu dans la Section 9.2.1.2, lorsque le visage est coupé par de fortes ombres, le spectre chromatique de la peau se disperse. Il est alors évident que la totalité d'une telle distribution ne sera pas couverte par une seule classe statistique. Le visage aura donc des pixels avec des probabilités beaucoup plus faibles que d'autres. Alors, au moment de l'analyse des régions, deux choses peuvent se passer : soit on détecte seulement une partie du visage et le système essaie de segmenter le personnage, soit aucune des deux parties n'est reconnue et par conséquent le système n'est pas capable de fournir le masque initial de la séquence.

9.6 Conclusions

Notre but dans ce chapitre était de fournir un algorithme capable d'initialiser le suivi du locuteur dans des séquences de vidéophonie ou vidéoconférence.

En guise d'étude préliminaire, nous avons abordé la modélisation statistique du spectre chromatique de la peau dans l'espace YUV ainsi que la modélisation géométrique de la silhouette d'un personnage. Ces deux indices étant ensuite utilisés comme éléments de support à la détection.

Nous travaux ici ont donné lieu à deux approches différentes :

- dans un premier temps, nous avons étudié la viabilité d'une technique de segmentation complètement automatique. Bien qu'elle ait fait preuve d'une grand robustesse lors de la détection du personnage dans des environnements complexes, elle est trop exigeante en temps de calcul pour être prise comme solution dans un système de temps réel. Ainsi,

- dans un deuxième temps, nous avons changé de stratégie, en abordant la mise en œuvre d'une procédure d'initialisation de nature interactive. Celle-ci garde la structure conceptuelle de la plupart des algorithmes automatiques dans un maximum de simplicité. L'interaction avec l'utilisateur permet de donner une même qualité des résultats dans un temps de calcul beaucoup plus court (de l'ordre du dixième de seconde sur le démonstrateur).

La force de ces algorithmes repose sur leur bon comportement dans des environnements non contrôlés. Ceci continuera à être notre premier souci lors de la conception d'une procédure de suivi du locuteur dans le prochain chapitre.

Pour ce qui concerne les améliorations envisageables, nous nous orientons vers le traitement des petits défauts de la segmentation. Au niveau de la prévention, on pourrait développer des filtres adaptés aux couleurs présentes dans la silhouette, de la même façon que nous nous sommes adaptés à la chromaticité de la peau. Ceci rendrait plus aisée la segmentation sur les parties faiblement contrastées. Au niveau de la correction, l'algorithme pourrait vérifier certaines contraintes sur la connectivité et la forme des différentes régions acceptées dans le masque, en forçant leur resegmentation si ceci donne une meilleure approximation de la forme de la silhouette.

Chapitre 10

Segmentation Récursive : le suivi du locuteur

Le corps principal du système est formé par une boucle de segmentation ayant en charge le suivi du masque du locuteur tout au long de la séquence. Chacun des algorithmes qui en fait partie a été choisi sur des critères de simplicité et de robustesse très strictes. Ainsi, le système tire son efficacité d'une part des nivellements, qui peuvent adapter l'intensité du filtrage au masque, et d'autre part de la segmentation hiérarchique, permettant l'analyse à plusieurs niveaux de détail. Deux étapes supplémentaires entourent ce noyau pour assurer une plus grande robustesse : la première concerne l'analyse de mouvement qui doit compenser le mouvement de la caméra et les déplacements les plus brusques du masque, la deuxième est une étape de correction d'erreurs pour donner plus de stabilité au système lors du traitement de longues séquences.

10.1 Présentation des algorithmes

La problématique du suivi d'objets a fait le sujet de nombreuses études pour donner des solutions à des applications très variées. Nous avons abordé le sujet d'un point de vue générique dans la deuxième partie de cette thèse, en développant une technique de mise en correspondance de partitions par graphes. Néanmoins, notre approche ici sera différente car la contrainte du temps réel fait de la simplicité une priorité face à d'autres critères comme celui de la versatilité.

Nous allons dédier cette première section à l'introduction de l'ensemble des algorithmes qui ont été choisis pour faire partie du système de suivi [32, 34, 35]. Le schéma de la Figure 10.1 montre la chaîne complète de traitement. La séquence d'images à l'entrée est traitée trame après trame en temps réel d'enregistrement. Comme sortie, l'algorithme rend une séquence de masques segmentant la figure du locuteur tant que la vidéoconférence dure. Le système se compose de trois blocs principaux :

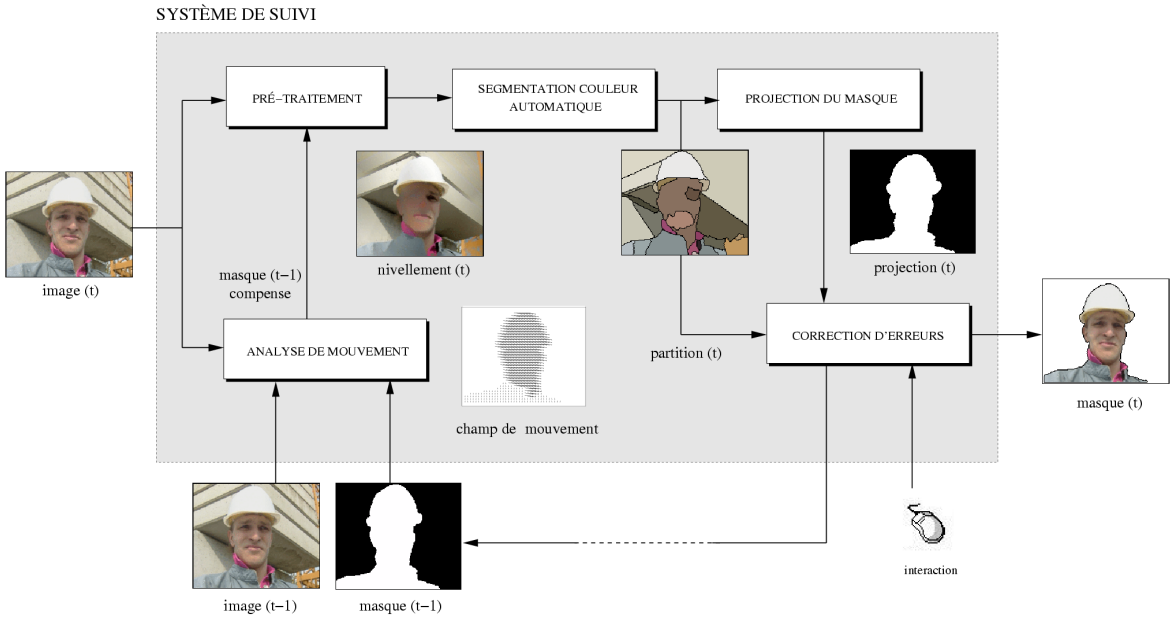


FIG. 10.1 – Schéma algorithmique du système de suivi.

Segmentation automatique : le corps du système est formé par un algorithme de segmentation qui détoure le locuteur au temps (t) en s'appuyant sur le masque obtenu à $(t-1)$. On procède ici en trois étapes : 1) étape de pré-traitement qui en simplifiant l'image d'entrée facilite la segmentation postérieure ; 2) étape de segmentation basée sur le gradient couleur ; et 3) étape de projection du masque sur la partition courante.

Analyse de mouvement : étant donné que la segmentation automatique utilise le masque à $(t-1)$ pour trouver l'emplacement des contours sur l'image courante, on doit compenser les déplacements grossiers du locuteur ainsi que le bougé de la caméra. Ceci est fait par une étape d'analyse de mouvement. Comme seule contrainte nous allons imposer à la compensation de préserver la connexité du masque.

Correction d'erreurs : dans des conditions réelles d'enregistrement le système doit tourner sans arrêt pendant plusieurs minutes en traitant des milliers d'images. Pendant ce temps il peut y avoir de petits accrochages à cause d'un mouvement trop rapide mal suivi ou d'une segmentation défectueuse. Ainsi, nous avons inclus une étape de contrôle interactive qui permet à l'utilisateur de corriger ces erreurs au vol par un simple clic de souris sur la zone erronée sans avoir besoin de réinitialiser le système.

Nous pouvons déjà avancer que l'ensemble des algorithmes va s'appuyer en grand partie sur le masque obtenu dans l'étape précédente, le but étant de maximiser la cohérence temporelle dans la chaîne de traitement. Dans les sections qui suivent nous rentrerons dans les détails de chacune des parties du système.

10.2 Segmentation automatique

Un algorithme de segmentation automatique tourne dans une boucle récursive tout au long de la séquence. On procède en trois étapes.

10.2.1 Pré-traitement des images

Pour permettre au système de s'adapter aux conditions d'enregistrement, il nous a fallu faire le partage entre les défauts qui peuvent être corrigés par la caméra (contrôle de gain, balance, etc.), et ceux qui ne peuvent être corrigés que par traitement des images (étirement du contraste et filtrage du bruit) que nous présenterons par la suite.

10.2.1.1 Etirement du contraste

Dans leur état normal, les composantes chromatiques de l'espace des couleurs YUV ont un très faible contraste, ce qui les rend peu appropriées à la détection de contours.

La solution la plus simple consiste à appliquer une anamorphose entre le maximum et le minimum des valeurs sur l'image. L'effet du contraste sera d'autant plus fort que l'intervalle (I_{min}, I_{max}) est petit.

Une deuxième approche consiste donc à restreindre l'anamorphose à un sous-intervalle $(I_a, I_b) \subset (I_{min}, I_{max})$, en obligeant toutes les valeurs en dessous de I_a à égaler I_a et, de la même façon, toutes les valeurs au-dessus de I_b à égaler I_b. En écartant des valeurs extrêmes, par exemple suite à un filtrage préalable de l'image, l'étirement de contraste est nettement plus fort. Néanmoins, le choix de (I_a, I_b) peut être critique, car si le personnage arrive à fusionner au fond la segmentation ne pourra plus les séparer.

Pour surmonter cet inconvénient, sans pourtant renoncer à étirer le contraste au maximum, nous proposons de rendre le calcul de (I_a, I_b) dépendant des valeurs qui entourent les contours du personnage. Pour cela, nous allons supposer que le masque à $(t-1)$ donne une approximation grossière de l'endroit où ceux-ci se trouvent sur l'image courante. Ainsi, nous proposons de calculer les valeurs de (I_a, I_b) de la façon suivante :

$$I_a = \vee \{ \wedge\{I_p, p \in (\delta M - M)\}, \wedge\{I_q, q \in (M - \varepsilon M)\}\} - 1 \tag{10.1}$$

$$I_b = \wedge \{ \vee\{I_p, p \in (\delta M - M)\}, \vee\{I_q, q \in (M - \varepsilon M)\}\} + 1 \tag{10.2}$$

où M est le masque à $(t-1)$, $(\delta M - M)$ et $(M - \varepsilon M)$ déterminent une bordure à l'extérieur et une bordure à l'intérieur du masque, et $\vee$ et $\wedge$ sont respectivement les opérateurs qui calculent le supremum et l'infimum dans ces ensembles. En d'autres mots, nous calculons les valeurs maximales et minimales dans chacune de ces bandes autour de la frontière du masque et ensuite nous prenons I_a comme le plus grand des minima moins un, et I_b comme le plus petit des maxima plus un. Pour éviter des écarts dûs au bruit une étape de filtrage précédera ces calculs.

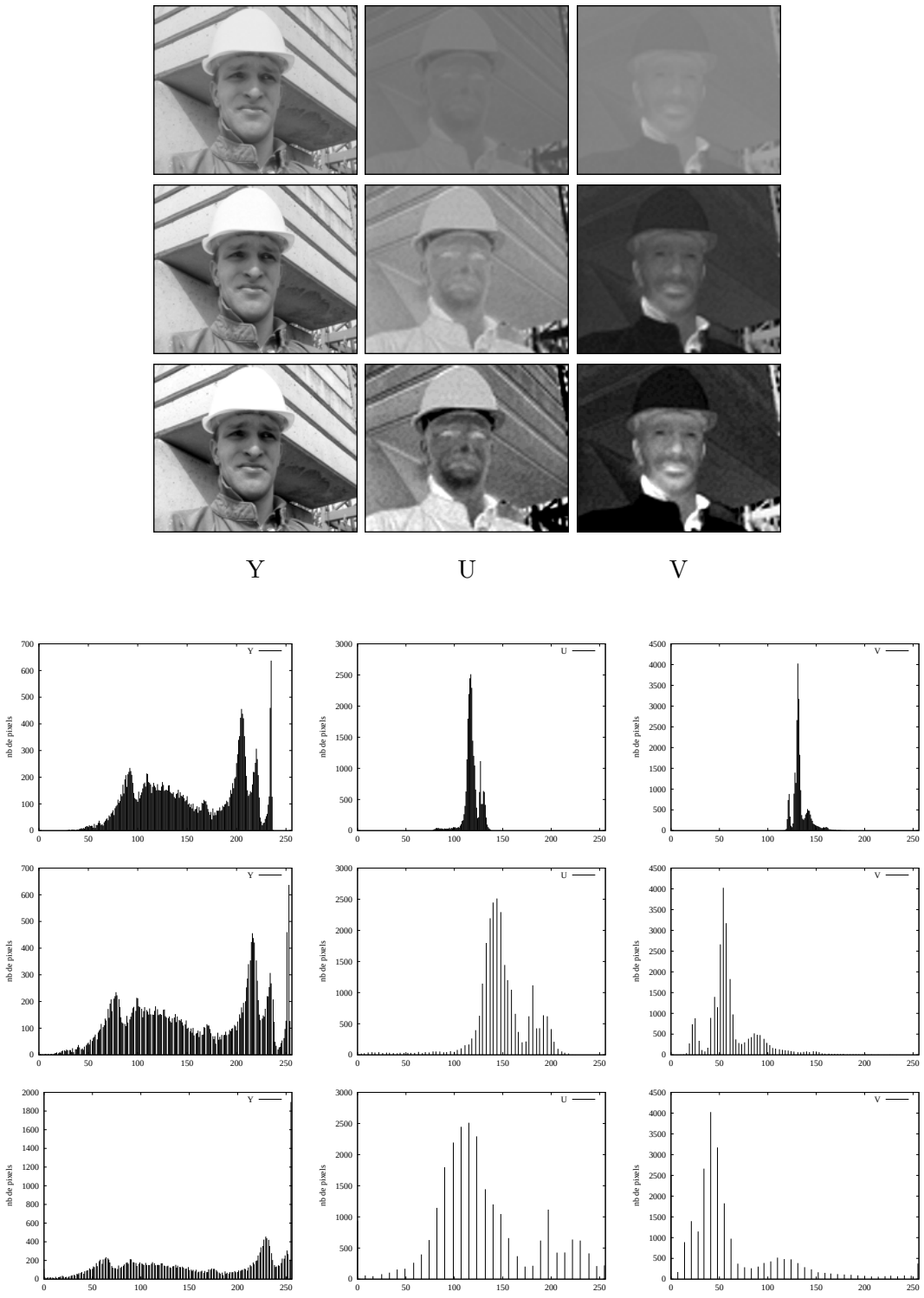


FIG. 10.2 – Etirement du contraste. Images et histogrammes correspondant aux composantes YUV originales, après étirement du contraste, et après étirement du contraste adapté au masque.

L'exemple montré dans la Figure 10.2 permet de comparer le résultat obtenu au moyen d'un étirement du contraste entre le maximum et le minimum de l'image, et celui calculé à partir des valeurs sur les frontières du masque. Le gain en contraste du personnage par rapport au fond est spécialement remarquable sur les composantes chromatiques, ce qui nous permettra ensuite d'avoir de meilleurs résultats dans la segmentation couleur.

10.2.1.2 Simplification de l'image par nivellement

Dans cette deuxième étape notre but est double, nous voudrions supprimer le bruit et les textures, mais aussi tout autre contour qui puisse perturber la segmentation du personnage. Pour cela, les nivellements étudiés dans le Chapitre 3 se présentent comme des outils idéaux. Ils arrivent à créer des zones plates ou quasi-plates très larges tout en préservant inchangée la position des fortes transitions du signal qui marquent les contours des objets.

Par ailleurs, dans leur rôle de filtres, les nivellements permettent d'adapter l'intensité de la simplification à la fonction marqueur. Parmi les exemples montrés lors de l'étude des applications, la possibilité d'adapter l'intensité du filtrage aux contours d'un masque connu à l'avance nous intéresse ici spécialement. Dans ce cas, le nivellement est construit en deux étapes au moyen d'un marqueur plat en dehors d'une bande qui garde les valeurs de l'image originale. Ainsi, ce que nous avons appliqué en premier lieu à l'analyse d'une scène routière dans la Figure 3.22 (page 46) donne des résultats très performants dans le cadre de l'application actuelle où notre but est de détecter les contours du personnage. Pour illustrer ceci nous avons fait une mise à jour de cet exemple dans la Figure 10.3. Le nivellement s'applique aux trois composantes couleur en utilisant le masque du personnage à $(t-1)$ pour simplifier l'image d'entrée au temps (t). Le fait de considérer des zones quasi-plates explique ici très bien le besoin de procéder à un étirement de contraste de façon préalable au calcul du filtrage. Sur ces résultats notons que,

- la simplification a lieu aussi bien à l'extérieur du masque qu'à l'intérieur. Notre intérêt est de réussir une forte simplification de l'image en gardant intacte la zone où se trouvent les contours du personnage avec le fond, en vue de leur détection postérieure ;

- l'intensité du filtrage dépend de la largeur de la bande qui contourne le masque. Plus la bande est étroite, plus la simplification est forte et plus elle s'approche des contours de l'objet. En bas de la Figure 10.3 nous avons illustré cette progression avec quelques images.

Pour éviter de fixer la largeur de la bande au début de la séquence, nous allons la calculer de façon adaptative sur chaque trame. Nous verrons ceci plus en détail dans la Section 10.4.

Cette capacité remarquable de simplification permettra ensuite d'assurer une robustesse de suivi maximale face à l'accrochage d'objets dans l'arrière plan.

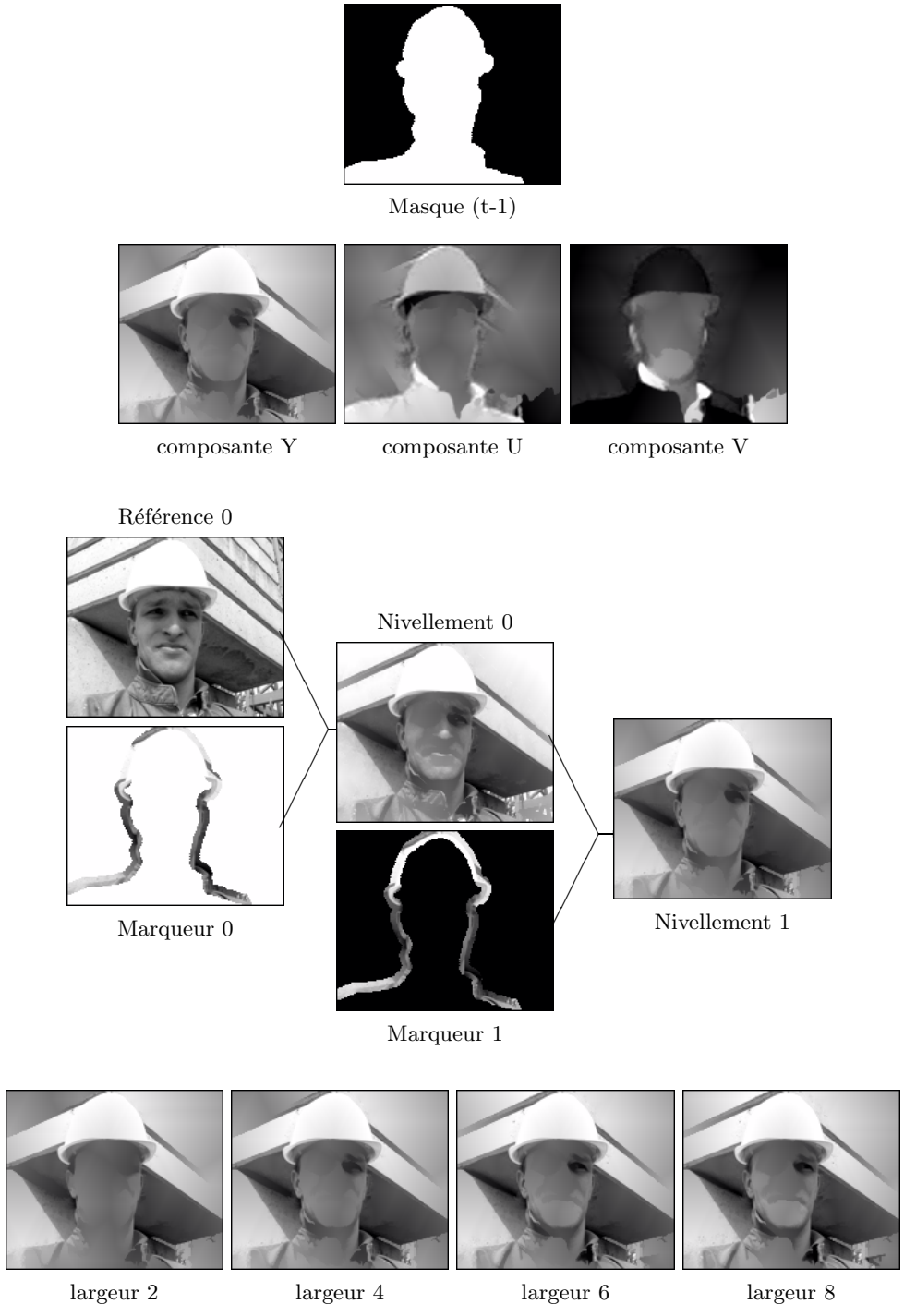


FIG. 10.3 – Du haut vers le bas. Nivellement adapté au masque à $(t-1)$ calculé sur les trois composantes de l'image couleur après étirement du contraste. Détails du processus de calcul sur la composante de luminance. Dépendance de l'intensité du filtrage à la largeur de la bande qui entoure le masque.

10.2.2 Segmentation hiérarchique

Suite aux étapes d'étirement du contraste et filtrage, où l'image d'entrée est rendue appropriée au traitement, un algorithme hiérarchique aura en charge la segmentation de la scène. Par leur versatilité, nous avons déjà eu recours à ces algorithmes à plusieurs reprises. Un bref rappel de leur mode de fonctionnement nous mène à différencier deux étapes :

1.- Calcul d'une image gradient à partir des trois composantes couleur.

2.- Inondation hiérarchique du relief du gradient. Les minima sont ordonnés par leur volume d'extinction au fur et à mesure qu'ils fusionnent.

La Figure 10.4 illustre ces étapes en quelques images. Les détails peuvent être consultés dans le Chapitre 4 où ces techniques ont été présentées par la première fois dans cet mémoire.

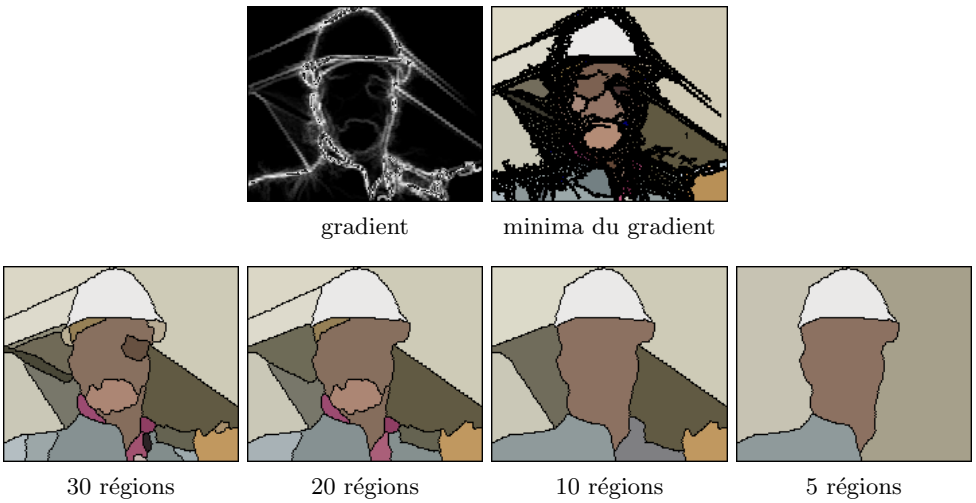


FIG. 10.4 – Hiérarchie de partitions.

L'adaptation du degré de simplification selon l'intérêt de chaque zone de l'image fait à présent que la finesse de la hiérarchie se concentre autour des contours du personnage, le découpage du fond étant plus grossier. Cette hiérarchie de partitions sera doublement utilisée par la suite :

– dans *l'étape de projection* on aura besoin d'une partition grossière de la scène sur laquelle projeter automatiquement le masque du locuteur à $(t-1)$ pour trouver les contours au temps (t).

– dans *l'étape de correction d'erreurs* on aura besoin d'une partition plus fine qui puisse donner un support au processus de correction des petites erreurs qui sera fait de façon interactive par l'utilisateur.

10.2.3 Projection du masque

La dernière étape du processus de segmentation concerne la *projection* du masque à $(t-1)$ sur une partition de la scène au temps courant. Ceci doit nous permettre de décider quelles régions vont constituer le nouveau masque et quelles régions appartiennent au fond. La façon la plus simple de traiter cette prise de décision est de calculer pour chaque région le pourcentage de la surface qui est couvert par la projection du masque et décider ensuite :

$$\frac{\text{Surf}(R_i \cap M_{(t-1)})}{\text{Surf}(R_i)} > s_{min} \quad \Rightarrow \quad R_i \in M_{(t)}$$

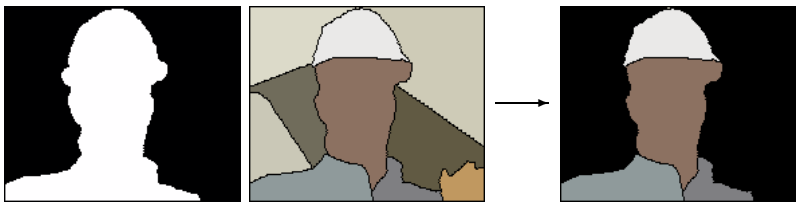


FIG. 10.5 – Etape de projection du masque.

Le nombre de régions de la partition utilisée pour ceci doit s'établir en fonction de la complexité de la scène à segmenter en tenant compte du fait que :

- si la partition est trop grossière on risque de fusionner le personnage au fond, mais
- si la partition est trop fine un déplacement mal compensé du masque précédent pourrait entraîner une assignation erronée des petites régions proches des contours.

A la recherche d'un compromis satisfaisant, nous avons décidé de travailler sur des partitions comportant de 10 à 15 régions, la finesse de celles-ci apparaissant appropriée compte tenu de la taille des images et la composition des scènes de type vidéoconférence. Face à des situations particulières, cette valeur peut être manuellement ajustée sur le démonstrateur à n'importe quel instant pendant la phase d'initialisation ou de suivi. Comme travail futur on pourrait envisager la conception d'un algorithme capable d'adapter automatiquement le nombre de régions à la complexité de la scène enregistrée. Néanmoins, la performance obtenue devrait être toujours pondérée par la complexité de calcul ajouté à la chaîne de traitement.

La projection du masque clôt le processus de segmentation automatique. A la sortie, on dispose d'une version actualisée du masque du locuteur sur l'image courante. Toutes les étapes qui se sont déroulées jusqu'à présent ont été conçues pour exploiter au maximum l'information proportionnée par le masque à $(t-1)$. Il est donc nécessaire d'inclure une étape de compensation de mouvement pour pouvoir assurer la robustesse du système lorsqu'il y a un mouvement de la caméra ou un fort déplacement du locuteur.

10.3 Analyse de mouvement

Pour ce qui concerne la perception du mouvement on doit différencier le mouvement réel du mouvement apparent des objets. Le *mouvement réel* c'est le mouvement qui s'est vraiment produit, définissant une trajectoire 3D dans l'espace. Le *mouvement apparent* c'est le mouvement qu'on aperçoit sur le plan de l'image, celui-ci étant connu sous le nom de *flot optique*. Le flot optique [44] assigne à chaque pixel de l'image un vecteur de mouvement défini sur un plan 2D.

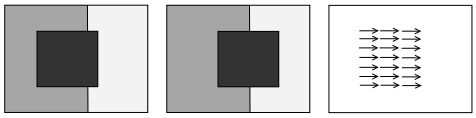


FIG. 10.6 – Estimation du mouvement.

Néanmoins, le mouvement ne peut pas être mesuré sur l'image, mais uniquement estimé à partir des changements qu'il provoque, trame après trame, au niveau des valeurs des pixels. Ainsi, nous dirons que l'estimation de mouvement est un problème de nature « mal posée », dans le sens où à partir des données disponibles sur l'image il se peut qu'on n'arrive pas à obtenir une solution pour tout pixel, ou que celle-ci ne soit pas unique.

Suite à ce bref aperçu, nous présentons le schéma de traitement que nous avons mis en place pour estimer et compenser les déplacements du masque. La Figure 10.7 montre l'enchaînement des trois étapes qui constituent notre algorithme :

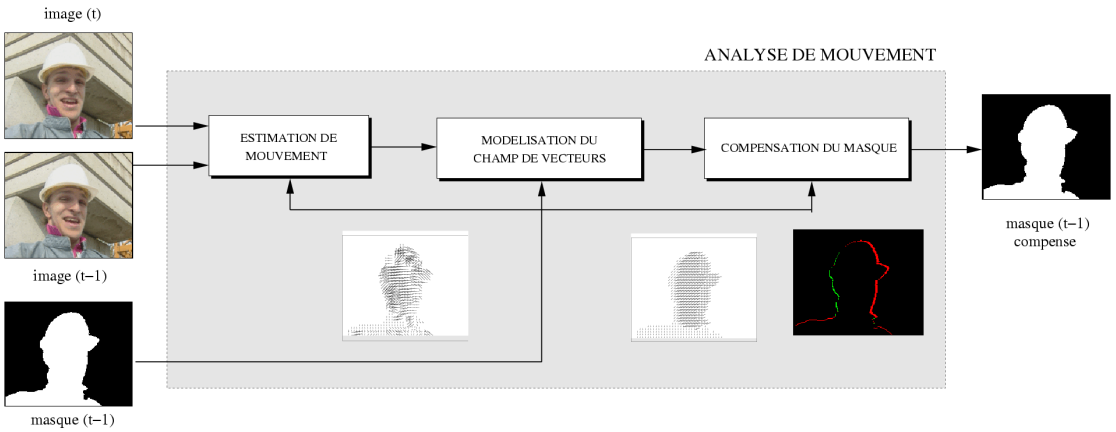


FIG. 10.7 – Schéma de l'algorithme d'analyse de mouvement.

Estimation du mouvement : cette première étape a pour but d'estimer le champ de vecteurs qui déplace les objets de $(t-1)$ à (t) selon le mouvement aperçu.

Modélisation du champ de vecteurs : l'estimation de mouvement étant trop sensible au bruit et aux textures, une étape de régularisation du champ de vecteurs s'impose.

Compensation du masque : finalement, la compensation en mouvement du masque à $(t-1)$ devient un simple étirement des frontières en accord avec le modèle calculé.

Ces étapes seront présentées en détail dans les sections qui suivent.

10.3.1 Estimation du mouvement

Les techniques d'estimation de mouvement se classent en deux grands groupes en fonction de leur travail au niveau des pixels ou au niveau des régions :

Estimation du mouvement au niveau du pixel : il s'agit de calculer un champ de flot optique, dense ou local, tenant compte des variations qui se produisent pixel à pixel. Sous cette rubrique nous pouvons mentionner les techniques différentielles [69], qui calculent le champ des vitesses à partir des dérivées spatio-temporelles de la luminance. Leur point faible concerne la sensibilité au bruit.

Estimation du mouvement au niveau des régions : il s'agit d'élargir le support de calcul en faisant l'hypothèse que tous les pixels d'une même région ont un déplacement cohérent. Parmi ces techniques, ressort pour sa simplicité le « Block Matching » [47], qui recherche le meilleur appariement entre des régions carrées. Des approches plus complexes vont s'appuyer sur une segmentation plus intelligente, car le découpage carré ne s'adapte pas au contenu de l'image.

Pour plus de détails nous reportons le lecteur à l'étude comparative de Barron dans [5]. Dans le cadre de cette application, nous avons adopté un algorithme de « Block Matching »de recherche exhaustive, car c'est une technique très répandue qui offre un bon compromis entre simplicité de calcul et qualité des résultats. La procédure de mise en œuvre est la suivante :

1.- L'image au temps $(t-1)$ est découpée en blocs de N×N pixels. Le choix de la taille du bloc est le résultat d'un compromis entre la robustesse face au bruit (blocs grands) et le fait de ne traiter qu'un seul objet dans le même bloc (blocs petits). Dans notre implémentation nous travaillerons avec des blocs de 8×8 sur des images QSIF.

2.- Ensuite l'algorithme cherche quel est le meilleur déplacement $(\vec{d})$ pour chacun des blocs de l'image à $(t-1)$ vers l'image au temps (t). Pour cela, on va limiter la zone de recherche à une zone autour du bloc de 3N×3N pixels, c'est-à-dire, 9 fois la taille du bloc. Cette valeur va déterminer le déplacement maximal que nous serons capables de détecter. Dans la zone de recherche le bloc peut se déplacer sur toutes les positions possibles avec une précision de 1 pixel, comme l'illustre la Figure 10.8.

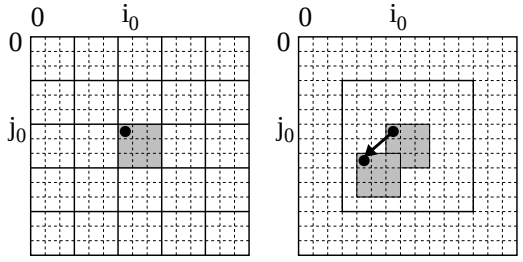


FIG. 10.8 – Exemple de « Block Matching » en prenant des blocs de 3×3 pixels.

3.- Le but est de trouver le vecteur $(\vec{d})$ qui minimise la différence entre le bloc au temps $(t-1)$ et son correspondant au temps (t). C'est-à-dire que nous allons minimiser la somme de différences de tous les pixels du bloc :

$$\vec{d} = \arg\min_{d \in R} \sum_{\vec{x} \in B} |I_{(t-1)}(\vec{x} + \vec{d}) - I_{(t)}(\vec{x})| \tag{10.3}$$

où B désigne la région de support du bloc et R l'espace de recherche de tous les déplacements possibles.

4.- De plus, nous allons exploiter la connaissance du masque du locuteur à $(t-1)$ afin de limiter le calcul du champ de vecteurs. On ne considérera pour l'analyse de mouvement que les blocs ayant une intersection non nulle avec le masque (voir Figure 10.9 (c)).

10.3.2 Modélisation du champ de vecteurs

Etant donné que le déplacement de chaque bloc est calculé indépendamment de ses blocs voisins, il arrive très souvent d'avoir des discontinuités de mouvement parmi eux. Si ces vecteurs sont appliqués directement au masque, ils seront à l'origine de l'apparition des trous et des nouvelles composantes dans le masque compensé.

Pour éviter ceci, une modélisation de l'ensemble des vecteurs, préalable à la compensation de mouvement, est devenue une pratique courante. L'utilisation d'un modèle permet d'uniformiser le champ de mouvement à l'intérieur d'une certaine région de support. Le choix d'un modèle est intimement lié

- au type de mouvements à traiter, car la complexité du modèle varie suivant qu'il s'agit de paramétrer de simples translations sur un plan ou, à l'extrême opposé, des transformations plus complexes dans un espace 3D ;

- aux caractéristiques des objets en mouvement, car le modèle sera de nature différente selon que les objets sont rigides ou déformables, de petite ou grande taille, etc.

Compte tenu de ceci, nous avons conçu une étape de modélisation spécifique à notre problème qui, tout en restant robuste, ne ralentisse pas significativement la chaîne complète de traitement. Pour minimiser la complexité des calculs, certaines contraintes ont dû être imposées. Le mouvement de la caméra sera considéré comme translationnel, ce qui permet de compenser le masque avec un seul modèle qui groupe les déplacements globaux ainsi que ceux du personnage. Par ailleurs, nous particulariserons le modèle à la compensation des mouvements de la tête et des épaules d'un personnage, ceux-ci étant les plus courants dans les séquences de vidéophonie.

Rappelons que dans le cadre de notre application l'analyse de mouvement a pour seul but de compenser le masque binaire de $(t-1)$ à (t). Par conséquent, nous ne sommes intéressés qu'à modeler le mouvement à l'intérieur de cette région. Dans cette approche, la modélisation du champ de vecteurs sera calculée séparément pour les composantes horizontales et verticales : d'un côté, ceci permet de paralléliser les deux processus dans une architecture hardware dédiée ; d'un autre côté, ceci permet d'économiser du temps de calcul en prenant un modèle adapté à chaque composante. Voyons à présent les ces analyses en détail.

Modélisation de la composante verticale

Nous avons observé que couramment, dans des séquences de vidéophonie, la composante verticale du champ de vecteurs ne compense que des mouvements uniformes dûs à la caméra ou au déplacement du personnage. Pour les compenser il suffit donc d'appliquer un modèle purement translationnel à l'intérieur du masque, ce calcul étant de complexité très réduite.

Ainsi, le déplacement vertical est modelé par un seul vecteur qui correspond à la médiane des vecteurs du champ de mouvement. Notons qu'il s'agit d'une médiane pondérée, tenant compte du fait que quelques uns des blocs qui sont intervenus dans l'estimation de mouvement n'étaient pas totalement à l'intérieur du masque. Le facteur de pondération du vecteur $v^i = (v_x^i, v_y^i)$ appartenant au bloc B^i est calculé comme la somme de tous les pixels du bloc qui sont à l'intérieur du masque M :

$$p(v^i) = Card(B^i \cap M) \tag{10.4}$$

A partir de ceci, la médiane pondérée de la composante verticale des vecteurs de mouvement v^i peut être calculée de façon efficace en trois étapes :

1.- Classer les composantes verticales par ordre croissant : $v_y^1 \leq v_y^2 \leq \ldots \leq v_y^K$
2.- Créer la liste des poids associée à ce classement : $(p_1, p_2, \ldots, p_K)$
3.- Obtenir la médiane pondérée comme la valeur $v_y^{k'}$ tel que l'index k' vérifie que :

$$\sum_{1<l\leq k'} p(v_y^l) \approx \sum_{k'<l\leq K} p(v_y^l) \tag{10.5}$$

Même si cette modélisation peut paraître grossière, on doit se rappeler que la compensation du masque de $(t-1)$ à (t) n'est faite que pour approximer l'emplacement du locuteur, les

algorithmes de segmentation étant en charge de l'extraction précise des contours sur l'image courante.

Modélisation de la composante horizontale

Les mouvements horizontaux peuvent altérer de façon importante les contours du masque. Cette variabilité est beaucoup plus faible sur la composante verticale où, même lorsque le personnage acquiesce d'un signe de tête, les contours du masque ne changent pas significativement. Nous verrons que ceci est plus critique lorsque le personnage tourne la tête. Une compensation correcte de ces mouvements exige donc une modélisation plus complexe que celle que nous avons implémentée précédemment.

De plus, nous devons faire ici particulièrement attention à la différence entre les mouvements de la tête et ceux des épaules. Ceci vient du fait que les mouvements horizontaux de la tête sont nettement plus prononcés. Nous risquons de compenser de façon erronée une partie des contours si un seul modèle est calculé sur la totalité du masque. Pour éviter ce problème, il suffit de modeler séparément la région de la tête et celle des épaules.

L'information qui rend possible cette différentiation provient de l'étape d'initialisation où un modèle de silhouette a été utilisé pour composer le premier masque. En revenant sur les exemples de la Section 9.2.2 on peut observer comment, aux contours de la silhouette, on avait adjoint un contour interne marquant la transition entre la tête et les épaules. Ce zonage permet de composer le masque par union de deux régions qui nous servent à mieux compenser les mouvements du personnage tout au long du processus de suivi.

La modélisation de la composante horizontale du champ de vecteurs à l'intérieur de chaque région est faite sur trois zones différentes : deux bandes latérales (b_g, b_d) et une partie centrale (b_c). Notons que la largeur de ces bandes s'adapte à la largeur du masque (voir Figure 10.9 (f,g)). Sur le centre du visage les bandes latérales comprennent chacune 20% de la largeur de la tête, évoluant vers une largeur nulle aux extrémités du masque. Nous verrons plus tard que l'importance de ce zonage est en rapport avec la détection des mouvements de la tête. Le déplacement du buste étant plus simple à modeler, nous pouvons utiliser des bandes de largeur uniforme (30% de la largeur totale du buste).

Etant donné que ces régions sont de faible taille, leur déplacement peut être représenté par une seule médiane locale calculée en suivant la procédure décrite auparavant. Ensuite, la comparaison de ces médianes $(v_x(b_g), v_x(b_c), v_x(b_d))$ permet de différencier trois types de mouvements : des translations, des rotations et des changements d'échelle. La notation utilisée lors des comparaisons est la suivante :

$$v_x(a) \approx v_x(b) \quad \Rightarrow \quad v_x(a) = v_x(b) \pm \varepsilon, \quad 0 \leq \varepsilon \leq 1 \tag{10.6}$$

$$v_x(a) > v_x(b) \quad \Rightarrow \quad v_x(a) = v_x(b) + \delta, \quad \delta > 1 \tag{10.7}$$

$$v_x(a) \geq v_x(b) \quad \Rightarrow \quad v_x(a) = v_x(b) + \delta, \quad \delta \geq 0 \tag{10.8}$$

Translations : un déplacement uniforme est détecté lorsque les médianes ont le même signe et la même intensité dans les trois zones d'analyse,

$$\begin{array}{lll} v_x(b_{g,c,d}) \leq 0, & v_x(b_g) \approx v_x(b_c) \approx v_x(b_d) & \text{à gauche} \\ v_x(b_{g,c,d}) \geq 0, & v_x(b_g) \approx v_x(b_c) \approx v_x(b_d) & \text{à droite} \end{array} \tag{10.9}$$

Rotations : lorsque le personnage tourne la tête, les médianes ont le même signe et montrent une relation d'intensité particulière telle que,

$$\begin{array}{llllll} v_x(b_{g,c,d}) \leq 0, & |v_x(b_c)| \geq |v_x(b_g)| & \text{et} & |v_x(b_c)| > |v_x(b_d)| & \text{à gauche} \\ v_x(b_{g,c,d}) \geq 0, & |v_x(b_c)| > |v_x(b_g)| & \text{et} & |v_x(b_c)| \geq |v_x(b_d)| & \text{à droite} \end{array} \tag{10.10}$$

A cause du volume de la tête, le mouvement aperçu sur la partie centrale du visage est plus intense. Ceci s'explique tenant compte du fait que pour un même déplacement, plus l'objet est proche de la caméra, plus le mouvement détecté est fort. Notons que dans ces cas, le champ de mouvement global demande un déplacement de la région beaucoup plus fort que celui qui a vraiment lieu sur les contours.

Changements d'échelle : lorsque le personnage s'éloigne ou s'approche de la caméra il se produit un changement d'échelle qui altère la taille du masque d'une image à la suivante. Du point de vue du modèle, ceci créé des vecteurs d'une même intensité mais opposés dans leur sens de déplacement,

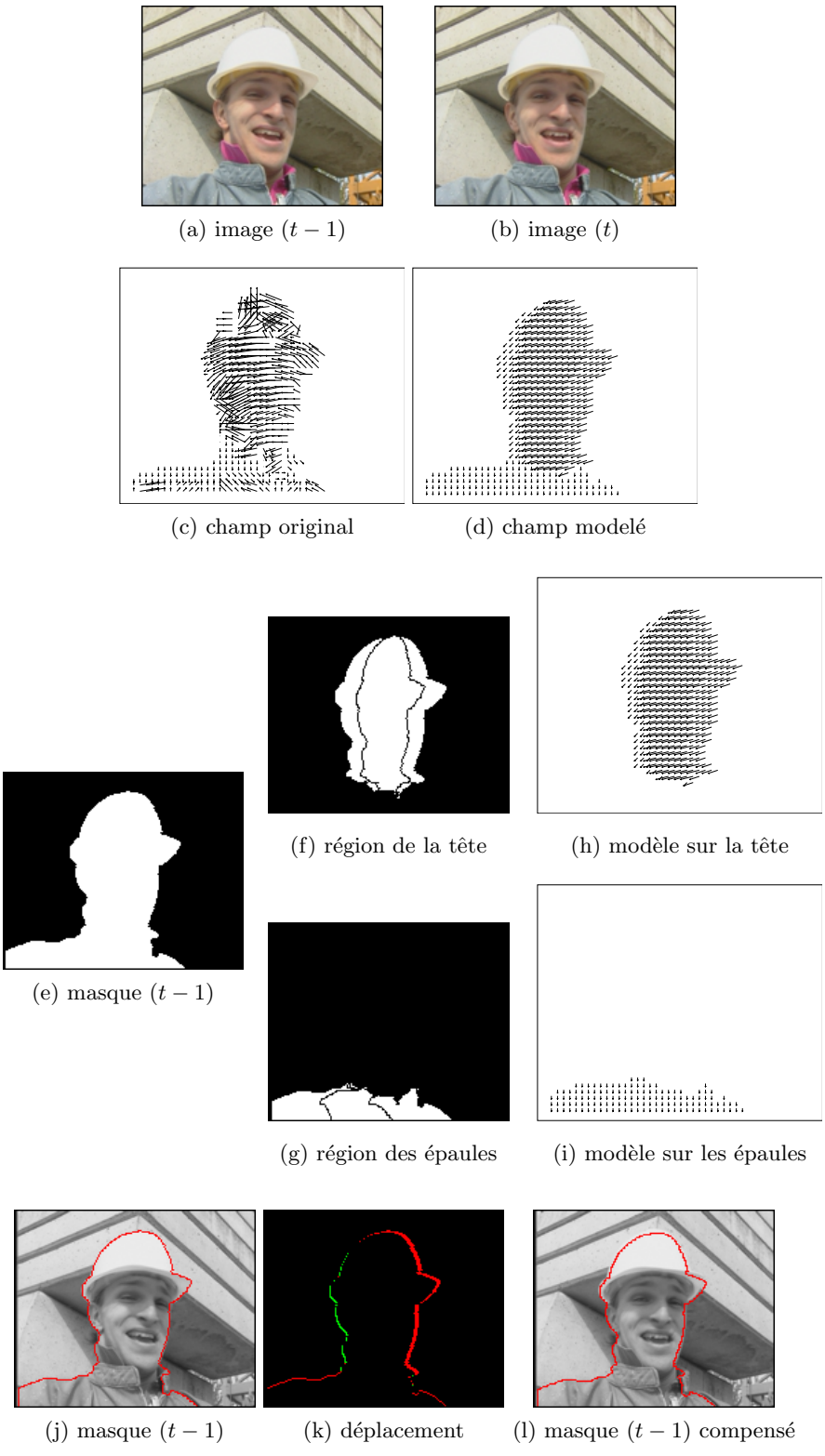
$$\begin{array}{llll} v_x(b_g) < 0, v_x(b_d) > 0 & \text{et} & |v_x(b_g)| \approx |v_x(b_d)| & \text{approchement} \\ v_x(b_g) > 0, v_x(b_d) < 0 & \text{et} & |v_x(b_g)| \approx |v_x(b_d)| & \text{éloignement} \end{array} \tag{10.11}$$

Nous avons vu donc comment le découpage en trois bandes de la région de la tête et celle des épaules permet de détecter des mouvements complexes (rotations 3D et changements d'échelle) avec une complexité de calcul très réduite. En effet, il a suffit de modeler chaque bande avec un seul vecteur de déplacement pour avoir des résultats complètement satisfaisants sans faire appel à des algorithmes plus complexes.

Sur l'exemple de la Figure 10.9 nous avons représenté les deux composantes du mouvement suite au calcul du modèle, pour la tête (h), pour les épaules (i) et pour le masque complet (d). Au niveau de l'implémentation, notons que le mouvement dans la zone centrale n'est pas modelé car seuls les vecteurs médians sur les bandes sont censés être utilisés pour compenser le déplacement des contours du masque.

10.3.3 Compensation du mouvement du masque

L'analyse de mouvement finit avec une dernière étape de compensation qui a pour but de déplacer le masque selon le mouvement modelé entre $(t-1)$ et (t). Remarquons que ce n'est pas l'image au niveau des gris qui est compensée mais le masque binaire qui marquera la



(a) image $(t-1)$ (b) image (t)

(c) champ original (d) champ modelé

(e) masque $(t-1)$ (f) région de la tête (h) modèle sur la tête

(g) région des épaules (i) modèle sur les épaules

(j) masque $(t-1)$ (k) déplacement (l) masque $(t-1)$ compensé

FIG. 10.9 – Analyse de mouvement.

position estimée du locuteur sur l'image courante. Grâce à la modélisation, la compensation du mouvement n'est ici qu'un simple étirement des contours du masque. Ceci est fait en deux étapes que nous détaillons par la suite.

Compensation horizontale

Pour mener à terme la compensation horizontale il suffit de déplacer les pixels du contour gauche (droit) du masque selon le vecteur de la bande gauche (droite). Ainsi, pour déplacer un pixel (i, j) du contour gauche

- *vers la gauche*, la valeur du pixel (i, j) est copiée sur les v_x pixels voisins à gauche.
- *vers la droite*, la valeur du pixel $(i-1, j)$ est copiée sur les v_x pixels voisins à droite.

où v_x est la composante horizontale du vecteur du mouvement. Si la position $(i-1, j)$ est en dehors de l'image, nous considérons toujours que le masque se sépare du bords. La procédure est symétrique pour les pixels du contour droit. Notons que cette façon de procéder évite de créer des trous lors des changements d'échelle.

FIG. 10.10 – Procédure de compensation des déplacements horizontaux sur le contour du masque.

Compensation verticale

Le déplacement vertical est plus facile a compenser, puisque nous l'avons modelé par une translation uniforme. Il suffit donc de déplacer tous les points du masque v_y pixels. Le seul point qui fasse problème concerne les bords de l'image, car suite à la compensation il se peut que de nouvelles zones apparaissent. Ainsi,

- si le déplacement vertical se fait *vers le bas*, les nouvelles lignes qui rentrent par le haut de l'image sont supposées être en dehors du masque pour éviter des accrochages ; et,

- si le déplacement vertical se fait *vers le haut*, les nouvelles lignes qui rentrent par le bas sont censées appartenir au masque en faisant l'hypothèse que le buste du personnage continue en dessous du bord de l'image. La valeur de ses lignes sera obtenue par copie de la ligne la plus basse du masque.

Comme dernière remarque il est important de souligner que la procédure d'analyse de mouvement que nous avons développée proportionne un masque compensé comme une seule

composante connexe sans trou. De plus le contour du masque est régulier sans les discontinuités causées par l'estimation de mouvement au niveau des blocs.

10.4 Capacité d'adaptation et de correction d'erreurs

Finalement, pour assurer au système une robustesse et une souplesse maximales, il nous a fallu lui conférer une capacité d'adaptation aux différents paramètres de la scène ainsi qu'une capacité de récupération en cas d'erreur.

Adaptation

La variabilité des séquences à traiter peut faire fluctuer la performance de tout système de suivi s'appuyant sur des paramètres fixés d'avance. Par contre, la capacité d'adaptation a un coût qui comporte parfois un retard temporel ou, dans la plupart des cas, un incrément important du temps de calcul. Ainsi, dans le cadre de notre application, nous avons limité l'adaptation du système aux paramètres qui se sont montrés les plus critiques du point de vue de la robustesse.

Implicitement, lors de l'analyse de mouvement nous avons déjà rendu variable la largeur des bandes qui interviennent dans l'étape de modélisation. Ceci permet de traiter avec la même précision le mouvement de personnages qui se trouvent à des échelles différentes.

De la même façon, nous permettons aux bandes utilisées dans l'étape de pré-traitement de s'adapter en largeur à la taille et à l'intensité du mouvement du personnage. Afin de rendre l'intensité du filtrage équivalente à des tailles différentes, nous forçons les bandes à être plus larges dans les plans courts et plus étroites dans les plans moyens. Ceci détermine la largeur maximale et minimale que la bande peut prendre.

Ensuite, c'est en fonction de l'intensité du mouvement que la valeur précise est fixée. Lorsque de faibles mouvements sont détectés, il est logique de supposer que le masque compensé donne une bonne prédiction de la position du personnage, ce qui nous permet de minimiser la largeur de la bande. Par contre, dès que les mouvements sont importants, les contours du masque compensé sont moins fiables et demandent une bande plus large car la zone d'incertitude augmente.

Ainsi, nous voudrions que la bande soit minimale autour des objets petits ayant un faible mouvement, et qu'elle s'élargisse jusqu'à prendre une valeur maximale lors du traitement des grands objets ayant des mouvements forts. Cette adaptation de la largeur des bandes est en rapport avec l'intensité du filtrage. Rappelons que plus étroite est la bande autour du masque, plus forte est la simplification atteinte.

Correction d'erreurs

Un deuxième point important en ce qui concerne la robustesse du suivi est en rapport avec la possibilité de corriger de petites erreurs pendant le traitement de la séquence.

Dans un premier temps, nous avons mis en place une procédure de détection et correction automatique basée sur l'analyse de la forme du masque résultant [35]. Ces algorithmes permettent de détecter des accrochages grâce à une deuxième inondation lors de la projection du masque. Néanmoins, cette procédure, trop coûteuse en temps de calcul, a dû être remplacée par une approche interactive qui n'entraîne aucun calcul supplémentaire.

Ainsi, sur la version finale du démonstrateur, la correction d'erreurs est faite de façon semi-automatique. L'utilisateur peut, par un simple clic de souris sur l'image, ajouter au masque des régions qui en ont été exclues par erreur ainsi que supprimer celles qui, appartenant au fond, se sont accrochées au personnage.

Pour mener à terme cette correction, l'algorithme s'appuie sur une partition de l'image non aperçue par l'utilisateur lors de l'interaction. Pour lui il suffit de cliquer sur un point de la zone erronée et c'est ensuite le système qui fait basculer la région sous-jacente de la partition d'un côté à l'autre du masque. Notons que cette partition est plus fine que celle que nous avons utilisée lors de la projection du masque afin de donner une plus grande précision lors de la correction des petites erreurs.

Ce algorithme a été conçu sur la base d'un partage dans la tâche du traitement des erreurs, l'utilisateur étant en charge de la détection, le système de sa correction. Ceci comporte une épargne importante dans le nombre d'opérations requises tout en restant très simple à gérer par l'utilisateur. Autrement, le système devrait évaluer des critères de stabilité plus complexes trame après trame, même lorsque le résultat était correct au départ.

Il est également important de signaler que cette interaction est faite « au vol », pendant que le suivi est en cours. Il n'y a pas besoin donc de réinitialiser le système lorsqu'une correction doit être faite.

10.5 Conclusions

Nous avons présenté dans ce chapitre la conception d'un système de suivi capable de tourner en temps réel avec un maximum de robustesse. L'ensemble de ces algorithmes constitue le noyau du démonstrateur implémenté dans le cadre du projet M4M. Nous allons dédier le chapitre suivant à l'analyse de résultats.

La vitesse étant ici une priorité, il nous a fallu renoncer à des algorithmes sophistiqués et prendre des approches plus simples et plus rapides qui tirent leur efficacité d'un enchaînement astucieux. La prise en compte du masque lors de l'étirement du contraste et le filtrage des images d'entrée rend la segmentation hiérarchique plus performante. A son tour, l'inclusion d'une étape de modélisation lors de l'analyse de mouvement rend robuste une estimation

autrement trop sensible aux erreurs. De plus, la technique de compensation que nous avons mise en place rend un masque sans trous et avec des contours réguliers.

Dans les limites imposées par la contrainte du temps réel, plusieurs améliorations sont encore envisageables. Nous présentons les deux que nous estimons les plus importantes :

Construction d'un modèle du personnage : nous avons utilisé un modèle du personnage pour donner un support à la procédure d'initialisation. De la même façon, il serait envisageable d'augmenter la performance du suivi en incluant un modèle du personnage qui évolue tout au long de la séquence. Ceci nous permettrait d'adapter les différentes étapes du système aux attributs du modèle (couleurs, textures ou formes).

Correction automatique d'erreurs : une deuxième possibilité d'amélioration concerne l'inclusion d'une procédure automatique de détection et de correction d'erreurs. D'un côté, on pourrait tirer profit de la connaissance d'un modèle du personnage pour éviter les erreurs plus voyantes. D'un autre côté, on devrait être capable de lancer une réinitialisation lors d'une déformation trop importante du masque ou d'une perte totale, comme il arrive si quelqu'un passe devant la caméra.

Chapitre 11

Analyse des Résultats

Après avoir décrit dans les chapitres précédents l'ensemble des algorithmes, nous procéderons ici à la présentation du démonstrateur construit dans le cadre du projet M4M, ainsi qu'à l'analyse des résultats obtenus. Nous voudrions ainsi illustrer la performance achevée par notre système tout en respectant les contraintes de simplicité et de robustesse fixées au début du projet. De même, nous présenterons deux des applications qui pourraient tirer profit du détourage automatique du locuteur : le filtrage sélectif, permettant une meilleure efficacité des codeurs, et l'édition multimédia, permettant la composition personnalisée de séquences.

11.1 Présentation du démonstrateur

La plus grande partie des efforts concernant la mise en œuvre de cette application a été consacrée à la réalisation d'un démonstrateur temps réel, constituant en quelque sorte une épreuve de vérité. Ainsi, suite à l'assemblage des algorithmes dans une interface utilisateur et grâce l'adjonction d'une caméra, nous avons pu tester la qualité, la robustesse et la fiabilité de notre système dans des conditions réelles d'enregistrement.

La totalité des algorithmes de segmentation et de suivi que nous avons élaboré ont été implémentés de façon optimisée dans le démonstrateur. Ceci a été fait par un groupe spécialisé, qui a dû entreprendre une série de travaux parallèles pour permettre de commander de manière fine tous les paramètres du système. De plus, une interface utilisateur a été développée pour avoir accès de manière simple au réglage du démonstrateur (voir Figure 11.1). Elle permet l'accès à deux sortes de paramètres, ceux qui sont liés à la caméra (gain, contraste, zoom, etc.), et ceux qui sont propres aux algorithmes, en particulier pour ce qui concerne le nombre de régions de la segmentation (en rapport avec la simplicité/complexité de la scène à segmenter) et la largeur de la bande d'incertitude autour du masque dans laquelle les contours de l'objet sont cherchés (en rapport avec l'intensité des mouvements du personnage).

Tout ceci permet de régler le système de manière optimale pour chaque séance de vidéoconférence. Alternativement, il est possible d'utiliser le démonstrateur sur des séquences enre-

gistrées au préalable. Cela permet entre autres de comparer les performances obtenues sur des séquences de test, avec celles d'approches concurrentes.

FIG. 11.1 – Interface utilisateur du démonstrateur. L'image de gauche correspond à celle enregistrée par la caméra, celle de droite est obtenue par détourage automatique du locuteur.

En suivant le schéma algorithmique des chapitres précédents, le démonstrateur opère en deux modes :

- *Mode d'initialisation* : au cours de cette phase, le système procède à la création du masque initial du locuteur. Il suffit d'appuyer sur le bouton `Init` et faire un clic de souris sur le visage pour déclencher le processus de segmentation.

- *Mode de suivi* : pendant cette phase le masque du locuteur est suivi et constamment affiché à l'écran. Face aux petites erreurs, il existe la possibilité de les corriger sans arrêter le système de suivi.

Finalement, pour ce qui concerne les spécifications techniques du démonstrateur, notons qu'il s'agit d'un logiciel qui, en tournant sur un ordinateur portable (pentium III, 500MHz), est capable de traiter une séquence de taille QSIF à 10 images par seconde.

11.2 Résultats en images

Nous montrons ici une séquence qui illustre parfaitement le type de situations qui peuvent avoir lieu pendant une vidéoconférence. Un sous-échantillonnage apparaît dans la Figure 11.2. Sur ces images on doit remarquer :

- la complexité du fond, contenant de petits objets et des couleurs proches de celles du personnage. Ceci rend complexe la segmentation car il existe un contraste fiable sur certains contours qui éventuellement peut empêcher la séparation de l'objet du fond.

– le mouvement du personnage, spécialement lorsque le bras apparaît et disparaît en couvrant une partie du visage. Ceci rend complexe le calcul du déplacement des contours du masque, car les mouvements détectées sur cette partie ne sont pas cohérents avec ceux du reste.

– le mouvement qui a lieu dans l'arrière plan, avec la présence d'un autre personnage non suivi. La présence de forts mouvements dans le voisinage du masque peut causer des erreurs suite à un accrochage sur une partie du fond.

Les résultats montrés dans la Figure 11.3 ont été obtenus en temps réel par le démonstrateur. Quatre interactions pour la correction des petites erreurs d'accrochage ont été réalisées aussi en temps réel. L'observation détaillée de ces images permet d'évaluer la bonne qualité de la segmentation produite. Ceci prouve la robustesse du système face au mouvements dans le fond et aux fuites, en gardant toujours les contours du masque parfaitement placés sur les contours du personnage.

11.3 Etude des fonctionnalités

Parmi les nouvelles fonctionnalités qui vont apparaître dans le domaine de la vidéophonie et la vidéoconférence, le détourage automatique du locuteur permet d'envisager deux grands axes d'application : l'un concerne le *codage* adapté à l'objet, permettant d'économiser du débit sans perte de qualité sur les régions d'intérêt ; l'autre concerne l'*édition* personnalisée des séquences, permettant à l'utilisateur de changer la composition de la scène. Dans les sections qui suivent nous verrons quelques exemples.

11.3.1 Application au codage

Jusqu'à présent la plupart des systèmes de codage avaient comme seul but de comprimer les images en vue de la transmission ou du stockage. Face à eux, le nouveau standard MPEG4 ouvre une porte vers de nouvelles fonctionnalités, nettement pour ce qui concerne le traitement de la séquence au niveau des objets. Dans ce domaine, les algorithmes de segmentation automatiques vont jouer un rôle clé en permettant aux codeurs de s'adapter au contenu. C'est-à-dire qu'ils doivent fournir les masques des objets à coder pour permettre de régler la qualité sur chaque région en fonction de son intérêt visuel.

Néanmoins, dans un certain nombre de circonstances, on n'aura peut-être pas la possibilité de procéder au codage des contours. En particulier au cours des communications mobiles, même la transmission des images est compromise, car dans ce type d'application le débit disponible est très faible et les canaux de transmission sont souvent bruités. Or, les travaux que nous avons menés à terme permettraient d'améliorer la qualité des ces transmissions. Il s'agit, à partir d'une segmentation de la scène, de simplifier les régions de peu intérêt. Dans ce cas, même sans information des contours, le fait d'avoir simplifié une partie de l'image entraîne une réduction du coût de codage. Deux options sont ici à considérer :

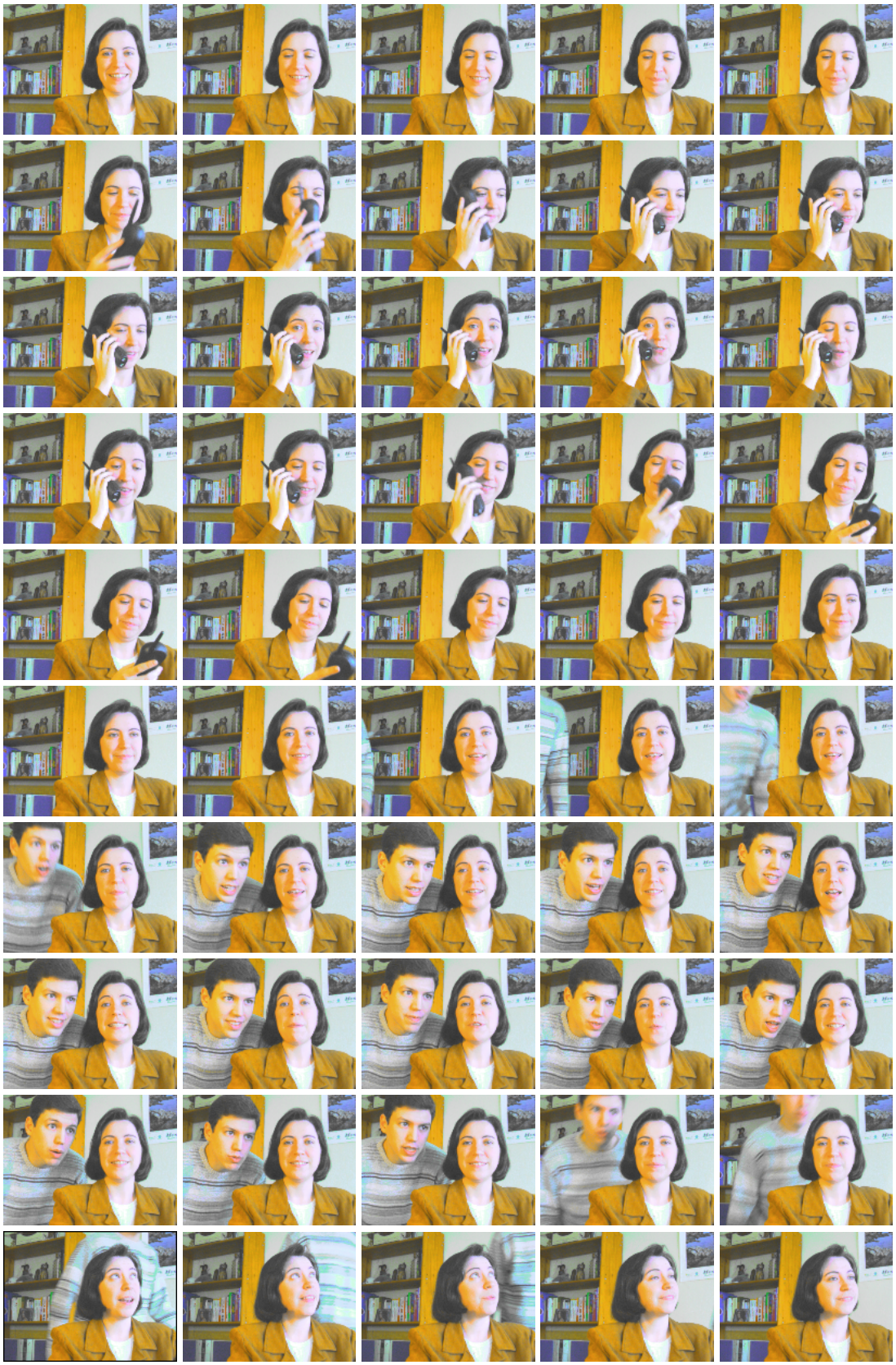


FIG. 11.2 – Séquence enregistrée. Trames : 0,8,16,...,400.

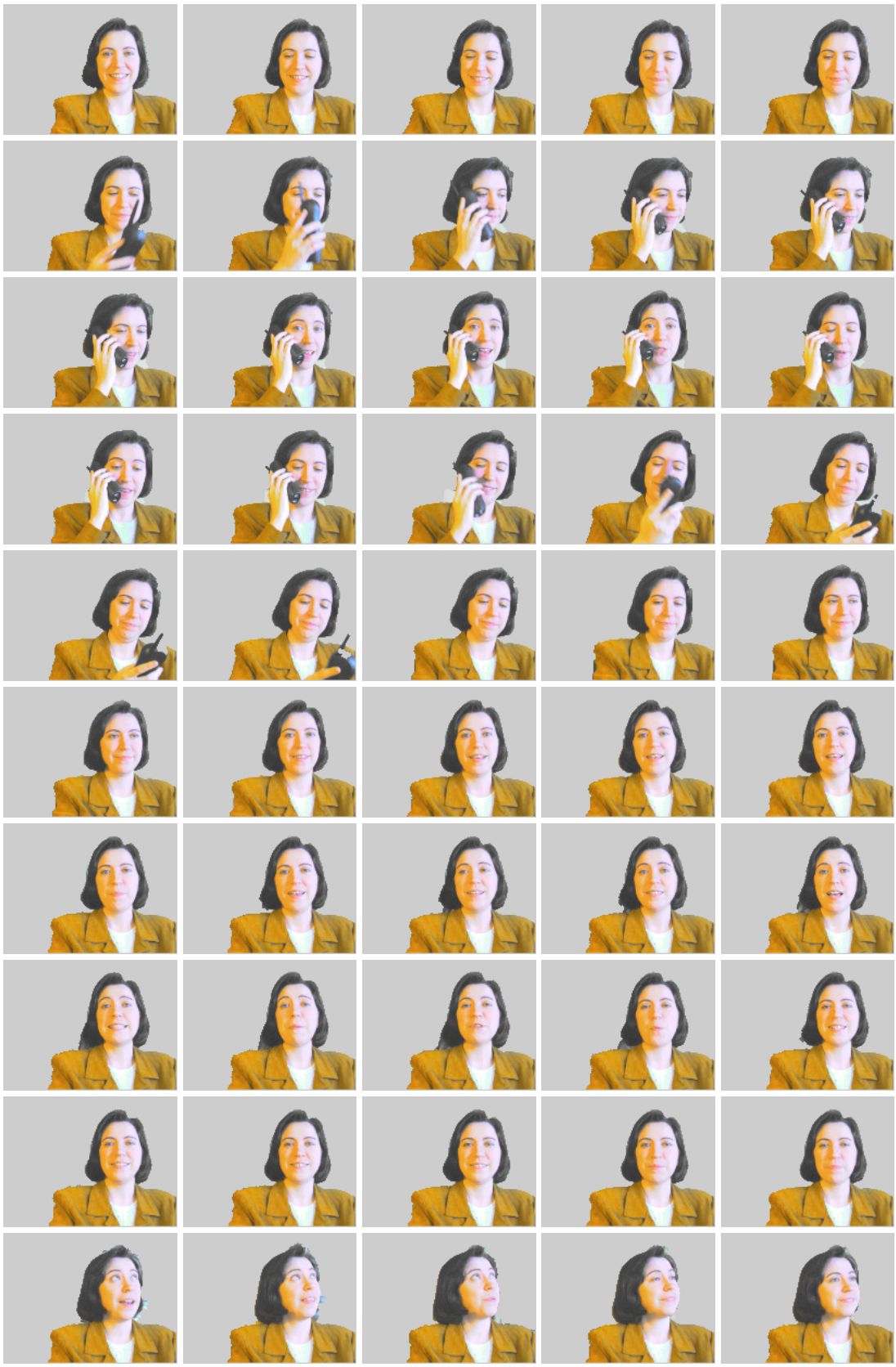


FIG. 11.3 – Séquence de masques. Trames : 0,8,16,...,400.

Transmission d'un fond fixe : pour atteindre des débits très faibles, le fond pourrait être complètement masqué. Le fait d'imposer un fond statique évite le codage des mouvements dans l'arrière plan, ce qui permet d'augmenter la qualité de l'objet d'intérêt.

Transmission d'un fond simplifié : dans le cas où un certain débit peut être utilisé pour coder l'information du fond, une meilleure efficacité du codeur est obtenue suite à l'application d'un filtrage. Celui-ci étant fort dans les zones de peu d'intérêt et faible, voir inexistant, dans les zones d'intérêt.

Pour illustrer ceci nous avons codé et comparé trois versions différentes de la séquence d'exemple : 1) séquence originale telle qu'elle a été enregistrée ; 2) séquence filtrée dans laquelle nous avons simplifié le fond tout en conservant l'image originale à l'intérieur du masque du locuteur ; et 3) séquence simplifiée de manière drastique en imposant un fond uniforme.

Dans le Tableau 11.1 on peut comparer la qualité obtenue en forçant les trois séquences à respecter un même débit[1]. On observe sur ces données un gain moyen de 1,6dB pour la séquence nivelée, arrivant jusqu'à 3,9dB pour celle à fond plat. Notons que le codeur utilisé traite tous les pixels de l'image à part égale, sans faire de différence entre l'objet et le fond. C'est-à-dire que tous les pixels sont codés, même dans le cas où il s'agit d'une zone du fond complètement plate. Bien évidemment, sur ces zones le système atteint une qualité de codage supérieure à celle des zones plus texturées, le gain sur le personnage étant légèrement inférieur à celui de l'image toute entière. Ceci est évitable pour les codeurs capables de concentrer le débit sur le masque et de coder le fond séparément avec moins de précision.

Dans le Tableau 11.2 la comparaison est faite en fixant une valeur de qualité et en laissant ensuite au codeur le choix d'adapter le débit à cette contrainte. Le codage de la séquence nivelée achève une réduction qui va jusqu'à 36% sans perte de qualité dans le masque, arrivant jusqu'à 60% dès que le fond est masqué. Observons que le gain est indépendant du débit, pouvant être réglé en fonction de l'intensité du filtrage. Pour un filtrage très faible la simplification du fond est moins visible mais la réduction du débit est aussi moins importante. Pour des filtrages de plus en plus forts le débit diminue, la réduction la plus forte étant obtenue pour un fond plat. Il ne faut pas oublier que ces valeurs dépendent de la taille du masque par rapport à celle de l'image, l'exemple montré étant typique des séquences de vidéoconférence (40% de l'image est couverte par le masque).

Finalement, dans la Figure 11.4 nous avons mené à terme une comparaison croisée entre les données des deux tableaux. Ceci permet de comparer visuellement les trois séquences, d'abord à qualité égale (colonne gauche et centrale), ensuite à débit égal (colonne centrale et droite). Comme seule remarque notons que le gain obtenu est plus important sur des images ayant de forts mouvements dans le fond comme celle utilisée dans l'exemple.

Sur cet exemple on a pu observer comment les nivellements simplifient l'image en supprimant un certain nombre de détails sans pour autant rendre l'image floue ou déplacer les

[1]Les données relatives au débit ont été obtenues au moyen d'un codeur MPEG4 sans information des contours.

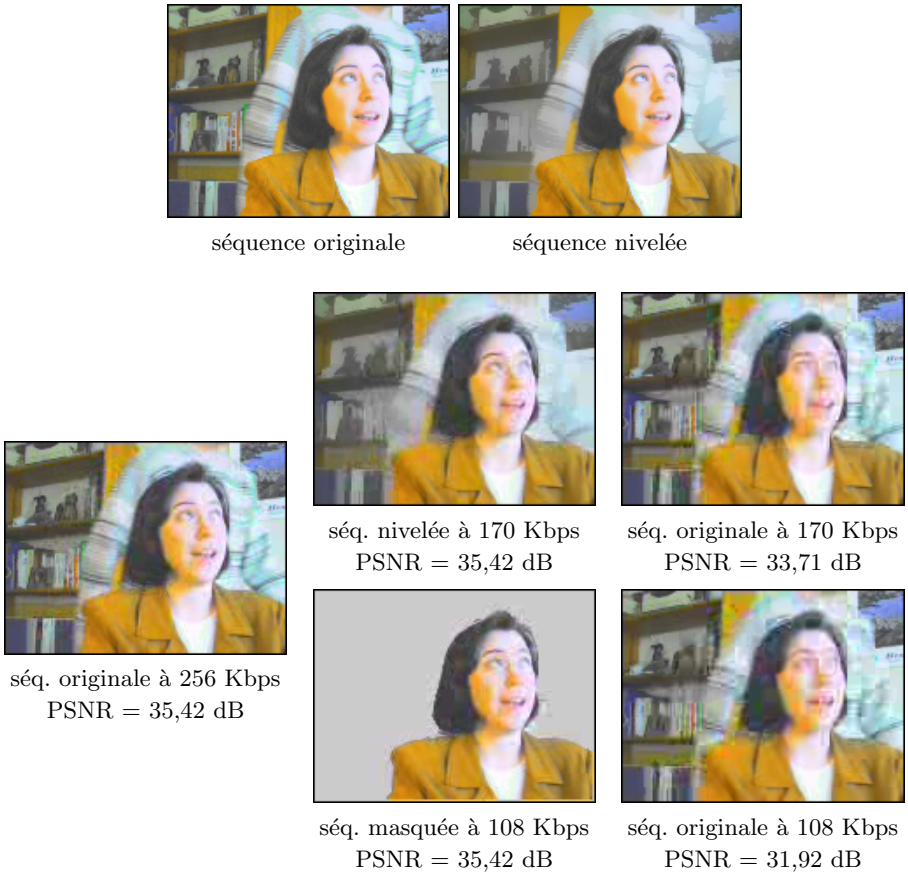


FIG. 11.4 – Comparaison des séquences originales, nivelées et masquées lorsqu'elles sont codées à qualité égale ou à débit égal.

	PSNR [dB]		
débit [Kbps]	séquence originale	séquence nivelée	séquence masquée
256 (22,11 : 1)	35,42	37,11 (+1,69)	38,79 (+3,37)
128 (43,18 : 1)	32,52	34,19 (+1,67)	36,05 (+3,53)
64 (82,20 : 1)	29,88	31,54 (+1,66)	33,77 (+3,89)

TAB. 11.1 – Comparaison du facteur de qualité PSNR à débit égal. Entre parenthèse à gauche, taux de compression calculé par rapport à la transmission de la séquence originale sans comprimer. Entre parenthèse à droite, gain en qualité.

	débit [Kbps]		
PSNR [dB]	séquence originale	séquence nivelée	séquence masquée
35,42	256	170 (−33%)	108 (−57%)
32,52	128	82 (−36%)	44 (−65%)
29,88	64	42 (−34%)	24 (−62%)

TAB. 11.2 – Comparaison du débit de codage requis pour assurer une même valeur de PSNR. Entre parenthèse, réduction du débit de codage par rapport à celui de la séquence originale.

contours. En d'autres termes, l'image est toujours parfaitement lisible et agréable à regarder ; par contre, le fait de simplifier certaines textures ou de supprimer certains détails facilite le codage de l'image et permet de réduire le débit de manière conséquente selon l'intensité du filtrage.

11.3.2 L'édition multimédia des séquences

Une deuxième application qui ressort naturellement est celle de l'édition multimédia des séquences. Il s'agit ici de tirer profit du détourage du locuteur pour le faire apparaître dans un autre environnement. Ceci permet d'envisager un grand nombre d'utilités, quelques-unes applicables dans un environnement de travail, d'autres étant plus ludiques. Pour illustrer la largeur de ce nouveau marché, nous avons composé une série de petites séquences d'exemples dans la Figure 11.5 que nous commentons par la suite :

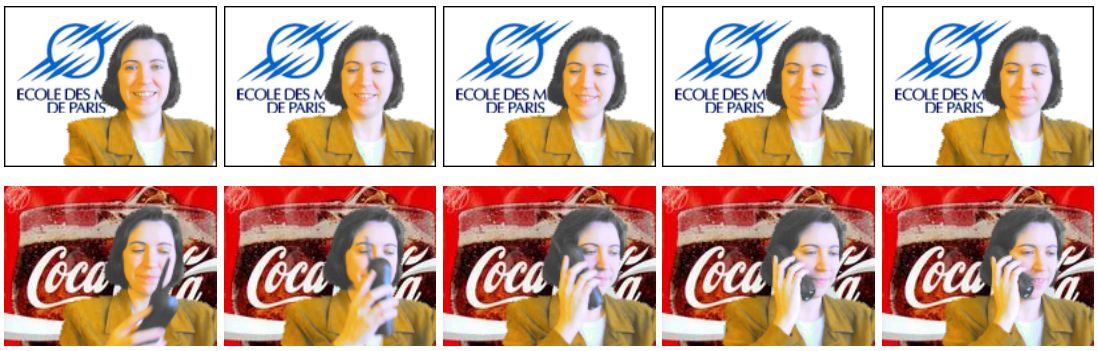
Inclusion dans le fond de logos corporatifs : une pratique habituelle des grandes compagnies et institutions est déjà d'habiliter une salle spéciale pour les vidéconférences. Ceci pour avoir des conditions d'enregistrement contrôlées et une homogénéité du décor. De telles contraintes ont nettement limité l'extension de ce service. Par ailleurs, dès que le détourage automatique du locuteur est faisable, le fait de changer le fond d'un bureau quelconque par un fond corporatif revient à un simple choix de l'utilisateur.

Inclusion de la publicité dans le fond : on a déjà vu comment certaines marques financent des appels téléphoniques en incluant des messages publicitaires vocaux. Alors, pourquoi ne pas envisager l'inclusion des affiches publicitaires dans le fond, du moment où la transmission de l'image vidéophonique est possible. Dans la séquence composée l'impact visuel est réussi sans pourtant entraîner des interruptions gênantes dans la communication.

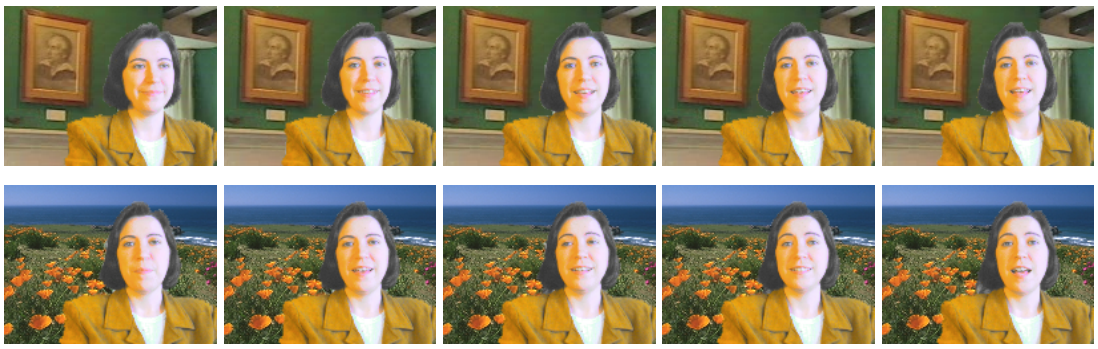
Changement de l'image du fond à finalité ludique : selon le choix de l'utilisateur, l'image du fond pourrait le transporter dans un autre environnement, soit simplement pour s'amuser, soit pour préserver le caractère confidentiel de son domicile, soit pour centrer l'attention sur lui en s'abstrayant des événements qui ont lieu dans l'arrière plan.

Inclusion de sous-titres ou d'objets quelconques dans l'avant plan : comme une fonctionnalité agrégée, on peut également situer le masque segmenté entre deux plans. C'est-à-dire, entre une image de fond et un avant-plan incluant des objets segmentés ou des sous-titres comme ceux de l'exemple.

Et ceci n'est qu'un bref aperçu des applications qui peuvent découler de la connaissance du masque du locuteur, beaucoup d'autres sortiront sûrement en réponse au marché. Actuellement, la segmentation interactive pour l'édition de séquences constitue un grand sujet d'intérêt industriel.



Inclusion dans le fond de logos corporatifs ou de publicité



Changement de l'image du fond à finalité ludique

Inclusion de sous-titres ou d'objets quelconques dans l'avant plan

FIG. 11.5 – Exemples des séquences composées.

11.4 Conclusions

Ce chapitre clôt la dernière partie de cette thèse dans laquelle nous avons présenté l'ensemble des travaux qui ont été faits dans le cadre du projet européen M4M. A la fin de ces quatre ans, nous avons accompli avec succès la tâche qui nous nous étions fixée, concernant le développement d'un système de segmentation en temps réel en vue de la vidéophonie. Sur les résultats obtenus on doit remarquer les performances atteintes au niveau de la robustesse et de la vitesse de traitement. La réalisation d'un démonstrateur met ainsi en valeur les techniques proposées.

Finalement, nous avons dédié une partie importante de ce chapitre à étudier les applications industrielles possibles qui peuvent valoriser nos travaux. Dans le domaine du codage nous avons montré comment le filtrage sélectif de la séquence permettait de réduire le débit de façon très importante (jusqu'à 60% si le fond est masqué) sans perte de qualité sur la région du personnage. Dans le domaine de l'édition multimédia des séquences nous avons montré plusieurs applications : inclusion de logos corporatifs ou de publicité dans le fond ou changement de l'arrière plan. Plusieurs voies ont été annoncés, beaucoup d'autres sortiront du fruit de l'imagination de nos lecteurs.

Chapitre 12

Conclusion

Ce chapitre clôt l'exposition des travaux faits au cours de cette thèse. Dans un rapide survol des techniques présentées, nous mettrons l'accent sur nos apports dans les différents domaines. En dernier lieu, une étude des perspectives aura pour but d'encourager la poursuite de ces recherches.

12.1 Apports de cette thèse

Notre objectif principal concernait la mise en correspondance de partitions en vue du suivi d'objets, la mise en œuvre d'un démonstrateur temps réel dans le domaine de la vidéophonie étant une application immédiate de ces recherches. Pour cela, un vaste domaine de travail a dû être couvert. Faisons ici un bref rappel de nos apports pour chacun des sujets principaux auxquels nous nous sommes adressés.

Partie I : Le Paradigme de la Segmentation

Dans un premier temps, avant d'entreprendre le traitement des séquences, nous avons considéré comme prioritaire de mettre au point des outils dans le domaine du filtrage et de la segmentation d'images fixes.

Les filtres jouent un rôle très important dans le processus de segmentation. Ils ont en charge de rendre appropriées au traitement les images bruitées, même dégradées pour la compression du codage. Dans ce domaine, nous nous sommes intéressés à une nouvelle classe de filtres morphologiques connexes connus comme *nivellements*. Lors d'une étude détaillée de leurs applications, nous avons montré qu'ils sont aussi puissants que versatiles. Particulièrement, par leur capacité de simplification sous une approche multiéchelle (en utilisant des marqueurs de plus en plus grossiers) ou sélective (lorsqu'un masque est disponible).

Notre apport principal ici a consisté au développement pour la première fois d'une extension des nivellements aux espaces vectoriels. Deux approches différentes ont été étudiées : la première, étant de nature pseudo-scalaire, produit des couleurs visuellement agréables, idéales pour des applications de codage ; la deuxième, étant purement vectorielle, présente des performances très attractives pour des applications basées sur la segmentation.

Pour ce qui concerne la technique de segmentation, nous avons repris un algorithme morphologique : la *Ligne de Partage des Eaux*. Pour elle, nous avons proposé une nouvelle formulation du processus d'inondation classique en introduisant un ensemble de lois d'absorption par analogie avec les algorithmes de fusion de régions. Le trait le plus marquant de cette nouvelle formulation est la versatilité qu'elle apporte au processus de segmentation par LPE. Elle permet d'associer à chaque marqueur des attributs et des conditions d'absorption à évaluer lors d'une rencontre avec d'autres lacs. Ceci permet au processus d'inondation de tenir compte des propriétés intrinsèques des régions au lieu de se restreindre exclusivement à l'information disponible sur le gradient.

La maîtrise acquise dans cette étape nous a permis par la suite d'exploiter tout le potentiel de ces techniques en proposant des solutions nouvelles à des problèmes déjà classiques.

Partie II : La Mise en Correspondance

La partie centrale de cette thèse a été consacrée au problème de la mise en correspondance de partitions. Nos apports dans ce domaine concernent la présentation d'une nouvelle méthode qui combine des algorithmes de mise en correspondance de graphes avec de nouvelles techniques d'édition basées sur la segmentation morphologique. Nous étions motivés par le large potentiel qu'une telle représentation de la scène peut offrir : d'un côté, la structure relationnelle du graphe fournit une plus grande robustesse au système ; de l'autre côté, elle permet d'extraire des informations additionnelles sur le contenu de l'image.

Pour surmonter les inconvénients classiquement attribués aux algorithmes de mise en correspondance de graphes, nous avons orienté nos recherches dans le but de (1) faire se ressembler les partitions au maximum en dehors du processus de mise en correspondance, et (2) d'accélérer la prise de décision pendant le calcul des appariements. Ainsi, les atouts les plus importants de notre méthode sont constitués par l'inclusion de

- deux étapes d'*édition des partitions*, qui profitant de la segmentation hiérarchique rassemblent au maximum leur contenu. Ceci permet de surmonter l'instabilité de la segmentation purement 2D ;

- la mise en œuvre d'un *processus de relaxation probabiliste*, où nous avons accéléré la prise de décisions en rendant fermes les assignations fortement probables.

La robustesse de ces algorithmes a été validée en deux étapes : d'abord lors de l'appariement d'images fixes, ensuite pour créer des séquences de partitions en vue du suivi d'objets. Pour ces dernières, nous avons mis en jeu un nouvel élément,

- le *graphe de mémoire* qui, en suivant l'évolution des régions occultées, offre la possibilité de reconnaître un objet s'il réapparaît après occultation.

De plus, nous arrivons à traiter une large variété de séquences sans avoir besoin de l'in-

formation de mouvement. Ceci rend nos résultats indépendants des erreurs qui apparaissent couramment lors de l'estimation des vecteurs de mouvement.

Partie III : Mise en œuvre d'une Application

Dans le cadre du projet M4M (MPEG f(o)ur Mobiles), nous avons abordé la mise en œuvre d'une application de vidéophonie. Il s'agissait d'être capables de détecter, segmenter et suivre un personnage en temps réel, ceci constituant une lourde contrainte à surmonter. Nos priorités donc ont dû évoluer vers la conception d'algorithmes robustes mais surtout rapides. Nos apports ici comportent la conception

- d'une *procédure d'initialisation*, capable de détecter et segmenter le locuteur dans les premières images, et
- d'un *algorithme de segmentation récursif* ayant en charge le suivi du masque tout au long de la séquence.

Nos travaux dans la procédure d'initialisation se sont basés sur une étude préliminaire où nous avons mené à terme des recherches concernant la création d'un modèle de la couleur de la peau et d'un modèle approchant la silhouette du locuteur. Sur ces bases, nous avons bâti deux approches. La première est complètement automatique grâce à l'incorporation d'une étape d'adaptation du modèle chromatique aux conditions particulières de l'image. La deuxième vise à une réduction du temps de calcul et remplace cette étape par une phase d'interaction avec l'utilisateur. L'atout principal de ces méthodes est en rapport avec leur robustesse, car elles sont capables de travailler dans des environnements non contrôlés.

Le suivi du locuteur est fait en suivant une approche récursive. Nos apports allant de la conception du système jusqu'à l'optimisation des techniques de filtrage et de segmentation développées dans la première partie de cette thèse. Ainsi, en développant des algorithmes simples et efficaces et en les mettant en œuvre nous avons valorisé nos approches théoriques et montré leur viabilité dans une finalité plus industrielle.

12.2 Perspectives

Différentes parties de cette thèse permettent d'envisager des travaux futurs adressés à l'amélioration des algorithmes, soit au niveau du filtrage, de la segmentation ou du suivi, la plupart d'entre eux étant en relation avec des études d'optimisation et de gain en robustesse, qui ont fait le sujet de remarques précises lors des conclusions de chaque chapitre.

A plus long terme, nous avons une prédilection claire pour ce qui concerne la poursuite des recherches dans le domaine de la mise en correspondance de partitions par graphes. Cette nouvelle approche, que nous estimons très prometteuse, fournit un support idéal pour l'analyse des séquences.

Dans la ligne de nos travaux, il nous paraît intéressant d'approfondir l'étude de ces algo-

rithmes en vue du *suivi d'objets*. Rappelons brièvement ici l'ensemble des améliorations qui ont été proposées suite à l'étude de nos résultats :

- *Incorporation d'un modèle d'objet.*
- *Exploitation d'autres attributs.*
- *Détection et correction automatique d'erreurs.*
- *Contrôle de la taille du graphe de mémoire.*

Bien que la mise en œuvre de ces démarches puisse rendre le système plus performant, il nous paraît aussi important d'envisager l'extension de cette méthode à d'autres domaines d'application, comme par exemple :

- *Recherche d'images dans des bases de données* : il s'agirait d'étudier ici la viabilité de la méthode comme élément de support des moteurs de recherche classiques, le processus de mise en correspondance donnant une approche fine pour mesurer la similitude entre les images au niveau de leur contenu.

- *Analyse sémantique de séquences* : beaucoup d'informations, qui n'ont pas été exploitées lors du suivi d'objets, découlent de l'étude détaillée de l'évolution du graphe de mémoire. On pourrait, par exemple, étudier l'importance des changements dans la scène ou la corrélation existante entre des plans différents de la séquence.

- *Adaptation du codage au contenu* : dans une nouvelle génération de codeurs, capables de tirer profit d'une description a priori de la séquence, la mise en correspondance par graphes donnant une information supplémentaire sur les changements qui se produisent dans le contenu global de la scène.

Si la mise en correspondance de partitions reste un problème qui est loin d'être résolu, nous espérons que l'ensemble des travaux qui ont été développés au cours de cette thèse auront permis d'évoluer vers la solution.

Bibliographie

[1] Abdel-Mottaleb, M. et Elgammal, A. Face detection in complex environments from color images. Dans *IEEE International Conference on Image Processing, ICIP'99*, Kobe, Japon, Octobre 1999.

[2] Albiol, A., Bouman, C.A., et Delp, E.J. Face detection for pseudo-semantic labeling in video databases. Dans *IEEE International Conference on Image Processing, ICIP'99*, Kobe, Japon, Octobre 1999.

[3] Astola, J., Haavisto, P., et Nuevo, Y. Vector median filters. *Proceedings of the IEEE*, 78(4) :678–689, Avril 1990.

[4] Barnett, V. The ordering of multivariate data. *Journal of the Royal Statistical Society, Series A*, 139 :318–354, 1976.

[5] Barron, J.L., Fleet, D.J., et Beauchemin, S.S. Systems and experiment performance of optical flow techniques. *International Journal of Computer Vision*, 12(1) :43–77, 1994.

[6] Barrow, H.G. et Brustall, R.M. Subgraph isomorphism, matching relational structures and maximal cliques. *Information Processing Letters*, 4(4) :83–84, 1976.

[7] Barrow, H.G. et Popplestone, R.J. Relational descriptions in picture processing. *Machine Intelligence*, pages 377–396, 1971.

[8] Bergen, L. *Analyse de l'Ordre de Profondeur des Objets dans une Scène par l'Étude du Mouvement*. Thèse doctorale, Centre de Morphologie Mathématique, Ecole des Mines de Paris, 1999.

[9] Beucher, S. Road segmentation by watershed algorithms. Dans *PROMETHEUS Workshop*, Sophia-Antipolis, France, Avril 1990.

[10] Beucher, S. *Segmentation d'Images et Morphologie Mathématique*. Thèse doctorale, Ecole des Mines de Paris, 1990.

[11] Beucher, S. et Lantuéjoul, C. Use of watersheds in contour detection. Dans *Intenational Workshop on image processing, real-time edge and motion detection/estimation*, Rennes, France, Septembre 1979.

[12] Bolles, R.C. et Fischler, M.A. Recognizing and locating partially visible objects, the local-feature focus method. *International Journal on Robotics Res.*, 1(3) :57–82, 1982.

[13] Boyer, K. et Kak, A. Structural stereopsis for 3D vision. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 10 :144–166, 1988.

[14] Brice, C.R. et Fenema, C.L. Scene analysis using regions. *Artificial Intelligence*, pages 205–226, 1970.

[15] Bunke, H. On a relation between graph edit distance and maximum commom subgraph. *Pattern Recognition Letters*, 18 :689–694, 1997.

[16] Bunke, H. Error correcting graph matching : On the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9) :917–922, Septembre 1999.

[17] Bunke, H. et Allerman, G. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4) :245–253, 1983.

[18] Caplier, A. et Mottin, N. Détection automatique de lèvres dans un visage parlant. Dans *Journées d'études et d'échanges sur la Compression et Représentation des Signaux Audiovisuels, CORESA'99*, Sophia Antipolis, Juin 1999.

[19] Chalom, E. et Bove, V.M. Segmentation of an image sequence using multi-dimensional image attributes. Dans *IEEE International Conference on Image Processing*, pages 525–528, Laussane, Suise, 1996.

[20] Chellapa, R., Wilson, C., et Sirohey, S. Human and machine recognition of faces : a survey. *Proccedings of the IEEE*, 83(5) :705–740, Mai 1995.

[21] Chirstmas, W., Kittler, J., et Petrou, M. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8) :749–764, 1995.

[22] Chleq, N., Brémond, F., et Thonnant, M. *Advanced Video-based Surveillance Systems*, chapitre Image understanding for prevention of vandalisme in metro stations, pages 106–116. Kluwer International Series in Engineering and Computer Science. Kluwer Accademic Publishers, 1999.

[23] Cross, A.D.J., Wilson, R.C., et Hancock, E.R. Inexact graph matching using genetic search. *Pattern Recognition*, 30(6) :953–970, 1997.

[24] Cuisenaire, O. et al. Model-based segmentation and recognition of anatomical brain structures in 3d mr images. Dans *Noblesse Model-Based Image Analysis Workshop, NMBIA'98*, pages 9–14, Glasgow, UK, July 1-3 1998.

[25] Falçao, A. et al. User-steering image segmentation paradigms : Live wire and live lane. *Graphical Models and Image Processing*, 60(4) :233–260, Juillet 1998.

[26] Feng, J., Laumy, M., et Dhome, M. *Pattern Recognition in Practice IV : Multiple Paradigms, Comparative Studies, and Hybrid Systems*, chapitre Inexact Matching Using Neural Networks, pages 177–184. North-Holland, 1994.

[27] Friedlander, F. *Le Traitement Morphologique d'Images de Cardiologie Nucléaire*. Thèse doctorale, Centre de Morphologie Mathématique, Ecole des Mines de Paris, 1989.

[28] Garey, M.R. et Johnson, D.S. *Computers and Intractability : A Guide to the Theory of the NP-Completeness*. Freeman and Company, San Francisco, CA, 1979.

[29] Garrido, L., Salembier, P., et Casas, J.R. Representing and retrieving regions using binary partition trees. Dans *IEEE International Conference on Image Processing, ICIP'99*, volume 2, pages 605–609, Kobe, Japan, Octobre 1999.

[30] Gelgon, M. et Bouthemy, P. A region-level motion-based graph representation and labeling for tracking a spatial image partition. *Pattern Recognition*, 33(4) :725–740, Avril 2000.

[31] Gomes, J. et Velho, L. *Image Processing for Computer Graphics*. Springer, 1997.

[32] Gomila, C. et Meyer, F. Détection et suivi du locuteur dans des séquences de vidéophonie ou vidéoconférence. Dans *Journées d'études et d'échanges sur la Compression et Representation des Signaux Audiovisuels, CORESA'99*, Sophia-Antipolis, France, June 1999.

[33] Gomila, C. et Meyer, F. Levelings in vector spaces. Dans *Proceedings of the IEEE International Conference on Image Processing, ICIP'99*, volume 2, pages 929–933, Kobe, Japon, Octobre 1999.

[34] Gomila, C. et Meyer, F. Tracking of video objects for videophony applications. Dans *Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'99*, pages 97–100, Berlin, Allemagne, Mai 1999.

[35] Gomila, C. et Meyer, F. Automatic video object generation tool : segmentation and tracking of persons in real-time. *Annales des Télécommunications*, 55(3-4) :172–183, Mars-Avril 2000.

[36] Gomila, C. et Meyer, F. Tracking objects by graph matching of image partition sequences. Dans *3rd IAPR-TC15 Workshop on Graph-based Representations, GbR2001*, Ischia, Italie, Mai 2001.

[37] Gomila, C. et Meyer, F. Tracking objects by joint segmentation and matching of image partition sequences. submitted to the Pattern Recognition Letters (special issue), Septembre 2001.

[38] Gondran, M. et Minoux, M. *Graphes et algorithmes*. Collection de la Direction des Etudes et Recherches d'Electricité de France. Editions Eyrolles, 1995.

[39] Gonzalez, R. et Woods, R. *Digital Image Processing*. Addison Wiley, 1991.

[40] Hanbury, A.G. et Serra, J. Morphological operators on the unit circle. Sousmis dans *IEEE Transactions on Image Processing*.

[41] Haralick, R.M. et Shapiro, L.G. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1) :100–132, Janvier 1985.

[42] Herault, L., Horaud, R., Veillon, F., et Niez, J.J. Symbolic image matching by simulated aneeling. Dans *Proceedings of the British Machine Vision Conference*, pages 319–324, Oxford, Royaume Unit, 1990.

[43] Horaud, R. et Skordas, T. Stereo correspondance through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11) :1168–1180, Novembre 1989.

[44] Horn, B.K.P. et Schunk, B.G. Determining optical flow. *Artifitial Intelligence*, 17 :185–203, 1981.

[45] Horowitz, S.L. et Pavlidis, T. Picture segmentation by directed split and merge procedure. Dans *Proceedins 2nd Int. Joint Conf. Pattern Recognition*, pages 424–433, 1974.

[46] Hummel, R.A. et Zucker, S.W. On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 5(3) :267–287, Mai 1983.

[47] Jain, J.R. et Jain, A.K. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communications*, 29(12) :1799–1806, Decembre 1981.

[48] Koller, D., Daniilidis, K., et Nagel, H. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3) :257–281, 1993.

[49] Lu, S.W., Ren, Y., et Suen, C.Y. Hierarchical attributed graph representation and recognition of handwritten chinese characters. *Pattern Recognition*, 24 :617–632, 1991.

[50] Marcotegui, B. *Segmentation de Séquences d'Images en Vue du Codage*. Thèse doctorale, Centre de Morphologie Mathématique, Ecole des Mines de Paris, 1996.

[51] Marcotegui, B. et al. A video object generation tool allowing friendly user interaction. Dans *IEEE International Conference on Image Processing, ICIP'99*, volume 2, pages 391–395, Kobe, Japon, Octobre 1999.

[52] Marichal, X. On-line web application using image segmentation. Dans *Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'99*, pages 141–144, Berlin, Allemagne, Mai 1999.

[53] Marqués, F. et al. Partition-based image representation as basis for user-assisted segmentation. Dans *IEEE International Conference on Image Processing, ICIP'00*, volume 1, pages 312–315, Vancouver, Canada, Septembre 2000.

[54] Marqués, F. et Gutiérrez, G. Shape representation for image retrieval based on subgraph matching and fourier descriptors. Dans *Proceedings of the Ninth European Signal Processing Conference, EUSIPCO'98*, volume 2, pages 585–589, Rhodas, Grèce, 1998.

[55] Marqués, F. et Llach, J. Tracking of generic objects for video object generation. Dans *Proceedings of the IEEE International Conference on Image Processing, ICIP'98*, pages 191–206, Chicago, USA, 1998.

[56] Marqués, F., Vilaplana, V., et Buxes, A. Human face segmentation and tracking using connected operators and partition projection. Dans *Proceedings of the IEEE International Conference on Image Processing*, Kobe, Japon, Octobre 1999.

[57] Matheron, G. Les nivellements. Rapport technique, Centre de Morphologie Mathématique, 1997.

[58] MEDEA project A116 full proposal. M4M - MPEG fo(u)r mobiles, MPEG4 encoding and decoding for mobile communications. Avril 1998.

[59] Meier, T. et Ngan, K.N. Automatic segmentation of moving objets for video object plane generation. *Invited paper in IEEE Trans. on Circuits and Systems for Video Technology*, 8(5) :525–538, Septembre 1998.

[60] Messmer, B. et Bunke, H. *Graphics Recognition : Lecture Notes in Computer Science*, volume 1072, chapitre Automatic learning and recognition of graphical symbols in engineering drawing, pages 123–134. Springer, Berlin, 1996.

[61] Meyer, F. Un algorithme optimal de ligne de partage des eaux. Dans *Proceedings 8ème Congrès AFCET*, pages 847–457, Lyon-Villeurbanne, France, 1991.

[62] Meyer, F. Connected morphological operators for binary images. Rapport technique, Centre de Morphologie Mathématique, 1997.

[63] Meyer, F. From connected operators to levelings. Dans Heijmans, H. et Roerdink, J., editeurs, *Mathematical Morphology and its Applications to Image and Signal Processing, Proc. ISMM'98*, pages 191–198. Kluwer Academic Publishers, 1998.

[64] Meyer, F. The levelings. Dans Heijmans, H. et Roerdink, J., editeurs, *Mathematical Morphology and its Applications to Image and Signal Processing, Proc. ISMM'98*, pages 191–198. Kluwer Academic Publishers, 1998.

[65] Meyer, F. Vectorial levelings and flattenings. Dans Goutsias, J., Vincent, L., et Bloomberg, D.S., editeurs, *Mathematical Morphology and its Applications to Image and Signal Processing, Proc. ISMM'00*, pages 51–60, Palo Alto, USA, June 2000.

[66] Meyer, F. Hierarchies of partitions and morphological segmentation. Dans *Proc. IEEE Workshop on Variational and Level Set Methods in Computer Vision*, Vancouver, Canada, Juillet 2001.

[67] Meyer, F. et Beucher, S. Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1(1) :21–46, Septembre 1990.

[68] MPEG home page, http ://www.cselt.it/mpeg.

[69] Nagel, H.H. Displacement vectors derived from second-order intensity variations in image sequences. *Computer Vision, Graphics, and Image Processing*, 21(1) :85–117, Janvier 1983.

[70] Niblack, W., Barber, R., Equitz, W., Flickner, M., Glasman, E., Petkovic, D., Yanker, P., et Faloutsos, C. The QBIC project : Querying images by content using color, texture, and shape. Dans *Proc. SPIE, Conference on Storage and Retrieval for Image and Video Databases*, volume 1908, pages 173–187, San José, California, USA, Février 1993.

[71] Pal, N.R. et Pal, S.K. A review on image segmentation techniques. *Pattern Recognition*, 26(9) :1277–1294, 1993.

[72] Palovic, V.I., Sharma, R., et Huang, T.S. Visual interpretation of hand gestures for human-computer interaction : a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7) :677–692, Juillet 1997.

[73] Pardàs, M. et Salembier, P. 3D morphological segmentation and motion estimation for image sequences. Dans *EURASIP Signal Processing*, volume 38, pages 31–43, Septembre 1994.

[74] Pardàs, M. et Salembier, P. Time-recursive segmentation of image sequences. Dans *VII European Signal Processing Conference, EUSIPCO'94*, pages 18–21, Edinburgh, , Royaume Unit, Septembre 1994.

[75] Pavlidis, T. Representation of figures by labeled graphs. *Pattern Recognition*, 4 :5–17, 1972.

[76] Pelillo, M. Replicator equations, maximal cliques and graph isomorphism. *Neural Computation*, 11(8) :1933–1955, 1999.

[77] Pereira (Guest Editor), F. *Image Communication, Tutorial Issue on MPEG-4 standard*, volume 15 (4-5). Elsevier, Janvier 2000.

[78] Pitas, I. et Tsakalides, P. Multivariate ordering in color image filtering. *IEEE Transactions on Circuits and Systems for Video Technology*, 1(3) :247–259, 1991.

[79] Pratt, W.K. *Digital Image Processing*. Willey, 2nd edition, 1991.

[80] Price, K.E. Relaxation matching techniques — a comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 7(5) :617–623, 1985.

[81] Read, R.C. et Corneil, D.G. The graph isomorphism disease. *J. Graph Theory*, 1 :339–363, 1977.

[82] Rosenfeld, A., Hummel, R.A., et Zucker, S.W. Scene labelling by relaxation opperations. *IEEE Transactions on Systems, Man and Cybernetics*, 6 :420–433, 1976.

[83] Saber, E., Tekalp, A.M., Eschbanch, R., et Knox, K. Automatic image annotation using adaptive color classification. *Graphical Models and Image Processing*, 58(2) :115–126, 1996.

[84] Salembier, P. et Serra, J. Flat zones filtering, connected opperators, and filters by reconstruction. *IEEE Transactions on Image Processing*, 3(8) :1153–1160, Août 1995.

[85] Sanfeliu, A. et Fu, K.S. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13 :353–363, 1983.

[86] Serra, J. *Image Analysis and Mathematical Morphology*, volume 1. Academic Press, London, Royaume Unit, 1982.

[87] Serra, J. *Image Analysis and Mathematical Morphology : Theoretical Advances*, volume 2. Academic Press, London, Royaume Unit, 1988.

[88] Serra, J. *Anamorphoses and function lattices (multivalued morphology) in Mathematical Morphology in Image Processing*, chapitre 13, pages 483–523. Dekker, New York, USA, 1993.

[89] Serra, J. Connections for sets and functions. *Fundamenta Informaticae*, 41 :147–186, 2000.

[90] Serra, J. et Salembier, P. Connected operators and pyramids. Dans *Proceedings of the SPIE, Image algebra and morphological image processing IV*, volume 2030, pages 65–76, San Diego,USA, Juillet 1993.

[91] Shapiro, L.G. et Haralick, R.M. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 3 :504–519, Septembre 1981.

[92] Shapiro, L.G. et Haralick, R.M. A metric for comparing relational descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7 :90–94, 1985.

[93] Siddiqi, K., Shokoufandeh, A., Dickinson, S.J., et Zucker, S.W. Shock graphs and shape matching. Dans *Sixth International Conference on Computer Vision*, Bombay, Inde, 1998.

[94] Smith, J.R. *Integrated Spatial and Feature Image Systems : Retrieval, Analysis and Compression.* Thèse doctorale, Columbia University, 1997.

[95] Stevens, M.R., Beveridge, J.R., et Goss, M.E. Reduction of brl/cad models and their use in automatic target recognition algorithms. Dans *Proceedings of the BRL-CAD Symposium, Army Research Labs.*, Aberdeen Proving Ground, USA, June 1995.

[96] Terrillon, J.C. et Fukamachi, H. Comparative performance of different chrominance spaces for color segmentation and detection of human faces in complex scene images. Dans *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 112–117, Nara, Japon, 14–16 Avril 1998.

[97] Torres, L. et Kunt Editors, M. *Video Coding : The Second Generation Approach.* Kluwer Academic Publishers, 1996.

[98] Torres, L., Lleida, E., et Casas, J.R. *Sistemas analógicos y digitales de televisión.* Edicions UPC, 1993.

[99] Ullman, J.R. An algorithm for subgraph isomorphism. *Journal of the Assotiation for Computing Machinery*, 23(1) :31–42, 1976.

[100] Vachier, C. *Extraction de Caractéristiques, Segmentation d'Image et Morphologie Mathématique.* Thèse doctorale, Centre de Morphologie Mathématique, Ecole des Mines de Paris, 1995.

[101] Vincent, L. et Soille, P. Watersheds in digital spaces : An efficient algorithm based on immersion simulations. *Pattern Analysis and Machine Intelligence*, 13(6) :583–598, Juin 1991.

[102] Wang, Y.K., Fan, K.C., et Horng, J.T. Genetic-based search for error-correcting graph isomorphism. *IEEE Transactions on Systems, Man and Cybernetics*, 27(4) :588–597, 1997.

[103] Weska, J.S., Nagel, R.N., et Rosenfeld, A. A threshold selection technique. *IEEE Transations on Computers*, C-23(12) :1322–1326, Decembre 1974.

[104] Wilson, R., Cross, D.J., et Hancock, E.R. Structural matching with active triangulations. *Computer Vision and Machine Understanding*, 72(1) :21–38, Octobre 1998.

[105] Wilson, S.S. Theory of matrix morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14 :636–652, 1992.

[106] Wison, R. et Hancock, E.R. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6) :634–648, Juin 1997.

[107] Yoo, J.R. et Huang, T. Image segmentation by unsupervised clustering and its applications. TR-EE 18–19, Perdue University, 1978.

[108] Zanoguera, F., Marcotegui, B., et Meyer, F. A toolbox for interactive segmentation based on nested partitions. Dans *Proceedings of the IEEE International Conference on Image Processing*, volume 1, pages 21–25, Kobe, Japon, Octobre 1999.

[109] Zanoguera, F., Marcotegui, B., et Meyer, F. A segmentation pyramid for the interactive segmentation of 3D images and video sequences. Dans Goutsias, J., Vincent, L., et Bloomberg, D.S., editeurs, *Mathematical Morphology and its Applications to Image and Signal Processing*, pages 223–232, Palo Alto, USA, 2000. Kluwer Academic Publishers.

[110] Zucker, S.W. Region growing : childhood and adolescence. *Computer Graphics and Image Processing*, 5 :382–399, 1976.

[111] Zucker, S.W., Krishnamurthy, E.V., et Haar, R.L. Relaxation processes for scene labeling : Convergence, speed and stability. *IEEE Transactions on Systems, Man and Cybernetics*, 8(1) :41–48, Janvier 1978.