Université François Rabelais
Tours
Ecole Doctorale : Santé, Sciences et Technologies
Année Universitaire : 2001-2002

## THESE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE TOURS

Discipline : **Informatique**

Présentée et soutenue publiquement
par :

**Sébastien LEFEVRE**

**le 13 décembre 2002**

# DETECTION D'EVENEMENTS

# DANS UNE SEQUENCE VIDEO

Directeurs de thèse : Nicole VINCENT, Christian PROUST

**Jury**

| | | | |
|---|---|---|---|
| **Brunet** Christophe | Ingénieur | Invité | Atos Origin Blois |
| **Miguet** Serge | Professeur | Examinateur | Université de Lyon II |
| **Nicolas** Henri | Chargé de Recherches (HdR) | Examinateur | IRISA Rennes |
| **Postaire** Jack-Gérard | Professeur | Rapporteur | Université de Lille I |
| **Proust** Christian | Professeur | Directeur, Invité | Université de Tours |
| **Sedes** Florence | Professeur | Rapporteur | Université de Toulouse III |
| **Stamon** Georges | Professeur | Président | Université de Paris V |
| **Vincent** Nicole | Professeur | Directeur | Université de Tours |

*A mon père, qui m'a montré la voie*

# Remerciements

Tout travail de thèse, pour qu'il puisse être mené à bien, nécessite un encadrement de qualité. Je remercie donc Nicole Vincent pour son encadrement sans faille et ses nombreux conseils qui m'ont permis d'avancer.

Plus généralement, je remercie également tous les membres du Laboratoire d'Informatique de l'Université de Tours, et plus particulièrement son directeur, Christian Proust, pour m'avoir accueilli et intégré au sein d'une structure de recherche dynamique et enthousiaste. Cet environnement de qualité a eu une influence directe sur le travail que j'ai effectué pendant la thèse. Dans ce cadre, je tiens à remercier aussi les étudiants de l'E3i (devenue département informatique de l'Ecole Polytechnique de l'Université de Tours) avec qui j'ai eu l'occasion de travailler.

Pendant trois ans, mes recherches n'ont pu être effectuées que grâce au soutien de l'entreprise Atos Origin, sous forme d'un contrat Cifre. J'adresse ici mes remerciements à mes anciens collègues d'Atos Origin avec qui les échanges ont toujours été très fructueux.

Pour avoir accepté de rapporter mon travail, et pour leurs remarques constructives, je tiens à remercier Florence Sédès et Jack-Gérard Postaire. Je remercie également les autres membres du jury, Serge Miguet, Henri Nicolas, et finalement Georges Stamon pour m'avoir fait l'honneur de le présider.

*Last but not least*, un très grand merci à Séverine et Elsa, qui m'ont permis, chacune à leur manière, de passer les caps difficiles et de toujours garder espoir. Sans elles, je n'aurais pu atteindre ce résultat.

# Sommaire

# Notations

| | |
|---|---|
| $X$ | Hauteur d'une image en nombre de pixels |
| $Y$ | Largeur d'une image en nombre de pixels |
| $C$ | Nombre de composantes de l'espace couleur (généralement $C = 3$) |
| $T$ | Nombre de trames d'une séquence vidéo |
| $N$ | Nombre de classes d'un histogramme |
| $B$ | Nombre de blocs d'une image |
| $\Gamma$ | Nombre d'itérations d'un processus itératif |
| | |
| $x$ | Coordonnée horizontale dans l'espace spatial 2-D ($1 \leq x \leq X$) |
| $y$ | Coordonnée verticale dans l'espace spatial 2-D ($1 \leq y \leq Y$) |
| $c$ | Indice dans l'espace couleur utilisé ($1 \leq c \leq C$) |
| $t$ | Indice dans l'espace temporel ($1 \leq t \leq T$) |
| $n$ | Valeur d'un niveau de gris dans un histogramme ($1 \leq n \leq N$) |
| $b$ | Indice d'un bloc dans une image ($1 \leq b \leq B$) |
| $\gamma$ | Indice de l'itération dans un processus itératif ($1 \leq \gamma \leq \Gamma$) |
| $r$ | Indice du niveau de résolution à laquelle une image est analysée (depuis $r = 0$ pour l'image originale jusqu'à $r = r_{\text{max}}$ pour l'image avec la résolution la plus faible) |
| | |
| $d$ | Mesure de distance calculée entre deux variables (par exemple des images) |
| $S$ | Valeur de seuil |
| | |
| $I$ | Image composée d'un ensemble de $X \times Y$ pixels |
| $\dot{I}$ | Image compressée à l'aide de la transformée en cosinus discrète et contenant un ensemble de $X \times Y$ coefficients |
| $I_t$ | Trame de la séquence vidéo obtenue à l'instant $t$ |
| $I^b$ | Bloc $b$ d'une image $I$ |

| | |
|---|---|
| $\underset{r}{I}$ | Représentation de l'image $I$ à la résolution $r$ |
| $P(x, y)$ | Pixel de coordonnées $(x, y)$ |
| $I(x, y)$ | Intensité du pixel de l'image $I$ situé à la position spatiale $(x, y)$ |
| $I(x, y, c)$ | Valeur de la composante couleur $c$ du pixel de l'image $I$ situé à la position spatiale $(x, y)$ |
| $H(I)$ | Histogramme d'une image $I$ |
| $H(I, n)$ | Nombre de pixels dans l'image $I$ ayant leur niveau de gris égal à $n$ |
| $H(I, n, c)$ | Nombre de pixels dans l'image $I$ ayant leur valeur, pour la composante couleur $c$, égale à $n$ |
| $E(x, y)$ | Energie d'un point de coordonnées $(x, y)$ |
| $\mathcal{O}$ | Complexité d'un algorithme en nombre maximal d'opérations |
| $\mathcal{P}$ | Nombre de pixels d'une image $I$ utilisés dans les calculs de complexité |
| $\mathcal{N}$ | Nombre de classes d'un histogramme $H$ utilisées dans les calculs de complexité |
| $\mathcal{B}$ | Nombre de blocs d'une image $I$ utilisés dans les calculs de complexité |
| $*$ | Opération de convolution |
| $\Delta_t$ | Intervalle (voisinage) temporel |
| $\Delta_s$ | Intervalle (voisinage) spatial |
| $\mathcal{V}_\Delta(x)$ | Voisinage de $x$ de diamètre $\Delta$ (parfois omis), où $x$ peut désigner un point ou un instant |

# Introduction

Le **multimédia** joue un rôle croissant dans la société contemporaine. Grâce à la puissance de calcul des processeurs actuels, il est possible de gérer de plus en plus de données. Les utilisateurs des nouvelles technologies de l'information et de la communication peuvent maintenant consulter et transmettre non seulement des textes, mais aussi des images, des sons, ou encore des séquences vidéo. Ces différents média peuvent contenir des données synthétiques (graphique dans le cas d'un dessin, animation dans le cas d'une vidéo) ou naturelles (photo dans le cas d'une image, film dans le cas d'une vidéo).

La quantité d'informations disponibles étant de plus en plus importante, il est nécessaire de disposer d'outils performants permettant l'indexation, la recherche, puis la visualisation des informations. En effet, «trop d'information tue l'information». Les utilisateurs souhaitent consulter uniquement l'information qui les intéresse à un instant choisi. Dans le cadre de cette thèse de doctorat, nous considérons uniquement le problème de **l'indexation de données multimédia**. Une fois le processus d'indexation effectué, il est alors possible de rechercher l'information pertinente en fournissant des critères de recherche appropriés, puis de la visualiser d'une manière adéquate (affichage textuel pour un texte, affichage graphique pour une image ou une vidéo, émission sonore pour un son, *etc.*).

L'indexation de données multimédia fait l'objet de nombreux travaux de recherche. De nombreux numéros spéciaux réalisés par des revues scientifiques [Lit95, Pen96, Pan96, Nga98, Bim99, Wol99b, Man00, Jia01, Dje02b, Dje02a] se sont consacrés à ce problème. Dans le cadre de cette thèse nous nous limiterons à **l'indexation de séquences vidéo**. Les séquences vidéo sont composées d'une suite d'images qui peut être accompagnée d'une piste audio. L'essentiel de nos travaux concernera l'indexation de séquences vidéo principalement basée sur l'analyse des images la composant. L'indexation de données sonores ne sera que partiellement étudiée et nous n'aborderons pas le thème de l'indexation de données textuelles.

On distingue habituellement deux types de système d'indexation de séquences vidéo. Il existe d'une part **les systèmes dits génériques** qui permettent d'obtenir une classification des

différentes séquences vidéo disponibles sans prendre en compte des informations de nature contextuelle. Ces systèmes permettent, par exemple, de classer les différentes séquences vidéo en fonction de la scène (intérieure ou extérieure), de la caméra (statique ou en mouvement), *etc.*. D'autre part, **les systèmes dits spécifiques** ne permettent d'indexer qu'un type bien particulier de séquences vidéo, comme par exemple les journaux télévisés, les vidéos de surveillance, les événements sportifs tels que les matchs de football ou de tennis, *etc.*. Dans ce cas, **l'indexation est contextuelle** car basée sur une **problématique précise**, et le résultat obtenu répond aux attentes des utilisateurs qui peuvent par exemple vouloir visualiser la météo présentée à la télévision le 20 décembre 1999, ou bien vérifier si des intrusions ont eu lieu dans l'usine le week-end dernier, ou encore voir les moments importants du dernier match de la Coupe Davis. Les systèmes spécifiques, même si leur utilisation est limitée à un type de séquence vidéo, permettent cependant de répondre à de nombreuses demandes de la part d'utilisateurs de systèmes d'indexation vidéo. Notre travail portera donc sur des systèmes spécifiques à un type de séquences vidéo.

Il est également possible de classer les systèmes d'indexation vidéo selon qu'ils fonctionnent en **temps réel** ou non. Nous employons ici le terme «temps réel» au sens large et nous limitons son sens au traitement des données à une fréquence similaire à celle de l'acquisition. C'est ainsi que dans le cas de séquences vidéo, on parle souvent de systèmes temps réel lorsque le traitement est effectué en moins de 40 millisecondes par image alors que la séquence est acquise à une fréquence de 25 images par seconde. Certaines applications ne nécessitent pas toujours la contrainte de temps réel, par exemple pour indexer le fonds des séquences télévisées de l'INA (Institut National de l'Audiovisuel). Dans notre cas, nous considérons cependant cet objectif de temps réel comme une contrainte forte car les applications visées nécessitent une réactivité très importante du système d'indexation vidéo.

Les systèmes spécifiques d'indexation vidéo dépendent donc fortement du contexte défini au préalable. L'indexation des séquences vidéo peut être vue comme une classification binaire de l'ensemble des images. Soit la suite d'images correspond à l'événement prédéfini, soit elle n'y correspond pas. Le problème revient alors à la **détection d'événements prédéfinis dans des séquences vidéo**, ou de scénarios comme dans [Nag88]. L'outil idéal dans le domaine de l'indexation selon une problématique donnée serait un logiciel qui fonctionnerait en deux étapes. Après une phase d'apprentissage des séquences vidéo correspondant ou non à l'événement prédéfini, une étape de reconnaissance permettrait de classer chaque suite d'images en deux classes : celles qui correspondent à l'événement prédéfini et celles qui n'y correspondent pas. Cette étape de reconnaissance serait bien sûr basée sur le processus d'apprentissage effectué dans un premier temps. Cependant, la réalisation d'un tel système d'apprentissage et de

reconnaissance relève encore aujourd'hui de l'utopie et l'état de la recherche dans le domaine de l'indexation vidéo ne permet pas la création d'un tel logiciel.

Dans le cadre de nos travaux, nous proposons une approche plus modeste qui consiste en la proposition d'**une architecture** et d'**un ensemble d'outils** permettant de **créer des systèmes d'indexation spécifiques**. Cependant, afin de pouvoir s'adapter à de nombreux contextes différents, l'architecture et les outils se doivent d'être **les plus génériques possibles**. La spécificité intervient alors uniquement lors de la réalisation du système final qui respecte l'architecture générique et intègre les différents outils proposés. Mais pour cela, il est nécessaire que les outils utilisés respectent les contraintes de l'application. Nous avons donc recensé **un ensemble de contraintes** notées $\mathcal{C}_{\text{i}}$ qui nous paraissent importantes dans un système d'indexation vidéo quelconque :

$\mathcal{C}_{\text{rapidité}}$ : le traitement doit s'effectuer en **temps réel** sur une architecture informatique standard (de type micro-ordinateur PC) ;

$\mathcal{C}_{\text{couleur}}$ : les images analysées sont stockées en **couleur**, ce qui implique un coût supplémentaire par rapport à des séquences en niveaux de gris ;

$\mathcal{C}_{\text{illumination}}$ : les conditions d'éclairage peuvent varier du fait d'une acquisition en intérieur ou en extérieur et des différentes sources de lumière rencontrées, et de ce fait des changements d'**illumination** peuvent apparaître dans les images ;

$\mathcal{C}_{\text{mouvement}}$ : les images analysées peuvent cependant contenir des **mouvements importants** (caméra mobile, facteur de zoom élevé, *etc.*) ou non (caméra fixe, facteur de zoom faible, *etc.*) et les paramètres d'acquisition (translation, rotation, zoom de la caméra) sont le plus souvent inconnus ;

$\mathcal{C}_{\text{compression}}$ : les séquences vidéo à traiter peuvent être **non-compressées ou compressées** (dans ce dernier cas nous considérons les normes Motion-JPEG ou MPEG [Che93]).

Selon les cas, toutes ou partie de ces contraintes devront être prises en compte pour indexer les séquences vidéo.

Une fois la séquence vidéo indexée, deux cas de figure se présentent : pour chaque suite d'images considérée, soit elle correspond à l'événement prédéfini, soit elle n'y correspond pas. L'information est généralement intéressante uniquement dans le premier cas (sauf lorsqu'on cherche «l'événement contraire»). Selon les cas, il peut alors être nécessaire de la transmettre à l'utilisateur. La transmission de séquences vidéo fait partie du domaine de la **communication de données multimédia** faisant l'objet de travaux de recherches spécifiques que nous n'aborderons pas ici. Nous nous limiterons à choisir et intégrer les moyens de communication les plus adéquats selon l'application visée.

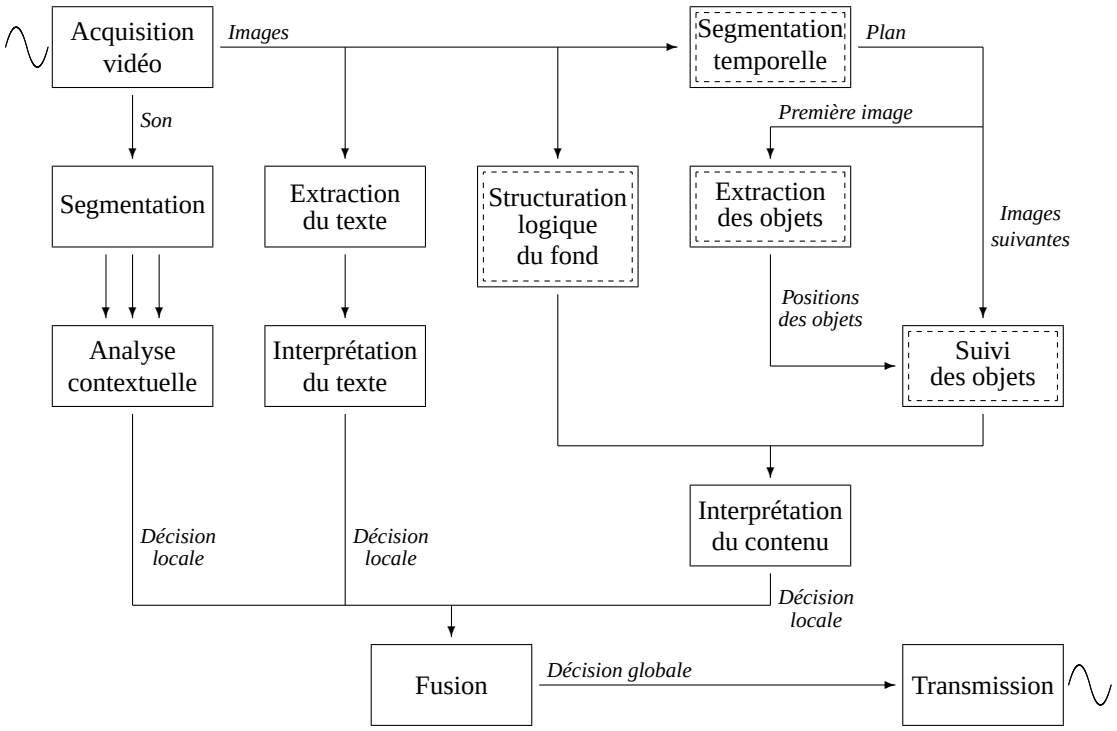


FIG. 1 – Schéma d'une architecture générique pour les systèmes d'indexation vidéo.

Dans ce mémoire, nous proposons l'architecture illustrée par la figure 1 car elle prend en compte les différents média contenus dans une vidéo :

- la bande son que nous proposons d'analyser après avoir extrait un certain nombre de types d'intervenants fonction de l'application ;
- le texte qui apparaît ici sous forme d'incrustation dans les images et qui doit donc être extrait avant d'être reconnu ;
- les images qui dans leur ensemble peuvent être regroupées en séquences plus homogènes (les différents plans) ;
- de plus l'interprétation de la séquence est réalisée après extraction des objets en mouvement et analyse de la scène ;
- la reconnaissance d'un événement prédéfini doit reposer sur les 3 types d'information ainsi extraits (il serait évidemment souhaitable qu'un conflit entre les différentes sources conduise à remettre en cause les résultats précédents).

Cette architecture, que nous avons voulue la plus générique possible, permet la mise en place de systèmes d'indexation vidéo spécifiques, tels ceux demandés par l'entreprise **Atos Origin**. Comme le montre la figure 1, la mise en place d'un tel système nécessite de résoudre des

problèmes très variés qui présentent tous des difficultés. Nous avons choisi de n'aborder au cours de cette thèse qu'une partie des problèmes recensés. Ainsi, nous nous limiterons aux problèmes représentés en pointillés dans la figure. Ils concernent la segmentation temporelle, l'extraction des objets, la structuration du fond, et enfin le suivi des objets. Ces problèmes seront abordés respectivement dans les parties I, II et III.

Plus précisément, ce mémoire s'organisera de la façon suivante. Dans une **première partie** nous aborderons le problème de la **segmentation temporelle** des séquences vidéo. Nous assimilons ce problème à celui de la détection des changements de plans. Ce domaine faisant l'objet de nombreux travaux, nous présenterons tout d'abord une classification possible de l'ensemble des méthodes, ainsi que les états de l'art publiés dans la littérature. Ensuite, nous proposerons une approche rapide et robuste aux conditions d'éclairage, permettant de traiter des séquences vidéo compressées ou non. Une fois les différents plans identifiés, il sera alors possible de les analyser plus en détail.

L'analyse des différentes images d'un plan peut être focalisée soit sur la scène soit sur les objets présents dans la scène. Dans la **seconde partie** de ce mémoire, nous proposerons une **étude de l'arrière-plan de la scène** lors de laquelle sera notamment abordé le problème de la séparation des objets et du fond. Nous proposerons une approche multirésolution permettant un traitement rapide même si la caméra est en mouvement. Nous présenterons aussi nos travaux concernant la structuration du fond, obtenue par une détection des lignes présentes dans l'image. L'étude de l'arrière-plan de la scène permettra donc, d'une part, de séparer les objets du fond (et de ce fait pourra servir d'initialisation à des algorithmes de suivi d'objet), et d'autre part, d'obtenir un modèle ou une structure de l'arrière-plan, donnant ainsi la possibilité de replacer les positions des objets suivis dans la scène pour effectuer ensuite une interprétation.

Après avoir étudié l'arrière-plan, nous aborderons dans une **troisième partie** le problème du **suivi d'objet**. Selon le type d'objet à suivre (rigide ou non-rigide) et la possibilité d'effectuer un apprentissage ou non, nous proposerons différentes méthodes reposant notamment sur les chaînes de Markov cachées et les contours actifs.

Enfin, dans une **quatrième et dernière partie**, nous présenterons les différents contextes pour lesquels nous avons réalisé des systèmes d'indexation spécifiques, qui sont basés sur l'architecture et les outils proposés ici. Trois **applications** seront décrites. La première concerne un système de détection de buts dans des matchs de football, tandis que les deux suivantes répondent aux demandes d'Atos Origin qui a financé cette thèse sous forme de contrat CIFRE. Plus précisément, les deux systèmes concernent, d'une part, l'obtention de statistiques de fréquentation d'un site et, d'autre part, la vidéosurveillance d'une salle informatique sécurisée.

# Première partie

# Segmentation temporelle

La **segmentation temporelle** peut être considérée comme l'une des premières étapes nécessaires à l'indexation de séquences vidéo. La plupart du temps, cette segmentation est obtenue par **détection des changements de plans**. Une fois les différents plans de la séquence obtenus, il est alors possible de les analyser indépendamment les uns des autres avec des outils adéquats. Ainsi, l'arrière-plan de la scène et les objets en mouvement peuvent être étudiés, de manière indépendante sur chaque sous-séquence, par les outils présentés respectivement dans les parties II et III.

Dans cette partie nous dresserons tout d'abord un panorama des méthodes de détection des changements de plans (chapitre 1). Le nombre de méthodes présentées dans la littérature est important. Nous les avons étudiées sous l'angle des contraintes énoncées dans l'introduction de notre mémoire. Nous verrons qu'aucune ne répond totalement à notre problématique. Nous nous attacherons à développer une méthode originale (chapitre 2) qui privilégie les aspects suivants : $\mathcal{C}_{\text{rapidité}}$, $\mathcal{C}_{\text{couleur}}$, $\mathcal{C}_{\text{illumination}}$, $\mathcal{C}_{\text{mouvement}}$, $\mathcal{C}_{\text{compression}}$.

# Chapitre 1

# Panorama des méthodes de détection des changements de plans

La détection des changements de plans permet de segmenter dans le domaine temporel une séquence vidéo. Chaque plan obtenu peut alors être analysé avec différents outils comme ceux présentés dans les parties II et III. La détection des changements de plans se doit donc d'être rapide et précise, puisqu'elle apparaît comme un prétraitement.

Dans ce chapitre nous dresserons un **panorama des différentes méthodes publiées dans la littérature**. Tout d'abord nous donnerons une définition du changement de plans et nous présenterons les différentes formes qu'il peut prendre. Nous décrirons ensuite les différentes mesures permettant d'évaluer la qualité des méthodes de détection d'un changement de plans. Puis nous proposerons une classification des différentes méthodes et donnerons les grands principes de chaque type d'approches. Enfin, nous examinerons les états de l'art abordant ce domaine. Il nous semble plus intéressant de présenter ces états de l'art plutôt que de réaliser une bibliographie qui ne pourrait qu'être non exhaustive dans le cadre d'un mémoire de thèse. En effet, nous avons recensé plus de 300 articles rien que dans le domaine de la détection des changements de plans.

## 1.1 Description des changements de plans

Un **plan** est défini comme une suite d'images issues d'une acquisition continue d'une caméra donnée. Ainsi, toutes les images d'un plan ont été acquises avec la même caméra. Le plan est souvent l'unité temporelle la plus petite pour une séquence vidéo si l'on ne prend pas en

compte l'image pour laquelle la notion de temps a disparu. On utilise souvent le terme "scène" pour définir un ensemble d'un ou plusieurs plans filmés successivement et représentant le même environnement.

Chaque plan est séparé du précédent et du suivant par une **transition**. On distingue deux types de transitions : les transitions brusques et les transitions progressives. Lors d'une transition brusque (appelée *cut*), la dernière image du premier plan est directement suivie par la première image du second plan. Aucun effet n'est inséré entre les deux plans, comme le montre la figure 1.1. Dans le cas où les deux plans sont connectés en utilisant un effet particulier, on parle de transition progressive. Différents types de transitions peuvent être utilisés. Les plus connus sont le fondu et le volet. On distingue le fondu du noir vers un plan, d'un plan vers le noir, ou d'un plan vers un autre plan. Au cours d'un fondu, le niveau de chaque pixel des images intermédiaires (appartenant à la transition progressive) est calculé en fonction des niveaux des pixels de la dernière image du premier plan et de la première image du second plan. La proportion varie au cours de la transition de 0 à 1 pour la première image du second plan et de 1 à 0 pour la dernière image du premier plan. Lors d'un volet, chaque pixel des images intermédiaires a un niveau égal à celui du pixel de mêmes coordonnées spatiales soit dans la dernière image du premier plan soit dans la première image du second plan. Les images appartenant à un volet vont donc contenir de plus en plus de pixels extraits de la première image du second plan et de moins en moins de pixels extraits de la dernière image du premier plan. Les figures 1.2 et 1.3 donnent respectivement un exemple de fondu et de volet.

FIG. 1.1 – Exemple d'une transition brusque.

FIG. 1.2 – Exemple d'une transition progressive de type fondu.

FIG. 1.3 – Exemple d'une transition progressive de type volet.

Les différentes méthodes se proposent de détecter ces différents changements. Pour comparer ces différentes méthodes, il est nécessaire de disposer de mesures pour évaluer leur qualité respective.

## 1.2 Evaluation de la qualité des méthodes proposées

Différentes mesures existent pour évaluer cette qualité. La plus utilisée reste **le taux de reconnaissance** de détection des changements, pour lequel on rencontre pourtant différentes définitions. Des travaux ont été effectués pour définir des mesures standards et pour discuter les mesures existantes [Gar96, Eic00, Ham95, Rui99]. Parmi toutes ces mesures, les mesures de précision et de rappel sont les plus communes. Le terme de **précision** désigne le rapport entre le nombre de changements correctement détectés et le nombre de changements détectés (détections correctes ou fausses). La mesure de **rappel** est le rapport entre le nombre de changements correctement détectés et le nombre de changements réellement présents dans la séquence. Ces deux mesures permettent de comparer différentes méthodes. La plupart des méthodes reposent sur un ensemble de paramètres que l'expert fixe empiriquement dans une phase de mise au point, en fonction d'un ensemble de tests. La variation de ces paramètres de la méthode permet de faire évoluer les mesures rappel et précision. On peut analyser les graphiques donnant la valeur de la précision en fonction du rappel. Un exemple est donné dans la figure 1.4. On juge souvent que la méthode A est considérée comme meilleure que la méthode B.

Bien que le taux de reconnaissance reste le critère le plus utilisé pour évaluer la qualité d'une méthode, d'autres mesures peuvent être employées. Ainsi, il est possible de prendre en compte **le nombre de paramètres ou seuils à fixer** pour une méthode donnée et les possibilités d'apprentissage de ces paramètres. Plusieurs auteurs proposent une procédure d'apprentissage afin d'utiliser une valeur de seuil appropriée [Ard00, Dre99]. Cependant la plupart des méthodes de détection des changements de plans utilisent des paramètres ou seuils fixés de manière empirique. Ces seuils constituent des paramétrages spécifiques au type de séquences vidéo analysées,

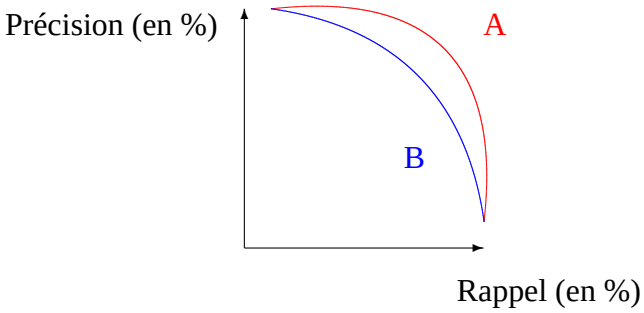


FIG. 1.4 – Exemple de graphiques donnant le rappel et la précision de 2 méthodes.

et ne permettent donc pas une acquisition de données non contrainte ni une méthode générique, c'est-à-dire indépendante du domaine des séquences vidéo traitées. La qualité des méthodes peut alors être évaluée selon le caractère adaptatif ou non de ces seuils.

Enfin un autre critère à prendre en compte concerne **la complexité algorithmique** des méthodes. Ce critère est de la plus haute importance lorsque la segmentation temporelle doit s'effectuer en temps réel sur du matériel informatique standard. Dans ce cas, le temps de calcul dépend en effet directement de la complexité algorithmique. Quelques articles dans la littérature proposent d'utiliser ce critère pour comparer un ensemble de méthodes, limité [Dai95] ou relativement exhaustif [Lef02e, Lef02f]. Précisons tout d'abord les notations employées. En considérant deux fonctions $f$ et $g$, nous définissons la relation $f = \mathcal{O}(g)$ quand est vérifiée la propriété suivante :

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1 \tag{1.1}$$

Nous exprimerons la complexité en fonction de $\mathcal{P}$, $\mathcal{N}$, et $\mathcal{B}$ qui représentent respectivement le nombre de pixels dans une image, le nombre de classes dans un histogramme, et le nombre de blocs dans une image.

Les différents critères présentés ici permettent de comparer les méthodes. Nous avons pour notre part établi une classification hiérarchique des méthodes tout d'abord selon le type de séquences vidéo (compressées ou non) et ensuite suivant le type d'informations utilisées pour réaliser la segmentation.

## 1.3 Classification des méthodes

La plupart des méthodes proposées pour résoudre le problème de la détection des changements de plans fonctionnent en deux étapes : le calcul d'une mesure de dissimilarité entre deux trames successives d'une séquence vidéo, puis la comparaison de la valeur obtenue avec un seuil, afin de déterminer ou non la présence d'un changement de plans. Suivant ce principe, la détection d'un changement de plans est effective si la condition suivante est respectée :

$$d(I_t, I_{t-1}) > S \tag{1.2}$$

Nous rappelons que $I_t$ représente l'image de la séquence vidéo obtenue à l'instant $t$, $d$ une distance, et $S$ un seuil. Dans le cas d'image en niveaux de gris, on note le domaine $\mathcal{I} = [1, X] \times [1, Y] \subset \mathbb{N} \times \mathbb{N}$ dans lequel est défini la fonction $I$. Une définition similaire est donnée pour des images en couleur.

Afin de mieux appréhender ce domaine de détection des changements de plans, il est nécessaire de classer les différentes méthodes en plusieurs catégories. La première famille de méthodes concerne celles travaillant sur des séquences vidéo non-compressées. Dans ce cas, on dispose des valeurs (en niveaux de gris ou en couleur) des pixels composant les différentes images de la séquence vidéo. La seconde famille de méthodes rassemble les méthodes dédiées aux séquences vidéo compressées. Ces méthodes considèrent la plupart du temps des standards de compression vidéo [Che93], tels que le MPEG [LeG91] et le MJPEG (ou Motion-JPEG). Ce dernier, même s'il n'est pas un standard proprement dit, est associé à une compression indépendante de chaque trame de la séquence selon le standard JPEG [Wal91]. Dans cette section nous proposons une classification des méthodes pour chaque type de séquences vidéo (non-compressées et compressées). Davantage d'informations sur la compression sont fournies dans les ouvrages de Watkinson [Wat01] d'une part, et de Barlaud et Labit [Bar02] d'autre part.

### 1.3.1 Séquences vidéo non-compressées

La plupart des méthodes de détection des changements de plans sont basées sur l'utilisation de séquences vidéo non-compressées. Nous proposons ici une classification des différentes méthodes selon le type ou la forme des informations utilisées. Les méthodes peuvent être basées sur les pixels, les histogrammes, un découpage en blocs, des caractéristiques robustes extraites dans l'image, une information liée au mouvement ou encore une combinaison de ces différentes approches.

**Méthodes basées sur les pixels**

Les premières méthodes comparent deux images successives en utilisant directement les informations des **pixels**. Ainsi, différentes mesures plus ou moins robustes ont été progressivement proposées tout le long des recherches, parmi lesquelles on peut citer :

- la différence de l'énergie (mesurée par la somme des carrés des intensités) des deux images,
- la somme des différences absolues terme à terme pour des images en niveaux de gris ou en couleur,
- l'énergie des différences normalisées.

Il est également possible de déterminer, dans un premier temps, pour chaque couple de pixels appartenant aux deux images si leurs intensités sont proches ou non. La mesure de différence est alors obtenue, dans un second temps, en comptant le nombre de couples ayant des intensités éloignées. Des approches plus évoluées ont également été proposées. Ainsi, l'analyse de l'évolution de l'intensité de chaque pixel au cours du temps peut permettre d'étiqueter le pixel en question selon différentes classes : constant, changement brusque, changement linéaire, ou sans étiquette. Un changement est alors détecté en comptant le nombre de pixels ayant une étiquette donnée. Finalement, l'évolution des dérivées temporelles des intensités des pixels peut aussi être utilisée comme critère de détection. La complexité de ces méthodes varie de $\mathcal{O}(\mathcal{P})$ à $\mathcal{O}(6\mathcal{P})$ pour des images en niveaux de gris. Ce type de méthode, quoique simple, s'est avéré peu robuste au bruit et aux mouvements présents dans la séquence.

**Méthodes basées sur les histogrammes**

Afin de pallier au manque de robustesse des méthodes basées sur les pixels, certains auteurs ont proposé d'utiliser des statistiques plus globales, les **histogrammes**. De la même manière que dans les approches à base de pixels, il est possible de calculer la différence terme à terme en considérant les classes de l'histogramme afin de comparer deux images successives. Cette différence peut être calculée en considérant les images en niveaux de gris ou en couleur, dans un espace relativement limité (par exemple 2 bits pour chaque composante couleur) ou dans l'espace original. Dans le cas d'images couleur, la mesure de distance entre deux images peut être calculée comme la plus grande des différences entre deux histogrammes de même couleur. D'autres mesures de distance peuvent être utilisées, telles que la mesure de similarité du cosinus, la distance quadratique, le test du $\chi^2$, l'entropie croisée, la divergence, la distance de Kullback-Liebler ou encore la distance de Bhattacharya. La statistique de Kolmogorov-Smirnov peut aussi être employée en utilisant des histogrammes cumulatifs. Dans le cas d'images cou-

leur, il est possible de favoriser certaines composantes apportant plus ou moins d'informations dans l'image. Le choix des coefficients pondérateurs peut être effectué empiriquement, proportionnellement à la luminance, ou encore après une étape d'apprentissage. Il est enfin possible de calculer l'intersection entre deux histogrammes de deux images successives. Plus celle-ci sera faible, plus la probabilité d'être en présence d'un changement de plans sera importante. Le test le plus employé pour ce type de méthode est le suivant [Nag91] :

$$\sum_{n=1}^{N}\sum_{c=1}^{C}|H(I_t,n,c)-H(I_{t-1},n,c)|>S \tag{1.3}$$

Dans le cas des images en niveaux de gris, la complexité des méthodes à base d'histogramme varie de $\mathcal{O}(2\mathcal{N})$ à $\mathcal{O}(11\mathcal{N})$ auquel il faut ajouter $\mathcal{O}(\mathcal{P})$ pour la construction de l'histogramme. Le principal défaut de ce type de méthode est leur caractère purement global.

**Méthodes basées sur les blocs**

L'analyse des groupes d'images successives, dans la détection des changements de plans, peut aussi être effectuée à un niveau intermédiaire entre le niveau local (les pixels) et le niveau global (les histogrammes). Les méthodes entrant dans cette catégorie analysent l'image par **blocs**. De cette façon, il est possible de comparer deux images en mesurant les différences pour chaque paire de blocs *via* un calcul de la moyenne et de l'écart-type des intensités pour chaque bloc. Une fois chaque paire de blocs étiquetée selon leur similarité, la détection dépend du nombre de paires de blocs différents. Il est aussi possible de calculer entre deux blocs le taux de vraisemblance de Yakimovsky ou la statistique de Freund pour mesurer leur similarité. Certaines approches à base d'histogrammes ont été appliquées à un traitement de l'image par blocs. Dans ce cas, on calcule pour chaque bloc son histogramme et on le compare avec celui du bloc correspondant dans l'image précédente. En considérant l'ensemble des blocs ou seulement ceux dont la différence avec le bloc précédent est la plus faible, il est possible de mesurer la similarité entre deux images successives. La formule suivante donne un exemple de détecteur de changements de plans, relativement similaire à [Swa93], basé sur une analyse des histogrammes calculés sur les blocs de l'image :

$$\sum_{b=1}^{B}\sum_{n=1}^{N}\sum_{c=1}^{C}\left|H(I_t^b,n,c)-H(I_{t-1}^b,n,c)\right|>S \tag{1.4}$$

Ce type de méthode est un bon compromis entre les approches purement locales et les approches globales, justifiant une complexité algorithmique variant entre $\mathcal{O}(\mathcal{P}+5\mathcal{B})$ et $\mathcal{O}(10\mathcal{P})$ selon

la méthode employée. Certains auteurs ont cependant proposé des méthodes basées sur des caractéristiques plus robustes que la moyenne ou l'écart-type d'un bloc.

#### Méthodes basées sur des caractéristiques robustes

Les méthodes basées sur des **caractéristiques** robustes calculées sur les trames de la séquence vidéo sont supposées offrir de meilleurs taux de détection des changements de plans. Elles sont également caractérisées par des complexités plus élevées. Les caractéristiques utilisées pour détecter les changements de plans peuvent être les moments calculés sur l'image, les contours extraits de l'image, des points caractéristiques obtenus par l'utilisation de la transformée de Hough, des mesures statistiques. Nous plaçons aussi dans cette catégorie les approches bayésiennes, celles basées sur l'utilisation de chaînes de Markov cachées, ainsi que celles qui modélisent explicitement les différentes transitions. Certains auteurs proposent d'effectuer une Analyse en Composantes Principales (ACP), d'autres une décomposition en valeurs singulières. Les approches que nous avons étudiées sont caractérisées par des complexités algorithmiques de $\mathcal{O}(3\mathcal{P}+\mathcal{N})$ à $\mathcal{O}(27\mathcal{P}+3\mathcal{N})$ (pour la méthode décrite dans [Yu97]). Les différentes caractéristiques utilisées par les méthodes de cette catégorie sont de nature uniquement spatiale.

#### Méthodes basées sur le mouvement

Comme les séquences vidéo fournissent également une information de nature temporelle, certains auteurs ont proposé d'utiliser les informations de **mouvement** présentes dans la séquence. En effet, le mouvement apparent des pixels est généralement continu au cours du temps. Il peut donc être utilisé comme critère pour la détection d'un changement de plans. Ainsi, il est possible d'estimer dans chaque image le mouvement global qui peut par exemple être dû au mouvement de la caméra effectuant l'acquisition des images, puis de comparer ces informations dans les images successives. Deux trames successives d'une séquence vidéo caractérisées par des mouvements globaux différents peuvent être assimilées aux limites d'un plan. Des approches basées sur la technique bien connue de *block matching* (utilisée notamment dans le codage MPEG) ont aussi été proposées : pour chaque bloc dans une image donnée, on recherche la position de coïncidence optimale dans l'image suivante. Si le nombre de recherches infructueuses est trop élevé, un changement de plans est détecté. De même, il est possible de calculer le flux optique pour chaque image et de comparer les résultats obtenus. Ces différentes méthodes, quoique plus adaptées à la nature temporelle des séquences vidéo, sont souvent caractérisées par des complexités algorithmiques élevées [Bou99], rendant alors impossible une exécution en temps réel. Lorsque le temps de calcul n'est pas une contrainte à respecter, il est

aussi possible de combiner différentes approches.

#### Combinaison de méthodes

Cette dernière catégorie de méthodes travaillant sur des séquences vidéo non-compressées s'accompagne de temps de calcul importants. L'intérêt de ces méthodes réside dans le fait qu'en **combinant différentes approches** plus simples, on peut obtenir de meilleurs résultats puisqu'on dispose des avantages des différentes approches [Qué99], comme par exemple une approche basée sur les pixels et une approche basée sur les histogrammes. Les différentes techniques sont soit exécutées successivement avec des seuils de tolérance de plus en plus élevés, soit simultanément en définissant une règle de fusion des décisions obtenues, par exemple un "ET" ou un "OU". La complexité mesurée pour ce type de méthodes varie de $\mathcal{O}(3\mathcal{P} + 3\mathcal{N})$ à $\mathcal{O}(26\mathcal{P} + 5\mathcal{N})$.

Les différentes méthodes décrites précédemment sont liées à une analyse des séquences vidéo non-compressées. Dans le cas de séquences vidéo compressées, d'autres méthodes spécifiques ont été proposées.

### 1.3.2 Séquences vidéo compressées

Les méthodes travaillant sur des séquences vidéo compressées supposent généralement une compression vidéo suivant les normes MJPEG ou MPEG. Après avoir présenté brièvement ces deux normes, nous décrirons les différentes méthodes dédiées aux données compressées en suivant une classification en trois parties : les méthodes liées aux coefficients issus de la TCD, celles basées sur les vecteurs de mouvement, et enfin les autres approches.

La norme **MJPEG** ou Motion-JPEG est un standard de compression qui consiste en l'adaptation pour la vidéo du JPEG, dédié à la compression des images. Dans une séquence vidéo MJPEG, chaque trame est compressée en suivant le standard JPEG (*Joint Picture Expert Group*). Cette compression est basée principalement sur la Transformée en Cosinus Discrète (TCD). D'autres traitements sont aussi effectués, comme une étape de quantification ou un codage de Huffman. Afin d'appliquer la TCD sur une image donnée, l'image est partitionnée en blocs de $8 \times 8$ pixels. Une fois la TCD calculée pour un bloc, 64 valeurs sont obtenues. La première représente l'information de basse-fréquence et est appelée coefficient DC. Les 63 restantes sont nommées coefficients AC. Dans le cas d'images couleur, les informations de luminance et de chrominance doivent être traitées. L'œil humain étant plus sensible aux changements de luminance que de chrominance, ces données ne sont pas compressées de la même manière. La

plupart du temps, la résolution de la luminance est quatre fois plus fine que celle de la chrominance. Les images sont alors compressées dans l'espace YCbCr, utilisant un ratio de $4:1:1$.

Plusieurs standards **MPEG** (*Motion Picture Expert Group*) existent pour la compression vidéo. La plupart des méthodes de détection des changements de plans travaillant sur des séquences vidéo compressées selon une norme MPEG supposent que la norme utilisée est le MPEG-1. Dans ce cas, une séquence vidéo est composée d'un ensemble de groupes d'images (GOP). Un GOP peut contenir trois types de trames appelées I, P, et B. Les trames I (*intra-coded*) sont compressées de la même manière que les trames MJPEG ou les images JPEG. Les trames P (*forward-predicted*) sont codées par prédiction avant, c'est-à-dire en utilisant les trames I et P précédentes et un processus de compensation de mouvement. Les trames B (*bidirectionnally-predicted*) sont codées par prédiction bidirectionnelle (souvent appelée interpolation) en utilisant les trames I et P les plus proches dans les deux sens (précédent et suivant). L'information de mouvement n'est donc présente que dans les trames P et B d'une séquence vidéo MPEG. Les trames vidéo MPEG contiennent différents types de macroblocs : *intracoded*, *forward-predicted*, *backward-predicted*, et *bidirectionnally-predicted*. Les macroblocks *intra-coded* suivent le schéma de compression basé sur la TCD. Les macroblocks *forward-predicted* et *backward-predicted* contiennent chacun un vecteur de mouvement, tandis que les macroblocs *bidirectionnally-predicted* en contiennent deux. Un macrobloc est dit *skipped* lorsqu'il a un vecteur de mouvement nul. Les trames I ne sont composées que de macroblocs *intracoded* et ne contiennent pas d'information de mouvement. Les trames P contiennent des macroblocs *intracoded* et *forward-predicted*. Enfin, les trames B contiennent tous les types de macroblocs.

#### Méthodes basées sur les coefficients de la TCD

Puisque les informations contenues dans les séquences vidéo MJPEG ou MPEG sont codées par les **coefficients DC et AC** (relatifs à une notation anglo-saxonne) issus de la TCD, certains auteurs ont proposé de détecter les changements de plans par analyse de ces coefficients dans les trames successives [Zha95a]. Ainsi, il est possible de calculer le produit scalaire entre deux vecteurs de caractéristiques, chaque vecteur étant composé des coefficients DC d'une image donnée. Si le produit scalaire est faible, une analyse des histogrammes couleur peut alors être effectuée pour confirmer la présence d'un changement. Une autre approche consiste en la comparaison bloc-à-bloc de deux images successives, toujours *via* l'analyse des coefficients de la TCD. Le nombre de paires de blocs différents permet de déterminer si un changement de plans est présent ou non. Le traitement précédent peut être accéléré en ne prenant en compte que les coefficients DC au lieu de l'ensemble des coefficients (DC et AC).

En utilisant uniquement les coefficents DC, il est aussi possible de construire des histogrammes basés sur une représentation faible résolution des images. Les méthodes à base d'histogrammes présentées dans la section précédente peuvent alors s'appliquer. Les histogrammes peuvent être calculés à partir de l'image globale, ou alors sur chaque ligne ou chaque colonne de l'image. D'autres mesures, telles que la moyenne et la variance de l'intensité des images, peuvent être calculées à partir des coefficients DC et utilisées pour détecter les changements de plans. Comme dans le cas des histogrammes, ces mesures peuvent être calculées sur l'image globale ou alors selon les directions horizontale et verticale. Dans le cas de séquences MPEG, les coefficents DC ne sont disponibles que dans les trames I. Certains auteurs ont proposé d'estimer ces coefficients dans les autres types de trames (P et B), permettant ainsi une analyse plus précise puisque toutes les trames de la séquence sont alors considérées. D'autres proposent d'analyser les images dans un autre espace de représentation, soit de dimension plus faible, soit de nature spatio-temporelle. Dans ce dernier cas la détection d'un changement de plans est assimilée à la détection d'une ligne verticale ou oblique dans la nouvelle image obtenue. Finalement, d'autres auteurs proposent d'utiliser différents outils mathématiques comme les B-splines, les ondelettes, ou la transformée de Radon pour détecter les transitions présentes dans les séquences vidéo compressées. La complexité de ce type de méthodes varie de $\mathcal{O}(3\mathcal{B})$ à $\mathcal{O}(6\mathcal{P} + 19\mathcal{B})$. Pour celles basées sur les histogrammes (qui nécessitent un coût d'au moins $\mathcal{O}(\mathcal{B})$ pour le calcul de l'histogramme), la complexité maximale rencontrée dans la littérature est de $\mathcal{O}(18\mathcal{N})$. Toutes ces méthodes sont limitées à l'utilisation de l'information spatiale contenue dans les séquences vidéo. Dans le cas de séquences codées selon la norme MPEG, des informations temporelles sont aussi disponibles et peuvent de ce fait être utilisées.

**Méthodes basées sur les vecteurs de mouvement**

Ainsi, des approches basées sur l'utilisation de **vecteurs de mouvement** ont été proposées dans la littérature [Kob98]. Contrairement au cas de données non-compressées, l'extraction des informations de mouvement ne s'accompagne pas ici de temps de calcul importants puisque celles-ci sont déjà incluses dans le codage MPEG des séquences vidéo. La plupart des approches appartenant à cette catégorie associent la présence ou non d'un changement de plans à la quantité de tel ou tel type de macroblocs. Ainsi, si la proportion de macroblocs liés à une prédiction (avant, arrière, ou bidirectionnelle) est faible par rapport au nombre total de macroblocs de l'image, un changement de plans est détecté. Il est possible d'effectuer des analyses différentes pour les différents types de trames. Dans le cas des trames B, on pourra par exemple calculer le maximum entre le nombre de macroblocs de type *forward-predicted* et le nombre de macroblocs de type *backward-predicted*, puis comparer la valeur obtenue avec un seuil. Cer-

taines approches détectent des changements de plans dans une trame $I_t$ par analyse des trames voisines $I_{t'}$ avec $t' \in \mathcal{V}_\Delta(t)$ où $\mathcal{V}_\Delta(t)$ désigne un voisinage de $t$ de diamètre $\Delta$. De nombreuses heuristiques basées sur l'analyse des types de macroblocs présents dans les trames de la séquence vidéo ont été proposées. En ce qui concerne les approches basées sur les vecteurs de mouvement, les trames I ne sont pas analysées puisqu'elles ne contiennent pas d'information temporelle. Certains auteurs proposent donc d'estimer les vecteurs de mouvement dans ces images afin d'analyser toutes les trames d'une séquence vidéo. Il est possible de combiner cette analyse des informations de mouvement contenues dans les macroblocs avec l'information liée à la TCD, c'est-à-dire les coefficients DC et AC. Les différentes approches décrites ci-dessus se caractérisent par des complexités allant de $\mathcal{O}(\mathcal{B})$ à $\mathcal{O}(\mathcal{P} + 2\mathcal{N} + 4\mathcal{B})$. D'autres approches ne peuvent entrer ni dans la catégorie des méthodes basées sur les coefficients de la TCD, ni dans celle des méthodes basées sur les informations de mouvement.

#### Autres méthodes

Nous définissons alors une dernière catégorie regroupant les **autres méthodes** travaillant sur des séquences vidéo compressées. Parmi ces approches, on peut citer des approches utilisant la quantification vectorielle, la combinaison de différentes approches comme on l'a déjà vu dans le cas de séquences vidéo non-compressées, la taille de codage des différentes trames car la présence d'un changement de plans est directement liée à l'espace mémoire nécessaire pour coder la trame analysée, des informations de contour estimées avec ou sans décompression des trames, des opérations morphologiques, une décomposition en sous-bandes, des vecteurs de caractéristiques, des réseaux de neurones, *etc.*. Pour les approches que nous avons étudiées, nous notons des complexités comprises entre $\mathcal{O}(4\mathcal{B})$ (pour la méthode décrite dans [Fen96]) et $\mathcal{O}(12\mathcal{N} + 27\mathcal{B})$.

Comme on peut le constater, le nombre de méthodes de détection des changements de plans, que ce soit dans le domaine non-compressé ou compressé, est pour le moins important. Dresser un état de l'art exhaustif dans le cadre de ce mémoire n'est pas notre objectif. Les complexités des différentes catégories de méthodes présentées ici sont rassemblées dans le tableau 1.1 page 20. Les deux colonnes de droite représentent les complexités des méthodes caractérisées par les coûts les plus faibles (*coût min*) et les coûts les plus élevés (*coût max*). Plusieurs auteurs ont aussi proposé un état de l'art du domaine de la détection des changements de plans. Nous allons les retrouver dans la section suivante. Nous avons par ailleurs réalisé deux états de l'art respectivement dans le domaine non-compressé [Lef02f] et dans le domaine compressé [Lef02e].

| Classe de méthodes | Compression | Complexité algorithmique | |
|---|---|---|---|
| | | coût min | coût max |
| Pixels | non | $\mathcal{O}(\mathcal{P})$ | $\mathcal{O}(6\mathcal{P})$ |
| Histogrammes | non | $\mathcal{O}(2\mathcal{N})$ | $\mathcal{O}(11\mathcal{N})$ |
| Blocs | non | $\mathcal{O}(\mathcal{P}+5\mathcal{B})$ | $\mathcal{O}(10\mathcal{P})$ |
| Caractéristiques | non | $\mathcal{O}(3\mathcal{P}+\mathcal{N})$ | $\mathcal{O}(27\mathcal{P}+3\mathcal{N})$ |
| Mouvement | non | $\mathcal{O}(20\mathcal{P})$ | $\mathcal{O}(105\mathcal{P})$ |
| Combinaison d'approches | non | $\mathcal{O}(3\mathcal{P}+3\mathcal{N})$ | $\mathcal{O}(26\mathcal{P}+5\mathcal{N})$ |
| Coefficients de la TCD | oui | $\mathcal{O}(3\mathcal{B})$ | $\mathcal{O}(6\mathcal{P}+19\mathcal{B})$ |
| Vecteurs de mouvement | oui | $\mathcal{O}(\mathcal{B})$ | $\mathcal{O}(\mathcal{P}+2\mathcal{N}+4\mathcal{B})$ |
| Autres | oui | $\mathcal{O}(4\mathcal{B})$ | $\mathcal{O}(12\mathcal{N}+27\mathcal{B})$ |

TAB. 1.1 – Complexité des différentes méthodes de détection des changements de plans.

## 1.4 Description des états de l'art

Plusieurs états de l'art ont déjà été publiés dans la littérature. Aucun n'est exhaustif mais il est cependant possible d'avoir un aperçu global du domaine en se référant à plusieurs d'entre eux. Certains ont pour objectif de référencer l'ensemble des méthodes élaborées, d'autres se proposent de comparer expérimentalement les différentes approches, et enfin les derniers se focalisent sur un type particulier de méthodes.

– Ainsi, Ahanger *et al.* [Aha96] discutent des besoins et des architectures globales pour l'indexation vidéo. Quelques méthodes de segmentation, principalement liées au domaine non-compressé, sont présentées dans ce cadre. Idris *et al.* [Idr97] traitent de l'indexation d'images et de vidéos, dans les domaines compressé et non-compressé. Les caractéristiques liées aux images et les algorithmes de traitement des vidéos utiles pour l'indexation sont décrits. La détection des changements de plans est un des traitements des vidéos nécessaire pour caractériser une séquence vidéo. Brunelli *et al.* [Bru99] présentent aussi l'indexation vidéo, et décrivent en particulier les algorithmes de détection des changements de plans, concernant principalement des séquences vidéo non-compressées. Aigrain *et al.* [Aig96] proposent une revue des techniques pour l'analyse du contenu des vidéos. La détection des changements de plans dans des données compressées et non-compressées est l'une de ces techniques. Koprinska *et al.* [Kop01] effectuent une revue dans les deux domaines des algorithmes pour la détection des changements de plans et la reconnaissance des opérations de la caméra (translation, rotation, zoom, *etc.*). Lienhart [Lie01] présente les différents types de changements de plans et propose quelques méthodes de détection. La présentation des méthodes ne se base pas sur une classification en méthodes pour des vidéos compressées ou non compressées car l'auteur considère que

la plupart des méthodes travaillant sur des données non-compressées peuvent être adaptées au domaine compressé après décompression partielle. Jiang *et al.* [Jia98] présentent une revue des méthodes classées en trois catégories : les vidéos non-compressées, les vidéos compressées, et les modèles.

– Plusieurs états de l'art ne comparent qu'un panel limité d'algorithmes en se basant sur une implémentation personnelle de l'auteur. Boreczky *et al.* [Bor96] comparent la performance de cinq algorithmes (certains traitant des vidéos non-compressées tandis que d'autres sont dédiés à des vidéos compressées) en utilisant une méthodologie d'évaluation commune. Lienhart [Lie99] compare quelques méthodes et caractérise leur capacité à déterminer correctement le type des changements de plans qui ont été détectés. Dailianas *et al.* [Dai95] comparent plusieurs algorithmes de segmentation dédiés à des séquences vidéo non-compressées. Quelques détails sont donnés quant à la complexité algorithmique des méthodes évaluées. Yusoff *et al.* [Yus00] comparent plusieurs méthodes liées à des vidéos non-compressées et proposent des versions améliorées en utilisant un seuil adaptatif.
– Finalement, quelques articles proposent des revues ne concernant qu'un type de méthodes bien particulier. Ainsi ceux de Gargi *et al.* [Gar96, Gar00] sont dédiés aux méthodes de segmentation temporelle basées respectivement sur des histogrammes couleur et sur les informations de mouvement. Mandal *et al.* [Man99] se concentrent sur les méthodes travaillant dans le domaine compressé.

Malgré le nombre important de méthodes se proposant de résoudre le problème de la détection des changements de plans, il nous a semblé qu'aucune d'entre elles n'était totalement adaptée à notre problématique en respectant les contraintes $\mathcal{C}_i$ énoncées en page 3. Nous verrons donc dans le chapitre suivant comment nous proposons [Lef00b, Lef01d] une solution à ce problème.

# Chapitre 2

# Détection adaptative de changement de plans : Choix de l'espace TSL

Lors du chapitre précédent nous avons dressé un panorama des méthodes de détection des changements de plans proposées dans la littérature. Chacune de ces méthodes possède des avantages et des inconvénients mais aucune d'entre elles ne nous a semblé répondre aux différentes contraintes que nous nous posions, c'est-à-dire celles énoncées en page 3. Aussi proposons-nous ici **une nouvelle méthode [Lef00b, Lef01d]** permettant un traitement temps réel des séquences compressées ou non d'images couleur, robuste aux mouvements importants et aux changements d'illumination.

Dans ce chapitre, nous commencerons par énoncer plus en détail notre problématique. Puis nous effectuerons quelques rappels sur les différents moyens de représenter la couleur dans des images en indiquant les différents espaces de représentation. Nous décrirons notamment l'espace couleur TSL sur lequel notre méthode de détection des changements de plans est basée. Afin de minimiser le temps de calcul nécessaire à l'algorithme, nous proposons de diminuer la résolution spatiale des images. Deux approches peuvent être utilisées, selon que les données sont compressées ou non. Afin d'accroître la robustesse de la méthode dans le cas de mouvements importants, nous proposons d'utiliser une approche différente des méthodes classiques. En effet, alors que celles-ci consistent généralement à calculer une mesure de similarité entre deux images successives puis à comparer la valeur obtenue avec un seuil fixé *a priori*, notre approche ne compare pas directement la mesure de distance mais étudie plutôt l'évolution temporelle de cette mesure. De plus le seuil fixe classique est remplacé par un seuil adaptatif

permettant une autoadaptation de la méthode à n'importe quel type de séquence vidéo. Enfin des résultats montrant l'efficacité de la méthode seront présentés.

## 2.1 Problématique

Lors du choix ou de l'élaboration d'une méthode de détection de changement de plans, il est nécessaire de prendre en compte le **contexte** dans lequel celle-ci sera employée même si l'on souhaite, le plus souvent, disposer d'une méthode aussi générale que possible. Ainsi, la sélection d'une méthode sera guidée par la compression ou non des séquences, la nature des images qui peuvent être en niveaux de gris ou en couleur, les contraintes imposées quant au temps d'exécution (le temps réel n'est pas toujours nécessaire), ou encore la nature même des séquences vidéo. Ainsi, une méthode qui permet la détection des changements de plans dans des documentaires pourra ne pas être adaptée à la détection des transitions dans des retransmissions sportives. En effet, certains contenus de séquence vidéo sont caractérisés par de faibles mouvements alors que l'on peut observer dans d'autres des déplacements importants.

La détection des changements de plans permet une segmentation temporelle des séquences vidéo et différents algorithmes seront appliqués sur les images afin d'interpréter leur contenu en fonction de la sous-séquence. Puique l'objectif de cette thèse est de proposer une architecture et des outils génériques pour l'indexation contextuelle de séquences vidéo de différents types, il est nécessaire que la méthode de détection des changements de plans utilisée permette d'analyser correctement des séquences vidéo les plus variées. Nous nous plaçons donc dans le cas où les informations contenues dans la séquence vidéo sont représentées en couleur. La méthode s'appliquera *a fortiori* sur des séquences en niveaux de gris. De plus, aucune contrainte n'est imposée quant à l'acquisition des images. Plus précisément, cela signifie que la caméra effectuant l'acquisition peut être fixe ou mobile, en intérieur ou en extérieur. La méthode à utiliser doit donc, d'une part gérer des séquences où le mouvement peut être aussi bien faible qu'important, et d'autre part assurer une robustesse aux changements d'illumination. De plus, les séquences à traiter peuvent être compressées ou non. Finalement, puisque l'indexation des séquences vidéo doit être réalisée en temps réel, il est évident que la détection des changements de plans, considérée comme la première étape du traitement global, doit l'être aussi.

Les différentes contraintes à prendre en compte lors de la détection des changements de plans sont donc celles énoncées précédemment : $\mathcal{C}_{\text{rapidité}}, \mathcal{C}_{\text{couleur}}, \mathcal{C}_{\text{illumination}}, \mathcal{C}_{\text{mouvement}}, \mathcal{C}_{\text{compression}}$.

Les méthodes décrites dans le chapitre précédent permettent de gérer tel ou tel cas mais ne sont pas adaptées à une situation où les cinq contraintes énumérées ci-dessus seraient réalisées.

Nous proposons ici une nouvelle méthode [Lef00b, Lef01d] plus adaptée à une telle situation. Puisque la méthode proposée est basée sur la couleur, il est bon de faire quelques rappels sur les différentes représentations possibles de la couleur et ceci d'autant plus que la couleur sera aussi utilisée dans les parties ultérieures de la thèse.

## 2.2 Représentation de la couleur

La couleur est une notion complexe et sa perception par l'homme a fait l'objet de nombreux travaux de recherche depuis ceux de Newton il y a plus de trois siècles. Le but de cette section n'est évidemment pas de faire un état de l'art sur ces travaux mais plutôt de présenter brièvement les différents espaces de représentation couleur que l'on peut utiliser dans le cadre de l'analyse des images et des séquences d'images et qui permettront de justifier les choix que nous avons faits. D'autres travaux sont consacrés à une étude plus détaillée de la théorie de la couleur ou des espaces de représentation couleur, on peut citer en particulier les thèses de Carron [Car95] sur la segmentation d'image dans l'espace TSL, de Vandenbroucke [Van00] sur la définition d'un espace couleur hybride, et la thèse d'état de Bonnardot [Bon01] qui traite notamment d'une étude analytique du système visuel humain. Chang *et al.* ont dressé récemment un panorama des utilisations de la couleur dans le domaine de la segmentation d'images [Cha01a].

### 2.2.1 Rappels sur les espaces couleur

Le codage de la couleur dans des images numériques peut être effectué en utilisant différents espaces de représentation, appelés traditionnellement espaces couleur. Nous présentons ici les principaux espaces définis dans la littérature. Les formules de conversion d'un espace couleur à l'autre sont notamment présentées dans [Van00, For98, Poy99]. Cependant, ces formules sont à considérer avec précaution. En effet, de nombreux auteurs ont proposé leurs propres systèmes de représentation de la couleur et souvent des notations similaires recouvrent des définitions différentes qui ne sont pas toujours précisées. Il est donc difficile d'énoncer une définition unique d'un espace couleur donné.

#### Espaces proposés par la CIE

L'espace **RGB** proposé par la CIE (Commission Internationale de l'Éclairage) est très certainement le système de représentation couleur le plus fréquemment utilisé. Chaque couleur est définie par une combinaison de trois couleurs primaires : Rouge (R), Vert (G), et Bleu (B). Dans cet

espace, le noir et le blanc sont représentés respectivement par les triplets $(R = 0, G = 0, B = 0)$ et $(R = 255, G = 255, B = 255)$ si l'on considère le cas d'images numériques couleur codées sur 24 bits. Il est important de noter que d'autres espaces de représentation RGB ont été définis, notamment par la Federal Communication Commission (FCC) pour le standard de télévision NTSC (National Television Standards Committee) ou par l'European Broadcasting Union (EBU) pour le standard PAL (Phase Alternation by Line).

Un processus de normalisation des composantes du vecteur couleur permet de réduire l'influence de la luminance. L'espace couleur RGB normalisé est noté **rgb**. Dans cet espace, les différentes composantes ne représentent plus qu'une information de chrominance car les composantes sont liées par la contrainte $r + g + b = 1$.

La CIE a aussi défini un autre espace de représentation basé sur des primaires. Il s'agit de l'espace **XYZ** dans lequel la composante Y représente la luminance (c.f. [Van00]) et les deux autres composantes des informations additionnelles relatives à la chrominance.

De même qu'avec l'espace RGB, il est possible de définir un espace XYZ normalisé. Cet espace est noté **xyz** avec $x + y + z = 1$.

#### Espaces luminance-chrominance

En se basant sur les espaces XYZ et xyz, on peut obtenir l'espace **YUV** généralement utilisé dans les codages des séquences au format MPEG. La composante Y est commune à l'espace XYZ et représente donc toujours la luminance, tandis que les composantes U et V contiennent les informations de chrominance.

La CIE a défini également des espaces appelés $\mathbf{L}^*\mathbf{u}^*\mathbf{v}^*$ et $\mathbf{L}^*\mathbf{a}^*\mathbf{b}^*$ fréquemment utilisés dans la littérature. La composante $L^*$ représente ici la luminance tandis que les couples ($u^*$,$v^*$) et ($a^*$,$b^*$) permettent de modéliser la chrominance.

#### Espaces dans le domaine de l'imprimerie

Dans le domaine de l'imprimerie, l'espace **CMY** est souvent utilisé. Cet espace, "complémentaire" de RGB, considère les trois couleurs Cyan (C), Magenta (M), et Jaune (Y), calculées comme les complémentaires de Rouge, Vert, et Bleu.

Il est aussi possible d'utiliser une représentation quadrichromatique **CMYK**, où la composante additionnelle K représente le noir.

**Espaces dans le domaine télévisuel**

Pour les usages du monde télévisuel, plusieurs systèmes ont été définis. Le standard NTSC est basé sur le système $\mathbf{Y'I'Q'}$ de la FCC.

Le standard PAL utilise quant à lui le système $\mathbf{Y'U'V'}$ de l'EBU, relativement similaire au précédent.

Kodak a aussi proposé son propre système, appelé **YCC** ou $\mathbf{YC_1C_2}$.

**Espaces perceptuels**

En se basant fréquemment sur les notions de luminance-chrominance, de nombreux auteurs ont proposé dans la littérature des espaces représentant l'information couleur sous forme de luminance, chroma, et teinte ou encore d'intensité, saturation, et teinte. Ainsi, l'espace $\mathbf{C^*h_{uv}s_{uv}}$ est défini à l'aide de l'espace $L^*u^*v^*$.

De même, la représentation $\mathbf{C^*h_{ab}}$ définie uniquement par deux composantes liées respectivement à l'intensité et à la teinte (donc sans information de saturation) est basée sur l'espace $L^*a^*b^*$.

Travis [Tra91] définit l'espace **TSL** (Teinte, Saturation, Luminance). Les composantes L, S, et T sont calculées successivement à partir des composantes de l'espace RGB car T n'est définie que pour des valeurs non-nulles de S et de L, tandis que S n'est définie que pour des valeurs non-nulles de L.

Gonzalez et Woods [Gon92b] définissent quant à eux l'espace **TSI** en utilisant des formulations différentes des trois composantes. Comme dans l'espace précédent, T est définie si et seulement si S et I sont non nulles, et S est définie si et seulement si I est non nulle.

De nombreuses autres formulations d'espaces basées sur la teinte, la saturation et la luminance sont proposées dans la littérature. Pour un panorama étendu, le lecteur pourra se référer à [Car95] ou à [Van00].

### 2.2.2 TSL : Une représentation naturelle et robuste

Parmi les différents espaces de représentation de la couleur présentés précédemment, certains ont pour but de modéliser au mieux le système de la vision humaine. On peut citer par exemple les espaces TSL ou TSI.

Nous avons fait le choix d'utiliser ces espaces lors de nos travaux. Les différents algorithmes

proposés dans le cadre de cette thèse sont basés sur l'espace TSL de Travis [Tra91]. Avant de définir cet espace, nous introduisons les notations $I(x, y, \min_{\text{RGB}})$ et $I(x, y, \max_{\text{RGB}})$ définies comme suit :

$$I(x, y, \min_{\text{RGB}}) = \min(I(x, y, R), I(x, y, G), I(x, y, B)) \tag{2.1}$$

$$I(x, y, \max_{\text{RGB}}) = \max(I(x, y, R), I(x, y, G), I(x, y, B)) \tag{2.2}$$

Les coordonnées dans l'espace TSL s'expriment en fonction des coordonnées dans l'espace RGB par :

$$I(x, y, L) = I(x, y, \max_{\text{RGB}}) \tag{2.3}$$

Si $I(x, y, L)$ est nul, la saturation et la teinte sont indéfinies. Dans le cas contraire, on a :

$$I(x, y, L) \neq 0 \quad \text{et} \quad I(x, y, S) = \frac{I(x, y, L) - I(x, y, \min_{\text{RGB}})}{I(x, y, L)} \tag{2.4}$$

Si $I(x, y, S)$ est nul, la teinte est indéfinie. Sinon, la teinte est calculée en radians comme :

$$I(x, y, T) = \begin{cases} \frac{\pi}{3}(5 + I(x, y, B')) & \text{si} \quad I(x, y, R) \geq I(x, y, B) \geq I(x, y, G) \\ \frac{\pi}{3}(1 - I(x, y, G')) & \text{si} \quad I(x, y, R) \geq I(x, y, G) \geq I(x, y, B) \\ \frac{\pi}{3}(1 + I(x, y, R')) & \text{si} \quad I(x, y, G) \geq I(x, y, R) \geq I(x, y, B) \\ \frac{\pi}{3}(3 - I(x, y, B')) & \text{si} \quad I(x, y, G) \geq I(x, y, B) \geq I(x, y, R) \\ \frac{\pi}{3}(3 + I(x, y, G')) & \text{si} \quad I(x, y, B) \geq I(x, y, G) \geq I(x, y, R) \\ \frac{\pi}{3}(5 - I(x, y, R')) & \text{si} \quad I(x, y, B) \geq I(x, y, R) \geq I(x, y, G) \end{cases} \tag{2.5}$$

avec $I(x, y, R')$, $I(x, y, G')$, $I(x, y, B')$ calculés par les équations suivantes :

$$I(x, y, R') = \frac{I(x, y, \max_{\text{RGB}}) - I(x, y, R)}{I(x, y, \max_{\text{RGB}}) - I(x, y, \min_{\text{RGB}})} \tag{2.6}$$

$$I(x, y, G') = \frac{I(x, y, \max_{\text{RGB}}) - I(x, y, G)}{I(x, y, \max_{\text{RGB}}) - I(x, y, \min_{\text{RGB}})} \tag{2.7}$$

$$I(x, y, B') = \frac{I(x, y, \max_{\text{RGB}}) - I(x, y, B)}{I(x, y, \max_{\text{RGB}}) - I(x, y, \min_{\text{RGB}})} \tag{2.8}$$

qui peuvent également s'écrire :

$$I(x, y, R') = \frac{I(x, y, L) - I(x, y, R)}{I(x, y, S) \times I(x, y, L)} \tag{2.9}$$

$$I(x, y, G') = \frac{I(x, y, L) - I(x, y, G)}{I(x, y, S) \times I(x, y, L)} \tag{2.10}$$

$$I(x, y, B') = \frac{I(x, y, L) - I(x, y, B)}{I(x, y, S) \times I(x, y, L)} \tag{2.11}$$

Nous avons préféré cet espace à l'espace TSI défini par Gonzalez et Woods [Gon92b]. En effet, la conversion depuis RGB vers TSI s'accompagne d'un coût de calcul plus élevé, pour des résultats équivalents.

D'après [Car95], il est possible d'interpréter les différentes composantes comme suit :

**Teinte :** représente la couleur perçue (rouge, jaune, vert, *etc.*),

**Saturation :** mesure la pureté de la couleur (par exemple pour une teinte rouge, le rose se caractérise par une saturation plus faible que le rouge, tandis que le noir, le blanc, et le gris sont caractérisés par une saturation nulle),

**Luminance :** représente le niveau de gris, de "sombre" pour une valeur faible à "clair" pour une valeur élevée.

La figure 2.1 illustre ces différents concepts. La saturation et la luminance sont toutes deux représentées sur une échelle linéaire tandis que la teinte est modélisée par un angle. Elle fait donc l'objet d'une considération particulière lors des calculs de différentes mesures statistiques telles que la moyenne ou l'amplitude de variation.

Nous avons choisi d'utiliser la définition donnée dans [Mar00] et reprise dans [Cie01] pour

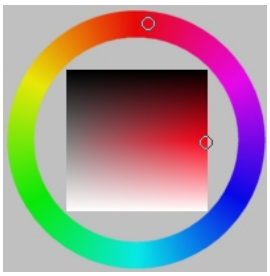

FIG. 2.1 – Représentation visuelle de l'espace TSL : le disque modélise la teinte, tandis que le carré modélise la saturation (direction horizontale) et la luminance (direction verticale).

le calcul de la moyenne $\bar{\theta}$ d'un ensemble d'angles $\{\theta_i\}_{i\in[1,\Theta]}$ qui s'obtient par la formule :

$$\bar{\theta} = \begin{cases} \mathrm{Arctan}\left(\dfrac{\sum_{i=1}^{\Theta}\sin(\theta_i)}{\sum_{i=1}^{\Theta}\cos(\theta_i)}\right) & \text{si } \sum_{i=1}^{\Theta}\cos(\theta_i) \geq 0, \\ \mathrm{Arctan}\left(\dfrac{\sum_{i=1}^{\Theta}\sin(\theta_i)}{\sum_{i=1}^{\Theta}\cos(\theta_i)}\right) + \pi & \text{sinon} \end{cases} \tag{2.12}$$

Dans [Mar00, Cie01] est donné également un algorithme permettant le calcul de l'amplitude de variation de l'ensemble $\{\theta_i\}_{i\in[1,\Theta]}$. Cette méthode nécessite un tri croissant préalable de tous les angles considérés, et est donc caractérisée par une complexité algorithmique relativement élevée. Nous proposons ici une méthode applicable dans le cas où l'angle moyen $\bar{\theta}$ a déjà été calculé. Cette méthode nécessite, en plus de la moyenne, la connaissance des angles minimum et maximum dans l'intervalle de longueur $2\pi$ choisi, respectivement notés $\theta_{\text{min}}$ et $\theta_{\text{max}}$. Si la moyenne est comprise dans l'intervalle limité par les deux angles, c'est-à-dire si $\theta_{\text{min}} \leq \bar{\theta} \leq \theta_{\text{max}}$, alors l'amplitude de variation est égale à $\theta_{\text{max}} - \theta_{\text{min}}$. Dans le cas contraire, l'angle complémentaire doit être considéré et l'amplitude de variation est égale à $2\pi - (\theta_{\text{max}} - \theta_{\text{min}})$. Si on a choisi des mesures d'angles comprises entre 0 et $2\pi$, l'amplitude de variation $\mathcal{A}(\{\theta_i\}_{i\in[1,\Theta]})$ est définie par :

$$\mathcal{A}(\{\theta_i\}_{i\in[1,\Theta]}) = \begin{cases} \max\limits_{i\in[1,\Theta]}(\theta_i) - \min\limits_{i\in[1,\Theta]}(\theta_i) & \text{si } \min\limits_{i\in[1,\Theta]}(\theta_i) \leq \bar{\theta} \leq \max\limits_{i\in[1,\Theta]}(\theta_i), \\ 2\pi - (\max\limits_{i\in[1,\Theta]}(\theta_i) - \min\limits_{i\in[1,\Theta]}(\theta_i)) & \text{sinon} \end{cases} \tag{2.13}$$

L'espace TSL a l'avantage de permettre l'élaboration de méthodes **robustes aux changements d'illumination**. En effet, puisqu'il sépare les composantes de luminance (L) et de chrominance (T et S), il est possible de ne considérer que la teinte (T) et la saturation (S) afin de diminuer la sensibilité aux changements d'illumination. Certains algorithmes présentés dans ce mémoire sont basés sur l'espace TS, sous-espace de l'espace TSL. La robustesse aux changements d'illumination est ainsi accrue en donnant plus de considération aux composantes T et S qu'à la composante L. Cependant, comme le précise Carron [Car95], **la teinte n'est significative que si la saturation est élevée. Les méthodes d'analyse basées sur la teinte des pixels doivent donc vérifier que ceux-ci ne sont pas achromatiques**.

De plus, on peut considérer que les représentations d'une image initiale (en considérant l'espace RGB) à différentes résolutions sont obtenues par moyennages successifs des valeurs de plusieurs pixels, c'est-à-dire qu'à la résolution $r$ les valeurs sont calculées comme la moyenne

de $k$ valeurs présentes à la résolution $r-1$, à partir d'une image originale avec $r=0$. Dans ce cas, la plage de valeurs pour les 3 composantes R, G, et B sera de plus en plus restreinte et l'écart-type de plus en plus faible. En calculant respectivement les composantes L et S par les équations (2.3) et (2.4), on observe pour les plages de valeurs et écart-types associés à ces composantes des phénomènes similaires à ceux observés pour l'espace RGB. La teinte, quant à elle, se caractérise par une plage de valeur et un écart-type indépendants de la résolution. Cette propriété sera prouvée dans la section 3.3.1 où l'on abordera plus en détail la multirésolution. La teinte étant moins sensible aux artefacts dus à des moyennages successifs des valeurs des pixels, elle sera donc à favoriser dans le cas des **méthodes multirésolution**.

Comparé aux autres espaces de représentation de la couleur, l'espace TSL présente différents avantages : modélisation de la vision humaine, robustesse aux changements d'illumination, robustesse aux moyennages successifs dans le cas de méthodes multirésolution. La teinte (T) est la composante la plus intéressante mais elle doit être analysée avec précaution. D'une part sa fiabilité dépend du niveau de saturation, et d'autre part sa nature mathématique (il s'agit d'un angle) nécessite l'utilisation de mesures statistiques spécifiques.

Nous proposons ici l'utilisation d'une partie de la représentation TSL (plus précisément les composantes T et S) qui permet l'élaboration de méthodes robustes aux changements d'illumination. Afin de traiter des séquences compressées ou non-compressées en temps réel, nous proposons de plus de diminuer tout d'abord la résolution spatiale des images.

## 2.3 Diminution de la résolution spatiale

Nous sommes concerné par l'analyse en temps réel de séquences vidéo. La détection des changements de plans, première étape de l'analyse, doit donc s'effectuer elle aussi en temps réel. Pour cela, nous proposons de diminuer la résolution spatiale des images à analyser. Ainsi, le nombre de pixels à traiter sera moins élevé, diminuant alors le nombre de calculs nécessaires.

La détection des changements de plans peut être effectuée sur des séquences vidéo compressées ou non. Seule la méthode utilisée pour diminuer la résolution spatiale dépend du type de séquences analysées. Le cas de données non-compressées et celui des données compressées seront donc présentés successivement. Dans les deux cas nous nous ramenons à des images de superficie 64 fois inférieure à celles des images originales.

### 2.3.1 Données non-compressées

Dans le cas de séquences vidéo non-compressées, chaque pixel est caractérisé par une valeur pour chaque composante couleur de la représentation utilisée. Le but de l'étape décrite ici est de représenter chaque image par un nombre de pixels inférieur à celui d'origine.

Nous proposons donc de créer à partir de chaque image originale (contenant $X \times Y$ pixels) une nouvelle image composée de $X' \times Y'$ pixels. On pose $X' = \frac{X}{8}$ et $Y' = \frac{Y}{8}$.

Pour chaque bloc $8 \times 8$ (64 pixels) et pour chaque composante couleur, la moyenne du bloc est calculée. Les valeurs obtenues permettent de caractériser les nouveaux pixels de l'image à faible résolution. Le procédé de calcul peut donc être formulé comme suit :

$$I'_t(x', y', c) = \frac{1}{64} \sum_{x=8(x'-1)+1}^{8x'} \sum_{y=8(y'-1)+1}^{8y'} I_t(x, y, c) \tag{2.14}$$

où $I'_t$ représente l'image à faible résolution (de taille $X' \times Y'$) obtenue à partir de la trame $I_t$ (de taille $X \times Y$).

### 2.3.2 Données compressées

Dans le cas de séquences vidéo compressées, il est possible de construire des images à faible résolution en évitant une décompression complète des données. En effet, dans les séquences vidéo compressées selon les normes M-JPEG, MPEG-1 ou MPEG-2, les coefficients obtenus par l'application de la TCD sur chaque bloc de l'image peuvent être utilisés. Nous ne considérons ici que les trames codées sans information de mouvement, c'est-à-dire toutes les trames des séquences vidéo M-JPEG ainsi que les trames I des séquences vidéo MPEG. Ces trames compressées ne contiennent plus des pixels mais des coefficients DC et AC. Elles seront notées $\dot{I}_t$ et non plus $I_t$.

Chaque bloc $8 \times 8$ (64 pixels) est caractérisé par un coefficient DC, représentant l'information de basse-fréquence, et 63 coefficients AC. Les coefficients DC des différents blocs de l'image peuvent être utilisés pour générer l'image $I'$ à faible résolution, de la manière suivante :

$$I'_t(x', y', c) = \frac{1}{8} \dot{I}_t^{b(x',y')}(0, 0, c) \tag{2.15}$$

où $b(x', y')$ représente un bloc de coordonnées spatiales $(x', y')$ (définies à l'échelle des blocs) et $\dot{I}_t^b(0, 0, c)$ le premier coefficient (position $(0, 0)$) du bloc $b$, c'est-à-dire le coefficient DC, en considérant la composante couleur $c$ de l'image compressée $\dot{I}_t$ de coefficients.

Certains auteurs ont proposé des techniques pour extraire des images à faible résolution à partir des trames P et B de séquences vidéo MPEG, par exemple Yeo et Liu [Yeo95] et Song et Yeo [Son98] pour le cas de séquences vidéo MPEG-1 et MPEG-2 respectivement. Plus généralement, l'analyse ou le traitement d'images ou de séquences vidéo dans le domaine compressé fait l'objet de nombreuses études [Cha95, Smi93, Sea96, Men96, Kob97, Smi95, She98, Ach98].

## 2.4 Evolution temporelle d'une mesure de distance

Comme il est décrit dans le chapitre I, la plupart des méthodes de détection des changements de plans se basent sur la comparaison à un seuil $S$ d'une mesure de dissimilarité $d$ calculée entre deux images successives (c.f. équation (1.2)). Le seuil utilisé doit souvent être fixé de manière empirique, et dépend du domaine vidéo étudié (sport, bulletin d'informations, *etc.*) ou du type de plans présents dans la séquence. Ainsi, un plan éloigné, où les objets en mouvement sont petits, sera caractérisé par une valeur $d$ relativement faible tandis qu'un plan proche ou serré, où les objets en mouvement occupent une portion importante de l'image, sera caractérisé par une valeur $d$ plus élevée. Le seuil $S$ devra donc être ajusté en conséquence afin d'éviter les fausses détections ou l'absence de détection. Comme certaines séquences vidéo, notamment les retransmissions télévisées d'événements sportifs, contiennent des plans éloignés et des plans proches, il est nécessaire d'utiliser une méthode plus générale qui puisse s'adapter à ces différents types de plans. Nous proposons donc d'introduire **un seuil adaptatif**, noté $S_d$, qui dépend du temps. La valeur du seuil est mise à jour pour chaque nouvelle image de la séquence, suivant la formule suivante :

$$S_d(t) = \alpha_{S_d} S_d(t-1) + (1 - \alpha_{S_d}) d(I_t, I_{t-1}) \tag{2.16}$$

où $S_d(t)$ représente la valeur du seuil $S_d$ à l'instant $t$. De cette façon, il s'adapte automatiquement avec une certaine inertie (représentée par le coefficient $\alpha_{S_d}$) au contenu de la vidéo étudiée, sa valeur étant modifiée en fonction des valeurs de $d(I_t, I_{t-1})$. Le paramètre $\alpha_{S_d}$ a une influence directe sur les mesures de précision et de rappel.

L'utilisation directe d'une mesure $d$ entre deux images successives est très sensible au bruit et au mouvement présent dans la séquence étudiée. L'introduction d'un seuil adaptatif permet de limiter cette sensibilité dans une certaine mesure, mais évidemment pas dans sa totalité. Nous proposons donc de considérer une mesure relative et non une mesure absolue. Cette mesure relative, notée $d'$, permet d'accroître la robustesse au bruit et aux mouvements importants présents

dans la séquence, et est définie par :

$$d'(I_t) = |d(I_t, I_{t-1}) - d(I_{t-1}, I_{t-2})| \tag{2.17}$$

Contrairement à la mesure $d$, la mesure $d'$ est définie de manière relative et son ordre de grandeur dépend donc moins du type de plan ou de vidéo étudié. Afin de détecter un changement de plans, cette mesure peut donc être comparée à un seuil $S_{d'}$ fixé empiriquement au début de la séquence. La valeur de $S_{d'}$ pourra évoluer en fonction du type de vidéo ou de plan analysé.

Nous avons justifié ici l'intérêt d'utiliser la variation d'une mesure de distance calculée entre deux images successives plutôt que la mesure de distance elle-même. Nous présentons maintenant la mesure de distance $d$ utilisée ainsi que l'algorithme de détection des transitions brusques ou progressives.

### 2.4.1 Définition de la mesure de distance utilisée

L'espace couleur TSL a été choisi et plus précisément le sous-espace composé de T et de S. La non prise en compte de la composante L permet d'accroître la robustesse aux changements d'illumination, fréquemment rencontrés lors de séquences vidéo représentant des scènes d'extérieur.

Pour chaque image réduite de la séquence vidéo, chaque pixel n'est donc caractérisé que par deux valeurs invariantes à l'illumination : sa teinte T et sa saturation S. Cette nouvelle réduction de la taille des données participe à satisfaire la contrainte de temps réel $\mathcal{C}_{\text{rapidité}}$ imposée.

La différence entre deux images est mesurée par la distance $d$ définie par :

$$d_k(I_{t_1}, I_{t_2}) = \sum_{x=1}^{X} \sum_{y=1}^{Y} \left(I_{t_1}(x,y) \ominus I_{t_2}(x,y)\right)^k \tag{2.18}$$

avec $k \in [0, 2]$, la convention $0^0 = 0$ et $\ominus$ un opérateur algébrique défini par l'équation (2.19). La valeur de $k$ nous permet de donner plus ou moins d'importance à la différence entre chaque couple de pixels issus de deux images successives. Si on choisit $k = 0$, une faible différence entre deux pixels consécutifs dans le domaine temporel aura la même influence qu'une forte différence, contrairement au cas $k = 2$ qui augmente l'influence des plus grandes différences

entre deux pixels. L'opérateur $\ominus$ utilisé pour comparer deux pixels est défini par :

$$\begin{aligned} I_{t_1}(x,y) \ominus I_{t_2}(x,y) = \alpha_{T,S}(I_{t_1}(x,y,T) - I_{t_2}(x,y,T)) \pmod{2\pi} \\ + (1-\alpha_{T,S})\,|I_{t_1}(x,y,S) - I_{t_2}(x,y,S)| \end{aligned} \tag{2.19}$$

où $\alpha_{T,S}$ est un coefficient permettant de donner plus ou moins d'influence aux composantes T et S. En effet, dans le cas de pixels achromatiques, il est important de ne pas donner d'importance à la composante T qui n'est alors pas fiable.

La mesure de distance $d$, quoique relativement simple, permet d'estimer correctement la différence entre deux images en assurant une invariance à l'illumination. L'utilisation d'une mesure de distance plus complexe pourrait apporter une quantité d'information supplémentaire mais engendrerait également un surcoût en terme de temps de calcul.

Les mesures $d$ et $d'$ peuvent alors être utilisées afin de détecter des changements de plans, qu'ils soient brusques ou progressifs.

### 2.4.2 Détection d'un changement de plans

Comme il a été précisé dans la section 1.1, un changement de plans peut être brusque ou progressif. En considérant **une transition progressive comme une transition brusque dont les effets sont étalés sur plusieurs images**, nous proposons ici une approche permettant de détecter les transitions brusques ou progressives de manière relativement similaire.

Pour détecter un changement brusque, nous comparons directement la valeur $d'$ avec un seuil $S_{d'}$. En effet, si la valeur de $d'$ est élevée, c'est-à-dire si la différence absolue entre $d(I_t, I_{t-1})$ et $d(I_{t-1}, I_{t-2})$ est significative, alors l'évolution du contenu de la séquence entre les images $I_{t-2}$ et $I_{t-1}$ n'est pas cohérente avec celle observée entre les images $I_{t-1}$ et $I_t$. Un changement brusque existe donc à l'instant $t-1$.

Si aucun changement brusque n'a été détecté, il est encore possible de se trouver en présence d'une transition progressive. La valeur $d'$ ne peut être utilisée directement dans le cas d'une transition progressive puisqu'elle ne reflète l'évolution de la mesure de distance $d$ qu'à un instant donné. Il est donc nécessaire de cumuler les valeurs $d'$ obtenues pour toutes les trames composant une transition progressive afin d'obtenir une mesure qui sera du même ordre de grandeur que la valeur du seuil considéré dans le cas d'une transition brusque. La détection des transitions progressives s'effectue donc en deux étapes successives.

La première étape consiste en la détection des trames susceptibles d'être les frontières

d'une transition progressive. Pour détecter celles-ci, nous analysons l'évolution de la mesure de distance $d$ et nous comparons à chaque instant $t$ la valeur $d(I_t, I_{t-1})$ avec le seuil adaptatif $S_d(t)$ défini par l'équation (2.16). L'utilisation de ce seuil adaptatif nous permet de gérer tout type de situation (plan proche ou éloigné, mouvement important ou pas, *etc.*). Si la condition $d(I_{t_1}, I_{t_1-1}) > S_d(t_1)$ est vérifiée, alors il est possible qu'une transition soit présente dans la séquence à partir de l'instant $t_1$. L'instant $t_2$ de fin de cette transition correspondrait à la première trame vérifiant la condition $d(I_{t_2}, I_{t_2-1}) < S_d(t_2)$ avec $t_2 > t_1$.

Une fois les frontières $t_1$ et $t_2$ d'une possible transition déterminées, il est nécessaire d'analyser les trames $t$ de cet intervalle de temps. Pour cela, nous calculons la valeur cumulée de $d'$ sur l'ensemble des trames $[t_1, t_2]$ notée $d'_{\text{cumul}}(t_1, t_2)$ :

$$d'_{\text{cumul}}(t_1, t_2) = \sum_{t=t_1}^{t_2} d'(I_t) \tag{2.20}$$

La comparaison de $d'_{\text{cumul}}(t_1, t_2)$ avec le seuil $S_{d'}$ permet alors de valider ou non la présence d'un changement progressif entre les trames $I_{t_1}$ et $I_{t_2}$.

Nous avons résumé tout le processus de détection d'un changement de plans dans l'algorithme 1 page 36.

La méthode décrite ici a été testée sur différentes séquences vidéo afin de vérifier son efficacité. Les résultats obtenus sont détaillés dans la section suivante.

## 2.5 Résultats

La méthode proposée ici a été testée sur des séquences vidéo de différents domaines. Afin de souligner ses capacités, nous présentons ici des résultats obtenus à partir de retransmissions de matchs de football. En effet, les séquences analysées sont caractérisées par des plans proches ou éloignés, un grand nombre d'effets différents utilisés, une variation importante des mouvements de la caméra et des objets présents dans la scène, des changements d'illumination fréquents liés au système d'éclairage, et enfin une certaine homogénéité des images de la séquence vidéo (même si elles appartiennent à différents plans) puisque l'on observe dans la plupart des images une couleur prédominante (le vert). La détection est plus difficile dans ce contexte. De plus, un ensemble de tests nous a permis de vérifier expérimentalement que les résultats obtenus avec cette méthode étaient meilleurs que ceux obtenus avec d'autres méthodes de la littérature.

Les images présentées ici sont issues de séquences télévisées acquises à une fréquence de

**Algorithme 1:** Détection d'un changement de plans dans l'espace TSL.

**Entrée** : Séquence vidéo compressée ou non, et contenant $T$ trames codées en RGB
Paramètres $\alpha_{T,S}$, $\alpha_{S_d}$, $S_{d'}$, et $k$

**Sortie** : Positions des transitions brusques et progressives

*Parcours de la séquence vidéo*
**pour** $t \leftarrow 1$ **à** $T$ **faire**
  *Diminution de la résolution spatiale*
  **si** Séquence vidéo compressée **alors**
    Calcul de $I'_t(x', y', c) \quad \forall x' \in [1, X'], \forall y' \in [1, Y'], \forall c \in [1, 3]$ par l'équation (2.15)
  **sinon**
    Calcul de $I'_t(x', y', c) \quad \forall x' \in [1, X'], \forall y' \in [1, Y'], \forall c \in [1, 3]$ par l'équation (2.14)
  *Changement d'espace couleur*
  Transformation depuis l'espace RGB vers le sous-espace TS
  *Calcul des mesures de distance*
  Calcul de $S_d(t)$ par l'équation (2.16)
  Calcul de $d(I_t, I_{t-1})$ par les équations (2.18) et (2.19)
  Calcul de $d'(I_t)$ par l'équation (2.17)
  *Recherche d'une transition brusque*
  **si** $d'(I_t) > S_{d'}$ **alors**
    Détection d'une transition brusque
    $d'_{\text{cumul}}(t) \leftarrow 0$
  *Recherche d'une transition progressive*
  **sinon si** $d(I_t, I_{t-1}) > S_d(t)$ **alors**
    $d'_{\text{cumul}}(t) \leftarrow d'_{\text{cumul}}(t-1) + d'(I_t)$
  **sinon si** $d'_{\text{cumul}}(t) > S_{d'}$ **alors**
    Détection d'une transition progressive
    $d'_{\text{cumul}}(t) \leftarrow 0$
  **sinon**
    *Fausse alerte*
    $d'_{\text{cumul}}(t) \leftarrow 0$

25 Hz et leur taille est de $160 \times 120$ pixels. Après l'étape de réduction, les images à traiter contiennent $20 \times 15$ pixels, comme le montre la figure 2.2. Les séquences vidéo contiennent différents effets (cut, volet, fondu, mais aussi certains effets combinant volet et fondu). Elles incluent aussi du bruit dû au mouvement global de la caméra, aux objets en mouvement, ainsi qu'aux effets d'illumination. Pour cette taille d'images ($160 \times 120$ pixels), le temps de calcul nécessaire est égal à 4 millisecondes par image. Cette mesure a été obtenue sur une architecture PC Pentium 4 candencé à 1700 MHz.

La figure 2.3 montre l'évolution des mesures $d$, $d'$, et $d'_{\text{cumul}}$ pour des séquences vidéo contenant différents types de transition. Les paramètres utilisés sont décrits dans le tableau 2.1.

Comme il est expliqué en 1.2, la qualité d'une méthode de détection de changements de plans peut être évaluée grâce aux mesures de rappel et de précision. Des tests effectués sur une vingtaine de séquences composées chacune de 500 images et contenant chacune entre un et trois changements de plans (brusques ou progressifs) ont permis d'obtenir les mesures de rappel et de précision présentées dans le tableau 2.2. La valeur faible de rappel obtenue dans le cas des transitions progressives peut être expliquée par la limite effective de l'approche proposée ici. Celle-ci est en effet sensible au mouvement présent dans la séquence. Ce mouvement apparent peut être provoqué par une accéleration brusque de la caméra ou par le mouvement d'un objet occupant quasiment toute l'image dans un plan rapproché.

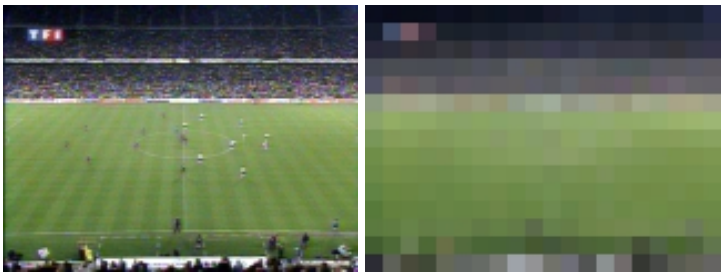

FIG. 2.2 – Réduction de la résolution spatiale d'une image par un facteur 64 : image originale (à gauche) et image réduite (à droite).

| Paramètre | Description | Valeur |
|---|---|---|
| $\alpha_{T,S}$ | Influence de la teinte et de la saturation | $0.3$ |
| $\alpha_{S_d}$ | Inertie du seuil adaptatif $S_d$ | $0.25$ |
| $S_{d'}$ | Seuil invariant utilisé pour la détection | $10$ |
| $k$ | Influence de la différence $\ominus$ entre deux pixels | $1$ |

TAB. 2.1 – Paramètres utilisés lors de la détection des changements de plans.

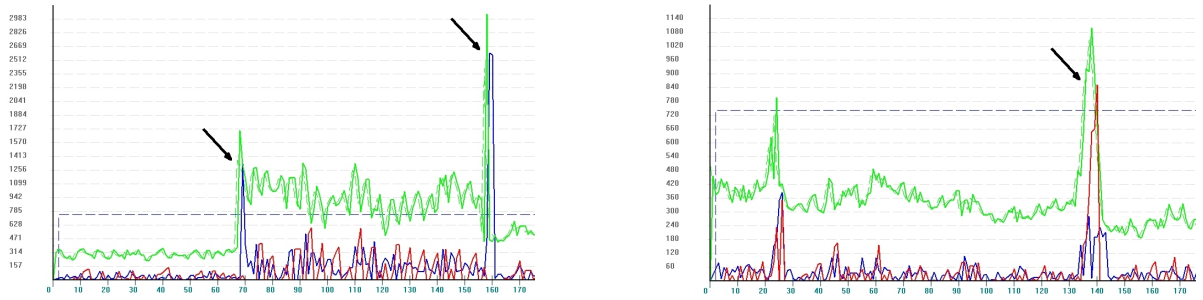

FIG. 2.3 – Evolution temporelle des mesures $d$ (en vert), $d'$ (en rouge), et $d'_{\text{cumul}}$ (en bleu) pour des séquences contenant des transitions brusques (à gauche) ou progressives (à droite) indiquées par les flèches.

| Type de changement | Rappel | Précision |
|---|---|---|
| Transitions brusques | 100 % | 100 % |
| Transitions progressives | 54 % | 100 % |
| Transitions quelconques | 82 % | 100 % |

TAB. 2.2 – Qualité de la détection de changements de plans.

Finalement, nous avons évalué la complexité théorique de l'approche proposée en utilisant les notations introduites dans le chapitre 1. La mesure obtenue est égale à $\mathcal{O}(3\mathcal{P} + 27\mathcal{B})$. Plus précisément, la réduction de l'image (dans le cas de données non compressées), le changement d'espace de représentation couleur, et le calcul de la mesure $d(I_{t_1}, I_{t_2})$ requièrent respectivement $3\mathcal{P} + 3\mathcal{B}$, $16\mathcal{B}$, et $8\mathcal{B}$ opérations. Si l'on compare avec les différentes méthodes présentées dans l'état de l'art (chapitre 1, et plus particulièrement le tableau 1.1), la complexité obtenue ici est relativement faible pour un résultat pour le moins satisfaisant.

## 2.6 Conclusion

Dans ce chapitre, nous cherchions à résoudre le problème de la détection des changements de plans en considérant les contraintes décrites en page 3.

Afin de satisfaire la contrainte $\mathcal{C}_{\text{rapidité}}$ tout en considérant la contrainte $\mathcal{C}_{\text{couleur}}$, nous avons proposé de diminuer la quantité de données à traiter, par réduction du nombre de pixels des images et du nombre de composantes couleur utilisées. Nous avons choisi effectivement de représenter les pixels par 2 paramètres de chrominance (les composantes T et S), ce qui permet de gérer efficacement la contrainte $\mathcal{C}_{\text{illumination}}$. La diminution du nombre de pixels des images est quant à elle indépendante de la nature des séquences vidéo, ce qui nous permet de gérer la

contrainte $\mathcal{C}_{\text{compression}}$. L'aspect $\mathcal{C}_{\text{mouvement}}$ est important à prendre en compte car il peut générer un grand nombre d'erreurs lors de la détection. Pour considérer cette contrainte tout en limitant le taux d'erreur, nous avons introduit d'une part un seuil adaptatif, et d'autre part une façon originale d'utiliser les mesures de dissimilarité entre images. Plus précisément, nous nous basons sur l'étude des variations des mesures de dissimilarité et non sur les mesures elles-mêmes.

Une fois la séquence vidéo découpée en plans, il est alors possible d'étudier le contenu des images constituant le plan en question. Nous considérons l'arrière-plan de la scène et les objets en mouvement présents dans la scène. Nous nous proposons dans la partie suivante **d'étudier l'arrière-plan de la scène**.

# Deuxième partie

# Etude de l’arrière-plan de la scène

Après avoir segmenté une séquence vidéo en plans, chaque plan peut ensuite être analysé indépendamment des autres. Le contenu des différentes images peut être classé en deux parties décrivant la scène observée : l'arrière-plan et les objets en mouvement.

Dans cette partie, nous étudierons **l'arrière-plan de la scène**. Pour cela, nous proposerons tout d'abord (chapitre 3) de résoudre le problème de la **séparation des objets et du fond** (c'est-à-dire l'arrière-plan). Le résultat obtenu permet d'une part l'obtention des positions des objets dans l'image. D'autre part, l'analyse des images pour leur interprétation nécessite de mettre en évidence la structuration du fond qui permettra de se référencer dans l'espace. La **structuration du fond** sera abordée dans un second temps (chapitre 4). Elle permettra notamment un changement de repère des positions des objets depuis l'espace 2-D de l'image vers l'espace de la scène réelle.

# Chapitre 3

# Séparation des objets et du fond

Après la détection de changements de plans, il est possible d'étudier le contenu des images d'un plan. Cette étude peut concerner soit l'arrière-plan, soit les objets en mouvement présents dans la scène. L'étude de l'arrière-plan permet de résoudre différents problèmes. Parmi eux, celui de la **séparation des objets et du fond** sera abordé dans ce chapitre. Ce problème est posé dans de nombreuses applications, telles que le suivi d'objet, l'interprétation du contenu des images et des séquences d'images, ou encore la compression. En effet, la norme de compression MPEG-4 décrit une scène par les différents objets qui la composent et par son arrière-plan [Has98, Sik97, Zha95b].

Après avoir justifié l'importance de la séparation des objets et du fond, nous présenterons les approches traditionnellement utilisées pour résoudre ce problème. Deux types d'approches existent, selon l'état de la caméra effectuant l'acquisition (statique ou dynamique). Nous proposerons une approche multirésolution permettant de résoudre ce problème [Lef02i].

## 3.1 Problématique

La séparation du fond et des objets permet une meilleure interprétation du contenu des images. En effet, une fois la segmentation en deux parties (objets et fond) effectuée, il est alors possible d'analyser plus finement chacune de ces parties. D'une part, les pixels correspondant à l'arrière-plan peuvent être étudiés afin d'obtenir un modèle de l'espace, c'est-à-dire des informations permettant de générer un modèle de la structure de la scène ou de l'environnement. D'autre part, les parties de l'image correspondant aux objets peuvent être traitées avec des algorithmes adéquats, comme les algorithmes de suivi d'objet (c.f. partie III). Ces algorithmes, même s'ils sont performants, nécessitent le plus souvent une étape d'initialisation des positions

des objets dans la première image de la séquence (ou plus précisément du plan). Dans le cas d'un système automatique d'indexation de séquences vidéo, une saisie manuelle des positions initiales des objets par l'utilisateur est bien évidemment à proscrire. L'obtention automatique de ces positions est donc nécessaire. Elle peut être réalisée grâce à l'utilisation du résultat de la séparation du fond et des objets. C'est d'ailleurs l'utilisation principale de ce type d'algorithme : **la séparation du fond et des objets permet l'initialisation des positions des objets dans la première image d'un plan**, tandis que les positions dans les images suivantes sont obtenues par utilisation d'un algorithme de suivi d'objet. Nous nous limitons donc au cas où tous les objets sont présents dans la scène avant d'être suivis, et nous ne considérons pas les objets pouvant apparaître au cours du plan.

Le domaine (noté $\mathcal{I}$) de définition des images d'une séquence vidéo peut être décomposé la plupart du temps en deux parties complémentaires : la scène ou l'environnement qui est représenté par l'arrière-plan, et les objets mobiles présents dans la scène. En notant $\mathcal{R}$ le domaine occupé par les objets, nous pouvons définir $\overline{\mathcal{R}} = \complement_{\mathcal{I}}\mathcal{R}$ le domaine occupé par l'arrière-plan de la scène.

**La détection des objets et celle de l'arrière-plan sont donc deux problèmes complémentaires** et il est possible de déterminer les pixels de l'image appartenant à l'une de ces deux parties par recherche des pixels de la partie complémentaire.

Dans le cadre de la détection d'événements dans des séquences vidéo, les images analysées présentent souvent un partitionnement non équitable de l'arrière-plan et des objets. En effet, l'arrière-plan occupe fréquemment **une partie du champ de vision plus importante** que les objets en mouvement présents dans la scène, ce qui se traduit par $\mathrm{aire}(\mathcal{R}) \ll \mathrm{aire}(\overline{\mathcal{R}})$. Cela permet une meilleure interprétation des actions effectuées par les différents objets dans leur environnement. Nous aborderons donc ici le problème de la séparation du fond et des objets dans ce type d'images. Les objets considérés ne doivent pas être prépondérants dans l'image. Nous ne traiterons pas par exemple le cas des images de visage où le visage occupe quasiment toute l'image.

L'intégration d'une méthode de séparation du fond et des objets dans un système d'indexation de séquences vidéo impose certaines contraintes. Ainsi, les images analysées seront des images couleur acquises avec une caméra qui peut être en mouvement et peut représenter des scènes soit d'intérieur soit d'extérieur. Dans ce dernier cas, les changements d'illumination sont fréquents, et la méthode devra donc être caractérisée par une certaine robustesse à ce type d'artefact. Finalement, l'exécution devra être effectuée en temps réel sur une architecture informatique standard, ce qui signifie que la complexité algorithmique de la méthode devra être relativement faible. Nous considérons donc ici les contraintes $\mathcal{C}_{\text{rapidité}}$, $\mathcal{C}_{\text{couleur}}$, $\mathcal{C}_{\text{illumination}}$, $\mathcal{C}_{\text{mouvement}}$.

Le problème de la séparation du fond et des objets fait l'objet de différentes méthodes. Ces approches sont souvent adaptées à des séquences acquises avec une caméra fixe (statique) ou en mouvement (dynamique). Nous présentons les principales techniques dans la prochaine section.

## 3.2 Approches classiques

Nous présentons les techniques les plus couramment utilisées en considérant d'abord le cas d'une caméra statique, puis celui d'une caméra dynamique.

### 3.2.1 Caméra statique

Dans le cas où la caméra effectuant l'acquisition est fixe, la séparation du fond et des objets peut être obtenue par différentes techniques de faible complexité algorithmique. Ces techniques sont basées sur la comparaison des images successives de la séquence. Plus précisément, la cohérence de l'intensité ou de la couleur d'un pixel au cours du temps influe directement sur la classification qu'on lui donne : fond ou objet.

Ainsi, une première approche suppose que l'arrière-plan est connu initialement *via* la disponibilité d'une image de référence $I_{\text{ref}}$. Cette image représente uniquement la scène, aucun objet n'étant présent. Pour une image donnée $I_t$, la classification d'un pixel $P_t(x, y)$ en objet ou en fond résulte de la comparaison avec l'image de référence :

$$P_t(x,y) \text{ classé en } \begin{cases} \text{objet} & \text{si } |I_t(x,y) - I_{\text{ref}}(x,y)| > S, \\ \text{fond} & \text{sinon} \end{cases} \tag{3.1}$$

$S$ étant un seuil de tolérance à fixer empiriquement ou après apprentissage. Différents auteurs [Ros98, Sah88] ont proposé des méthodes pour choisir la valeur du seuil $S$. La figure 3.1 page 45 montre le résultat obtenu avec cette méthode.

Cependant, il est possible dans certains cas qu'aucune image de référence ne soit disponible. La séparation des objets et du fond s'effectue alors par comparaison d'images successives, en considérant un intervalle de temps $\Delta_t$ plus ou moins important. La classification d'un pixel est alors donnée par :

$$P_t(x,y) \text{ classé en } \begin{cases} \text{objet} & \text{si } |I_t(x,y) - I_{t-\Delta_t}(x,y)| > S, \\ \text{fond} & \text{sinon} \end{cases} \tag{3.2}$$

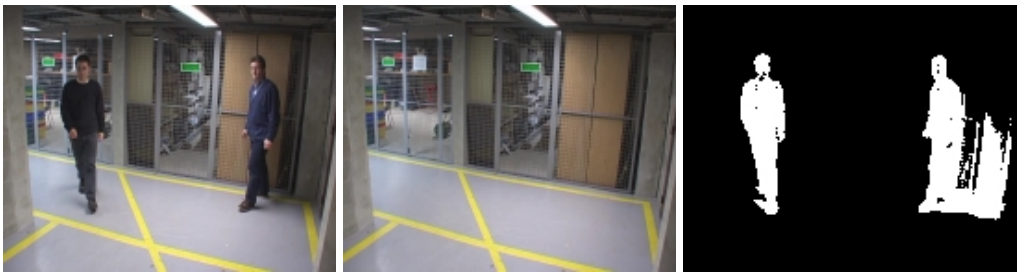

FIG. 3.1 – Séparation du fond et des objets par comparaison avec une image de référence : image analysée (à gauche), image de référence (au centre), et résultat (à droite).

FIG. 3.2 – Séparation du fond et des objets par comparaison d'images successives : image analysée (à gauche), image précédente (au centre), et résultat (à droite).

De même que pour la méthode précédente, nous illustrons cette technique par la figure 3.2.

Les deux techniques décrites précédemment considèrent un modèle figé de l'arrière-plan. Ce modèle est insuffisant dans certains cas tels que les changements d'illumination au cours de la journée ou encore l'arrêt durable d'un objet initialement en mouvement. Afin de gérer ce type de situation, des modèles plus robustes ont été proposés. Ces modèles sont dits adaptatifs et évoluent au cours du temps pour intégrer les éléments nouveaux. Ainsi Stauffer et Grimson proposent dans [Sta99] de modéliser chaque niveau d'un pixel comme un mélange de gaussiennes afin de le classer en objet ou en fond, tandis que Elgammal *et al.* [Elg00] préfèrent utiliser un modèle non-paramétrique. Young *et al.* comparent dans [You99] différents modèles statistiques pour résoudre ce problème.

Une autre approche consiste en l'ajout d'un post-traitement à l'une des deux techniques décrites par les équations (3.1) et (3.2). Ainsi, Martins *et al.* [Mar99] intègrent des filtres et opérations de morphologie mathématique afin d'améliorer le résultat. La méthode obtenue s'exécute en temps réel sur une architecture standard.

Finalement, la séparation des objets et du fond peut aussi être obtenue en considérant la scène en 3-D et non plus en 2-D. Il est dans ce cas nécessaire de considérer un capteur sup-

plémentaire, comme un matériel de profondeur [Gor99] ou un système de plusieurs caméras [Iva00]. Nous n'aborderons pas ce type d'approches dans notre étude.

Les principaux travaux menés dans le cadre de la séparation du fond et des objets se sont focalisés sur les séquences vidéo acquises à l'aide d'une caméra statique. Si la caméra effectuant l'acquisition des images est en mouvement, d'autres techniques doivent être employées.

### 3.2.2 Caméra dynamique

Dans le cas d'une caméra statique, certaines des méthodes décrites permettent de satisfaire trois des quatre contraintes énoncées dans la section 3.1 : $\mathcal{C}_{\text{rapidité}}$, $\mathcal{C}_{\text{couleur}}$, et $\mathcal{C}_{\text{illumination}}$. La dernière contrainte $\mathcal{C}_{\text{mouvement}}$, quant à elle, n'a fait l'objet que d'un nombre limité d'études. La séparation des objets et du fond dans le cas d'une caméra dynamique est le plus souvent obtenue par l'utilisation de la méthode que nous décrivons ici.

La méthode est composée de trois étapes successives : estimation du mouvement de la caméra, compensation de ce mouvement, et utilisation d'une approche pour caméra statique.

Tout d'abord, il est nécessaire d'estimer le mouvement de la caméra, ce qui revient à estimer le mouvement global observé dans la séquence. De nombreuses méthodes permettent de résoudre ce problème. On pourra par exemple se référer aux travaux d'Odobez et Bouthemy [Odo95] dans le cas de séquences vidéo non-compressées ou à ceux de Tan *et al.* [Tan00] dans le cas de séquences vidéo compressées. On trouvera dans [Kon00] et dans [Wan02] une présentation détaillée des modèles et estimateurs de mouvement. On pourra également se référer à l'ouvrage de Tziritas et Labit [Tzi94].

Une fois le mouvement de la caméra estimé, une technique de compensation du mouvement doit être utilisée [Ell99, Son00, Zhe02]. La compensation du mouvement permet d'annuler les effets dans l'image du mouvement de la caméra. Les images obtenues peuvent donc être considérées comme acquises avec une caméra fixe [Beu02].

La dernière étape consiste alors en l'utilisation d'une méthode pour caméra fixe, telle que celles présentées dans la section précédente. Là encore, la qualité du résultat peut être améliorée en utilisant un modèle plus robuste ou en ajoutant des post-traitements.

Parmi ces trois étapes, la dernière peut être exécutée en temps réel. Les deux premières, quant à elles, nécessitent un nombre important de calculs qui empêche une exécution en temps réel sur une architecture informatique standard. En effet, l'estimation et la compensation de mouvement sont deux traitements connus pour leur complexité algorithmique. Même si les différentes méthodes présentées dans la littérature pour gérer le cas d'une caméra dynamique

satisfont aux contraintes $\mathcal{C}_{\text{couleur}}$, $\mathcal{C}_{\text{illumination}}$, et $\mathcal{C}_{\text{mouvement}}$, elles ne peuvent satisfaire à la dernière contrainte $\mathcal{C}_{\text{rapidité}}$. Aucune méthode ne permet donc de satisfaire toutes les contraintes énoncées dans la section 3.1.

Partant de cette constatation, nous proposons dans les sections suivantes une approche permettant de résoudre ce problème. Puisque l'information de mouvement ne peut être traitée en temps réel, nous avons choisi de ne pas la prendre en compte. La méthode que nous présentons dans ce chapitre est donc basée sur l'analyse d'une seule image. Elle pourrait également être utilisée dans le cadre de la séparation du fond et des objets dans des images fixes, où le modèle de l'arrière-plan ne peut être amélioré au cours du temps et où aucune information de mouvement n'est disponible. Nous nous limitons ici au cas d'images non-compressées, contrairement aux travaux de Lamarre et Clark [Lam02] qui proposent d'analyser directement des images JPEG.

## 3.3 Une approche multirésolution

Nous proposons ici une solution [Lef02i] au problème posé, qui respecte les contraintes énoncées. La méthode est, de plus, robuste aux changements d'illumination et permet ainsi une analyse correcte des images représentant des scènes d'extérieur.

Afin de justifier l'utilisation d'une approche multirésolution, considérons que l'image $I$ représentant la scène est étudiée par un observateur donné. On peut remarquer que la distance $d$ entre l'observateur et l'image influe directement sur sa capacité à visualiser les détails présents dans l'image. La figure 3.3 page 48 illustre ce principe. Si l'observateur est suffisamment éloigné de l'image, il ne distinguera dans celle-ci que l'arrière-plan, puisque nous considérons toujours que l'arrière-plan occupe la partie la plus importante de l'image, comparativement aux objets présents dans la scène. Plus l'observateur est proche de l'image, plus il est capable de distinguer les objets de manière précise.

Le constat énoncé précédemment peut être vérifié de manière plus formelle. Pour cela, conservons les notations $\mathcal{I}$, $\mathcal{R}$, et $\overline{\mathcal{R}}$ représentant les domaines respectivement occupés par l'image, la scène, et les objets. Ceux-ci dépendent de la distance $d$ et seront donc notés plus précisément $\mathcal{I}_d$, $\mathcal{R}_d$, et $\overline{\mathcal{R}}_d$. Affinons de plus la définition de $\mathcal{R}_d$ par la propriété suivante. Quand une région $\mathcal{R}_d$ a une aire inférieure à $\varepsilon$, on redéfinit $\mathcal{R}_d$ comme l'ensemble $\varnothing$. $\varepsilon$ représente donc l'aire en dessous de laquelle une région ne peut plus être distinguée par l'oeil humain. D'après les définitions de $\mathcal{R}$ et $\overline{\mathcal{R}}$, nous avons :

$$\mathcal{I}_d = \mathcal{R}_d \cup \overline{\mathcal{R}}_d \qquad \forall d \in \mathbb{R}^+ \tag{3.3}$$

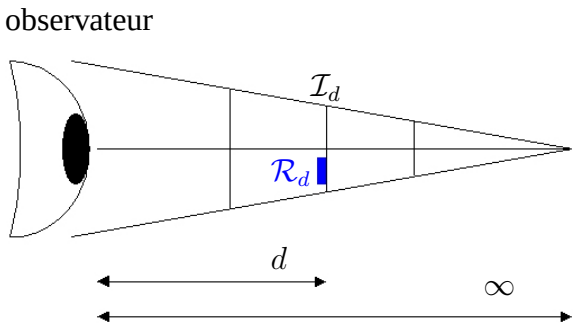


FIG. 3.3 – Vue d'une image par un observateur fixe à différentes distances $d$.

soit :

$$\text{aire}(\mathcal{I}_d) = \text{aire}(\mathcal{R}_d) + \text{aire}(\overline{\mathcal{R}}_d) \qquad \forall d \in \mathbb{R}^+ \tag{3.4}$$

d'une part, et :

$$\text{aire}(\mathcal{R}_d) \ll \text{aire}(\overline{\mathcal{R}}_d) \qquad \forall d \in \mathbb{R}^+ \tag{3.5}$$

d'autre part. En considérant les deux équations précédentes, on peut traduire l'éloignement de l'observateur à la scène par :

$$\lim_{d\to\infty} \text{aire}(\mathcal{R}_d) + \lim_{d\to\infty} \text{aire}(\overline{\mathcal{R}}_d) = \lim_{d\to\infty} \text{aire}(\mathcal{I}_d) \tag{3.6}$$

$$\lim_{d\to\infty} \frac{\text{aire}(\mathcal{R}_d)}{\text{aire}(\mathcal{I}_d)} + \lim_{d\to\infty} \frac{\text{aire}(\overline{\mathcal{R}}_d)}{\text{aire}(\mathcal{I}_d)} = 1 \tag{3.7}$$

En considérant les propriétés formulées précédemment, nous avons alors :

$$\lim_{d\to\infty} \frac{\text{aire}(\mathcal{R}_d)}{\text{aire}(\mathcal{I}_d)} = 0 \tag{3.8}$$

et

$$\lim_{d\to\infty} \frac{\text{aire}(\overline{\mathcal{R}}_d)}{\text{aire}(\mathcal{I}_d)} = 1 \tag{3.9}$$

En effet, nous pouvons affirmer que :

$$\exists d_0 \quad / \quad \forall d > d_0 \quad \mathcal{R}_d = \varnothing \quad \text{et} \quad \overline{\mathcal{R}}_d = \mathcal{I}_d \tag{3.10}$$

Cette réflexion nous amène à proposer ici une approche multirésolution. En considérant une image originale $\underset{0}{I}$, il est possible de diminuer fortement sa résolution pour obtenir une image à très faible résolution $\underset{r_{\max}}{I}$. Aucun objet n'est plus perceptible dans cette image qui n'est composée que de l'arrière-plan. Un modèle du fond peut donc être calculé à partir de cette image. En augmentant la résolution $r$ de manière itérative, il est alors possible de comparer les différentes régions de l'image $\underset{r}{I}$ avec le modèle de l'arrière-plan suivant un critère donné. Cette comparaison permet de déterminer si chaque région correspond ou non à l'arrière-plan.

Dans cette section, nous donnerons quelques rappels sur la multirésolution et notamment sur son implication dans la segmentation d'images. Nous décrirons ensuite le principe de segmentation utilisé. Le critère de comparaison d'une région avec le modèle de l'arrière-plan sera également présenté. Finalement des résultats obtenus sur des images représentant des scènes d'extérieur seront analysés.

### 3.3.1 Quelques rappels sur la multirésolution

Le principe de multirésolution permet d'analyser des données à un degré de précision plus ou moins fin selon le résultat attendu. De plus, une analyse multirésolution a généralement l'avantage de limiter le nombre de calculs nécessaires par rapport à son homologue monorésolution [Ros84]. L'inconvénient principal est lié à la difficulté du choix des résolutions initiale et finale de l'analyse.

Dans le domaine de la segmentation d'images, de nombreuses approches basées sur une analyse multirésolution ont été proposées, concernant des images en niveaux de gris mais aussi des images en couleur [Liu94]. Ces approches suivent généralement un schéma *coarse-to-fine*, dont le principe est le suivant. Tout d'abord l'image est analysée à une résolution grossière afin d'obtenir un résultat initial. Le résultat est ensuite affiné itérativement à mesure que la précision de la résolution augmente.

La représentation multirésolution d'une image peut être obtenue de différentes manières. Ainsi, une transformée en ondelettes [Mal89, Mey92] peut être utilisée, comme dans les travaux [Kop99, Ram97, Wan01]. Cependant ce type d'approche se caractérise par un temps de calcul important qui ne permet pas d'envisager une exécution en temps réel. Nous lui avons donc préféré une décomposition pyramidale, comme dans les travaux de Comer et Delp [Com95] ou l'ouvrage de Jolion et Rosenfeld [Jol93]. La décomposition pyramidale d'une image est illustrée dans la figure 3.4.

Dans une représentation pyramidale de l'image, on doit préciser le mode de calcul de la

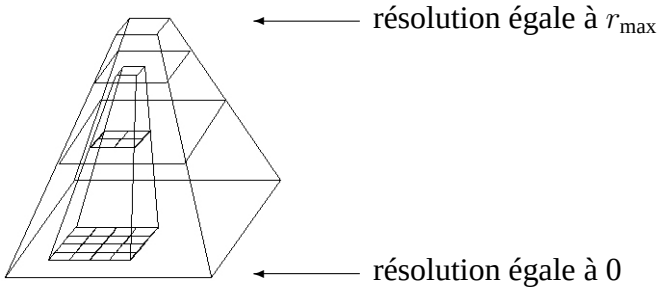


FIG. 3.4 – Représentation pyramidale d'une image.

valeur de chaque pixel d'une image à la résolution $r+1$ à partir des valeurs d'un ensemble particulier de pixels à la résolution $r$. A partir d'une image originale à laquelle on associe la résolution $r=0$, on obtient finalement une image à faible résolution caractérisée par $r=r_{\max}$.

Nous avons caractérisé précédemment la teinte T comme une composante plus robuste que la saturation S ou la luminance L dans le cas d'approches multirésolution (c.f. section 2.2.2). A l'aide des notations introduites précédemment, nous donnons ici une justification de ce que nous avons annoncé. Pour cela, nous considérons la représentation de l'image à différentes résolutions comme une suite $(\underset{r}{I})_{r\in\mathbb{N}}$ définie telle que :

$$\exists\, r_0 \in \mathbb{N} \quad / \quad \forall r > r_0 \quad \underset{r}{I} = I \tag{3.11}$$

En notant $\bar{I}(c)$ la valeur moyenne des pixels de $I$ en considérant la composante couleur $c$, on a alors les propriétés suivantes pour chaque point $M$ de l'image $I$ :

$$\lim_{r\to\infty} \underset{r}{I}(M,R) = \bar{I}(R) \tag{3.12}$$

$$\lim_{r\to\infty} \underset{r}{I}(M,G) = \bar{I}(G) \tag{3.13}$$

$$\lim_{r\to\infty} \underset{r}{I}(M,B) = \bar{I}(B) \tag{3.14}$$

En calculant en un point $M$ les composantes L et S à l'aide des équations (2.3) et (2.4), on peut affirmer que :

$$\begin{aligned}
\lim_{r\to\infty} \underset{r}{I}(M,L) &= \lim_{r\to\infty} \max\Big(\underset{r}{I}(M,R), \underset{r}{I}(M,G), \underset{r}{I}(M,B)\Big) \\
&= \max\Big(\lim_{r\to\infty} \underset{r}{I}(M,R), \lim_{r\to\infty} \underset{r}{I}(M,G), \lim_{r\to\infty} \underset{r}{I}(M,B)\Big) \\
&= \max\big(\bar{I}(R), \bar{I}(G), \bar{I}(B)\big) \\
&= \bar{I}(L)
\end{aligned} \tag{3.15}$$

et pour les pixels vérifiant $\underset{r}{I}(M,L) \neq 0$ :

$$\begin{aligned}
\lim_{r\to\infty} \underset{r}{I}(M,S) &= \lim_{r\to\infty} \frac{\underset{r}{I}(M,L) - \min\left(\underset{r}{I}(M,R), \underset{r}{I}(M,G), \underset{r}{I}(M,B)\right)}{\underset{r}{I}(M,L)} \\
&= 1 - \frac{\min\left(\lim_{r\to\infty} \underset{r}{I}(M,R), \lim_{r\to\infty} \underset{r}{I}(M,G), \lim_{r\to\infty} \underset{r}{I}(M,B)\right)}{\lim_{r\to\infty} \underset{r}{I}(M,L)} \\
&= 1 - \frac{\min\left(\bar{I}(R), \bar{I}(G), \bar{I}(B)\right)}{\bar{I}(L)} \\
&= \bar{I}(S)
\end{aligned} \tag{3.16}$$

La teinte, quant à elle, se caractérise par des valeurs indépendantes de la résolution. En effet, l'équation (2.5) ne permet pas de déterminer la valeur $\bar{I}(T)$ de la limite $\lim_{r\to\infty} \underset{r}{I}(M,T)$. La teinte est moins sensible aux artefacts dus à des moyennages successifs des valeurs des pixels, et sera donc à favoriser dans le cas des **méthodes multirésolution**.

Ayant décrit la représentation multirésolution utilisée (décomposition pyramidale), nous présentons maintenant le principe de segmentation de notre approche.

### 3.3.2 Principe de segmentation

La séparation du fond et des objets s'effectue en quatre étapes successives : création de la représentation multirésolution, estimation du modèle de l'arrière-plan, segmentation itérative des images aux différentes résolutions, et enfin segmentation finale.

La première étape consiste en la création de la représentation multirésolution de l'image (figure 3.4). Afin de calculer la valeur d'un pixel $P(x,y)$ à la résolution $r+1$ à partir d'un ensemble de $p^2$ pixels à la résolution $r$, nous utilisons la valeur moyenne des pixels de l'ensemble en question :

$$\underset{r+1}{I}(x,y,c) = \frac{1}{p^2} \sum_{x'=p(x-1)+1}^{px} \sum_{y'=p(y-1)+1}^{py} \underset{r}{I}(x',y',c) \tag{3.17}$$

La taille de l'image dépend alors de la résolution, et on a $\underset{r}{X} = \frac{X}{p^r}$ et $\underset{r}{Y} = \frac{Y}{p^r}$. La moyenne a été préférée à d'autres mesures nécessitant des calculs plus importants, comme par exemple la valeur médiane. Le calcul est effectué itérativement jusqu'à obtenir la résolution voulue $r_{\max}$.

Il est alors possible de modéliser l'arrière-plan avant de réaliser le processus de segmentation. Comme il a été précisé auparavant, nous considérons que l'arrière-plan peut être mo-

délisé par l'image $\underset{r_{\text{max}}}{I}$ , sommet de la pyramide. Cette considération n'est bien sûr valable que si l'arrière-plan occupe une partie importante de l'image, s'il diffère suffisamment des objets présents dans la scène, et s'il est assez homogène.

La segmentation peut ensuite être effectuée et améliorée de manière itérative depuis la résolution $r_{\text{max}} - 1$ jusqu'à la résolution initiale $r = 0$, base de la pyramide. A une résolution donnée $r'$ avec $r_{\text{max}} > r' > 0$, l'image $\underset{r'}{I}$ doit être analysée. Cette image est comparable à l'image initiale $\underset{0}{I}$ qui aurait été découpée en $K = (K_0)^{r_{\text{max}} - r'}$ régions, avec $K_0$ une constante dont la valeur pourrait être en toute logique égale à $p^2$. Chacune de ces régions est alors comparée avec le modèle de l'arrière-plan en utilisant une condition $\mathcal{Q}$ que nous préciserons par la suite. Si l'appariement de la région avec le modèle de l'arrière-plan est correct, on effectue ensuite un second test pour vérifier la cohérence de la région étudiée. Si le contenu de la région est homogène, on l'étiquette alors comme représentant le fond ou l'arrière-plan. Dans ce cas la région n'est plus analysée à de meilleures résolutions. A l'opposé, une région sans étiquette sera analysée plus en détail à la résolution $r' - 1$.

Cette segmentation est effectuée si nécessaire jusqu'à la résolution initiale $r = 0$. Dans le cas d'applications nécessitant une segmentation très précise, les régions correspondant aux objets peuvent être analysées par la suite afin d'affiner les contours des objets. Au contraire, si la précision des contours des objets n'est pas nécessaire (comme dans le cas de l'initialisation d'un algorithme de suivi d'objet), le processus peut être arrêté à une résolution $r_{\text{final}}$ avec $r_{\text{max}} > r_{\text{final}} \geq 0$. Dans ce cas, nous considérons que les régions sans étiquette à la résolution courante $r_{\text{final}}$ représentent les objets.

Afin d'améliorer la qualité du modèle de l'arrière-plan, il est possible de recalculer celui-ci au cours du processus de segmentation. Dans ce cas, le modèle obtenu à la résolution $r_{\text{max}}$ ne représente que l'état initial de l'arrière-plan. A mesure que la résolution devient plus fine, les résultats sont plus précis, et il est possible d'obtenir un modèle de l'arrière-plan plus fiable en ne considérant que les parties de l'image déjà étiquetées comme le fond. En effet, supposons que le processus a été réalisé depuis la résolution $r_{\text{max}}$ où le modèle de l'arrière-plan a été estimé jusqu'à la résolution $r'$, soit en utilisant un nombre $r_{\text{max}} - r'$ d'itérations. Chaque pixel à la résolution $r'$ a été étiqueté comme fond ou objet. Le modèle de l'arrière-plan peut donc être amélioré, c'est pourquoi nous le caractérisons d'évolutif. Le nouveau modèle obtenu peut alors être utilisé pour étiqueter les pixels de l'image à la résolution $r' - 1$.

Les caractéristiques de la méthode présentée (telles que le temps de calcul nécessaire ou la relative sensibilité ou robustesse aux changements d'illumination) dépendent bien évidemment du critère utilisé lors de l'appariement d'une région avec le modèle de l'arrière-plan. Le choix

de ce critère va donc maintenant être présenté et justifié.

### 3.3.3 Choix du critère de décision

Afin de comparer et d'apparier une région de l'image à la résolution $r$ avec le modèle de l'arrière-plan, il est nécessaire d'utiliser un critère de décision. Ce critère doit permettre à la méthode de segmentation une robustesse aux changements d'illumination sans pour autant ajouter un surcoût de calcul important, la méthode devant s'exécuter en temps réel. De plus, ce critère sera lié à la couleur puisque la segmentation intervient sur des images en couleur.

Contrairement à Vandenbroucke *et al.* qui proposent dans [Van98] d'utiliser un espace couleur hybride en sélectionnant les composantes couleurs les plus discriminantes dans une image donnée, nous avons fait le choix de n'utiliser qu'une seule composante couleur pour effectuer la segmentation, ce qui nous permet de limiter le temps de calcul. Puisqu'on souhaite limiter la sensibilité aux changements d'illumination, certaines composantes ou certains espaces sont proscrits. Ainsi, les composantes de luminance ou d'intensité (comme Y dans YUV, L dans TSL, I dans TSI) sont directement liées aux conditions d'éclairement et ne doivent pas être prises en compte. De même, certains espaces sont plus sensibles à la lumière, tels que RGB ou CMY. De plus, la composante sélectionnée doit être caractérisée par une faible sensibilité aux moyennages successifs, nécessaires ici pour construire la représentation multirésolution de l'image. De ce fait, l'utilisation de la composante S de TSL doit être évitée. Au contraire, la composante T permet une robustesse aux changements d'illumination et aux moyennages successifs.

Le second critère à prendre en compte concerne le temps de calcul nécessaire pour comparer une région donnée avec l'arrière-plan. Puisque la méthode doit s'exécuter en temps réel, celui-ci doit être le plus faible possible. De ce fait, nous proposons de limiter le critère couleur à une seule valeur. Ainsi la comparaison d'une région au modèle de l'arrière-plan se traduira par la comparaison entre deux valeurs scalaires. Différentes mesures peuvent être calculées pour modéliser le contenu d'une région ou de l'arrière-plan. Parmi elles, nous avons choisi de calculer la moyenne des valeurs des pixels de la région. Cette mesure a été préférée à d'autres (telles que la médiane ou la variance) afin de diminuer le temps de calcul nécessaire.

En utilisant la moyenne des teintes des pixels, il est possible d'obtenir une seule valeur, notée $\varphi$, pour une région donnée ou pour le modèle de l'arrière-plan. En utilisant l'équation

(2.12), on obtient :

$$\varphi(I) = \begin{cases} \text{Arctan}\left(\frac{\sum_{x=1}^{X}\sum_{y=1}^{Y}\sin(I(x,y,T))}{\sum_{x=1}^{X}\sum_{y=1}^{Y}\cos(I(x,y,T))}\right) & \text{si } \sum_{x=1}^{X}\sum_{y=1}^{Y}\cos(I(x,y,T)) \geq 0, \\ \text{Arctan}\left(\frac{\sum_{x=1}^{X}\sum_{y=1}^{Y}\sin(I(x,y,T))}{\sum_{x=1}^{X}\sum_{y=1}^{Y}\cos(I(x,y,T))}\right) + \pi & \text{sinon} \end{cases} \tag{3.18}$$

L'appariement, à la résolution $r$, d'une région $k$ de l'image notée $\underset{r}{I}^k$ avec le modèle de l'arrière-plan estimé à la résolution précédente dans la pyramide et noté $\varphi\left(\underset{r+1}{I}^{\text{fond}}\right)$ s'effectue en calculant la mesure $\delta$ sur les moyennes respectives des teintes :

$$\delta\left(\varphi\left(\underset{r}{I}^k\right), \varphi\left(\underset{r+1}{I}^{\text{fond}}\right)\right) = \min\left(\begin{array}{c} \left|\varphi\left(\underset{r}{I}^k\right) - \varphi\left(\underset{r+1}{I}^{\text{fond}}\right)\right| \\ 2\pi - \left|\varphi\left(\underset{r}{I}^k\right) - \varphi\left(\underset{r+1}{I}^{\text{fond}}\right)\right| \end{array}\right) , \tag{3.19}$$

Une fois la mesure $\delta$ calculée entre le modèle d'une région donnée et le modèle de l'arrière-plan, elle est comparée à un seuil $S_1$ afin d'effectuer ou non la reconnaissance. Si la condition $\delta\left(\varphi\left(\underset{r}{I}^k\right), \varphi\left(\underset{r+1}{I}^{\text{fond}}\right)\right) < S_1$ est vérifiée, alors la région $\underset{r}{I}^k$ est considérée comme l'arrière-plan. Cette condition, que nous notons $\mathcal{Q}$, est donc définie en fonction de $\varphi$, $\delta$, et $S_1$.

Il a été précédemment mentionné un test de vérification d'homogénéité effectué sur les régions semblables à l'arrière-plan. Ce test est nécessaire afin d'éviter les artefacts liés à l'utilisation de la moyenne. En effet, si une région est composée de deux pixels ayant des teintes opposées, la moyenne ne reflétera pas correctement le contenu de la région. Nous analysons donc la cohérence de chaque région appariée avec l'arrière-plan. Pour cela, plusieurs mesures de dispersion sont disponibles, comme la variance ou l'amplitude de variation. C'est cette dernière mesure qui est utilisée pour évaluer ici la cohérence d'une région : plus la plage de valeurs d'une région est faible, plus celle-ci est homogène. Pour calculer cette plage de valeurs angulaires, nous n'utilisons pas la méthode donnée dans [Mar00, Cie01] mais l'approche originale présentée dans la section 2.2.2. Une fois la plage de valeurs calculée pour une région $\underset{r}{I}^k$ par l'équation (2.13), nous comparons cette mesure de dispersion avec un second seuil $S_2$. Une plage inférieure au seuil assure l'homogénéité de la région concernée. Celle-ci est alors étiquetée en fond ou arrière-plan. Dans le cas contraire, l'hétérogénéité de la région candidate à l'étiquetage implique son rejet.

L'algorithme 2 récapitule les différentes étapes de notre méthode.

**Algorithme 2:** Approche multirésolution pour la séparation du fond et des objets.

**Entrée** : Image $\underset{0}{I}$ à la résolution originale $r = 0$
Paramètres $r_{\text{max}}$, $r_{\text{final}}$, $p^2$, $K_0$, $S_1$, et $S_2$
**Sortie** : Etiquetage des différents pixels en "fond" ou "objet"

Création de la pyramide
Estimation du modèle de l'arrière-plan $\varphi\left(\underset{r_{\text{max}}}{I}^{\text{fond}}\right) = \varphi\left(\underset{r_{\text{max}}}{I}\right)$
$r \leftarrow r_{\text{max}} - 1$
**tant que** $r \geq r_{\text{final}}$ **faire**
  Mise à jour optionnelle du modèle de l'arrière-plan
  Création de nouvelles régions à partir de celles restantes au niveau $r + 1$
  **pour** chaque région $\underset{r}{I}^k$ **faire**
    Calcul de $\varphi\left(\underset{r}{I}^k\right)$
    **si** $\delta\left(\varphi\left(\underset{r}{I}^k\right), \varphi\left(\underset{r+1}{I}^{\text{fond}}\right)\right) < S_1$ **alors**
      Estimation de l'homogénéité de $\underset{r}{I}^k$
      **si** $\underset{r}{I}^k$ est homogène **alors**
        Etiqueter $\underset{r}{I}^k$ en "arrière-plan"
  Nouvelle estimation du modèle de l'arrière-plan $\varphi\left(\underset{r}{I}^{\text{fond}}\right)$
  $r \leftarrow r - 1$
Etiqueter les régions restantes en "objet"

Nous avons décrit précédemment la méthode de séparation du fond et des objets. Elle a été testée sur différentes images afin de vérifier son efficacité et sa rapidité.

### 3.3.4 Résultats

Le choix de la teinte comme critère pour représenter les régions ou l'arrière-plan a été précédemment justifié de manière théorique. Des expériences pratiques ont été menées afin de déterminer si ce choix était le plus adéquat. Ces expériences ont pris en compte les différents espaces présentés dans la section 2.2. En comparant les résultats obtenus sur différentes images, il s'avère que la teinte T de l'espace TSL est la composante couleur donnant les meilleurs résultats de segmentation. Le choix théorique a donc été validé expérimentalement.

La méthode présentée ici a été testée sur des images d'extérieur, où l'illumination n'est pas

constante. La figure 3.5 montre les résultats obtenus sur des images extraites d'une séquence vidéo représentant un match de football, où les tailles des objets varient considérablement. Les objets sont correctement détectés, indépendamment de l'aire qu'ils occupent dans l'image. Les temps de calcul observés dépendent de la taille de l'image et du nombre de régions étudiées. Ils varient de 30 à 350 millisecondes pour des images caractérisées par un fond plus ou moins complexe et une taille comprise entre $192 \times 128$ pixels et $704 \times 576$ pixels. Ces mesures ont été obtenues en considérant une architecture PC Pentium 4 candencé à 1700 MHz.

La segmentation a été obtenue en utilisant les paramètres présentés dans le tableau 3.1. La structure de la pyramide est régulière, nous avons donc $p^2$ et $K_0$ égaux à 4.

Le processus de segmentation est itératif. La figure 3.6 illustre la diminution progressive de la résolution pour obtenir le modèle initial de l'arrière-plan, tandis que la figure 3.7 présente le résultat de la segmentation aux différentes résolutions. On peut observer que le choix de la résolution finale $r_{\text{final}}$ influe directement sur la précision du résultat. Cependant, même en considérant une résolution finale similaire à la résolution originale (*i.e.* $r_{\text{final}} = 0$), les contours des objets détectés resteront grossiers et parallèles aux côtés de l'image. Puisque nous considérons la séparation des objets et du fond comme une étape intermédiaire (dont le but est de permettre l'initialisation d'algorithmes de suivi d'objets) et non comme un but en soi, la précision obtenue est néanmoins suffisante et permet de localiser les objets sans en connaître la forme exacte. Elle répond correctement au problème posé de l'initialisation d'un suivi d'objet.

La méthode présentée considère qu'à une faible résolution, les données observées, représentant le fond, sont homogènes. Elle ne permet donc pas de traiter des images caractérisées par un fond non-uniforme (figure 3.8). De plus, l'utilisation de la teinte pour modéliser l'arrière-plan suppose que l'image analysée n'est pas achromatique.

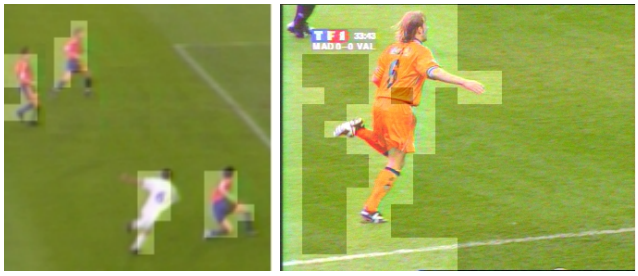

FIG. 3.5 – Résultats obtenus en considérant des objets de différentes tailles. Les régions étiquetées comme "objet" sont représentées plus claires que le reste de l'image.

| Paramètre | Description | Valeur |
|---|---|---|
| $r_{\text{max}}$ | Nombre de couches de la pyramide | 7 |
| $r_{\text{final}}$ | Résolution du résultat final | 5 |
| $p^2$ | Nombre de pixels utilisé à la résolution $r$ pour générer un pixel à la résolution $r+1$ | 4 |
| $K_0$ | Constante influant sur le nombre de régions | 4 |
| $S_1$ | Seuil utilisé lors de l'appariement | $\pi/9$ |
| $S_2$ | Seuil utilisé lors du test d'homogénéité | $\pi/9$ |

TAB. 3.1 – Paramètres utilisés lors de la séparation du fond et des objets par approche multirésolution.

FIG. 3.6 – Représentation d'une image à différentes résolutions, de la résolution originale $r = 0$ (à gauche) à $r = r_{\text{max}} = 5$ (à droite).

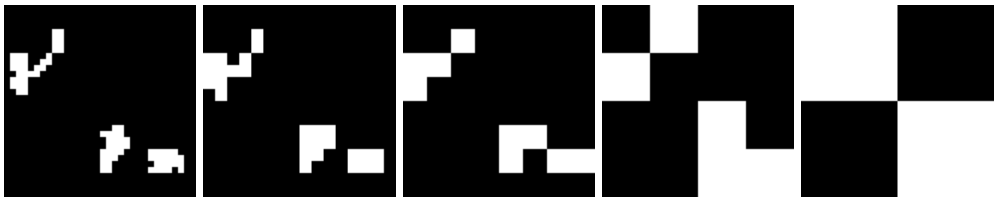

FIG. 3.7 – Résultat obtenu aux différentes résolutions, de $r_{\text{final}} = r_{\text{max}} - 1 = 4$ (à droite) à $r_{\text{final}} = 0$ (à gauche).

FIG. 3.8 – Exemple d'une image caractérisée par un fond non-uniforme.

## 3.4 Conclusion

Le problème abordé dans ce chapitre était celui de la séparation du fond (ou arrière-plan) et des objets (en mouvement) présents dans la scène. Ce problème a une importance majeure dans

de nombreuses applications telles que la compression (notamment MPEG-4), l'interprétation du contenu des images et vidéos, ou encore le suivi d'objet. La séparation des objets et du fond permet notamment d'initialiser automatiquement les positions des objets dans la première image de la séquence.

Afin de résoudre ce problème tout en prenant en compte les contraintes $\mathcal{C}_{\text{rapidité}}$, $\mathcal{C}_{\text{couleur}}$, $\mathcal{C}_{\text{illumination}}$, $\mathcal{C}_{\text{mouvement}}$, nous avons proposé une approche originale basée sur une analyse multirésolution d'une seule image. En effet, nous avons montré qu'à une faible résolution, seul l'arrière-plan était observable dans l'image. Plus la résolution augmente, plus nous pouvons distinguer les objets présents dans la scène. Afin de considérer dans le même temps les deux contraintes $\mathcal{C}_{\text{mouvement}}$ et $\mathcal{C}_{\text{rapidité}}$, nous avons proposé de ne pas utiliser l'information de mouvement. De même, nous n'avons considéré qu'une seule composante couleur lors de l'analyse de l'image pour respecter les contraintes $\mathcal{C}_{\text{couleur}}$ et $\mathcal{C}_{\text{rapidité}}$. Plus précisément, nous avons choisi la composante T de l'espace TSL. Ainsi nous pouvons gérer l'aspect $\mathcal{C}_{\text{illumination}}$.

Dans l'architecture que nous proposons pour l'indexation de séquences vidéo, un autre problème relatif à l'étude de l'arrière-plan doit être résolu. Il s'agit de la structuration du fond qui est abordée dans le prochain chapitre.

# Chapitre 4

# Structuration du fond

L'étude de l'arrière-plan peut aider à résoudre différents problèmes. Parmi eux, celui de la séparation du fond et des objets a été traité dans le chapitre précédent. Dans ce chapitre, nous aborderons un autre problème, celui de la **structuration du fond**. Une fois la structure de l'arrière-plan obtenue, il est alors possible d'effectuer un changement de repère pour référencer les positions des objets suivis, depuis l'espace de l'image vers l'espace (réel) de la scène ou de l'environnement. Les positions des différents objets dans la scène étant connues, le contenu des images peut finalement être interprété.

Dans ce chapitre, nous commencerons par expliquer l'approche adoptée pour obtenir un modèle de la structure de l'arrière-plan. Parmi les primitives pouvant être extraites d'une image, nous avons choisi d'utiliser les lignes pour représenter la structure de l'arrière-plan. Deux familles de méthodes peuvent être utilisées pour détecter les lignes présentes dans une image : les approches locales et les approches globales. Nous illustrerons chacune de ces familles par une méthode représentative. Nous proposons ensuite une approche originale permettant de résoudre le problème posé en respectant les différentes contraintes.

## 4.1 Problématique

Afin de comprendre et d'interpréter le contenu d'une séquence vidéo, l'approche utilisée consiste le plus souvent en l'analyse des différents objets composant la scène et de leurs positions relatives à un instant donné. Ces objets peuvent être séparés de l'arrière-plan (c.f. chapitre 3) puis suivis dans la séquence vidéo en utilisant des méthodes appropriées (c.f. chapitres 5 à 7). Les algorithmes de suivi d'objet permettent d'obtenir la position 2-D $(x_O, y_O)$ d'un objet $O$ dans l'image. L'information obtenue est donc limitée aux deux variables $x_O$ et $y_O$ avec

$1 \leq x_O \leq X$ et $1 \leq y_O \leq Y$. Or cette information ne permet pas d'interpréter précisément et de manière fiable le contenu de la séquence vidéo. En effet, il serait pour cela nécessaire de disposer de la position 2-D ou 3-D de l'objet $O$ relativement à un repère de la scène. Cette nouvelle information permettrait de repositionner l'objet par rapport à l'arrière-plan de la scène, et donc par rapport à son environnement. L'interprétation du contenu d'une séquence vidéo n'est donc possible que si l'on dispose d'un modèle permettant un passage des positions des objets dans l'espace de l'image vers l'espace de la scène, c'est-à-dire un espace représentant le monde réel. Nous proposons d'assimiler ce problème à celui de la modélisation de la structure de l'arrière-plan de la scène. Si la structure de l'arrière-plan de la scène (donc de l'environnement) est connu, nous disposons d'une information supplémentaire pour interpréter correctement les positions des différents objets à un instant donné.

Nous cherchons ici des outils génériques permettant l'extraction de la structure de l'arrière-plan. Puisque le modèle de la structure du fond sera utilisé pour interpréter en temps réel le contenu de la séquence vidéo, il est donc nécessaire ici aussi de prendre en compte la contrainte de temps réel $\mathcal{C}_{\text{rapidité}}$ pour résoudre ce problème. De plus, nous ne devons pas perdre de vue que les séquences vidéo à analyser sont composées d'images couleur (contrainte $\mathcal{C}_{\text{couleur}}$), pouvant présenter des changements d'illumination (contrainte $\mathcal{C}_{\text{illumination}}$). Afin d'effectuer la structuration du fond tout en respectant les différentes contraintes énoncées, plusieurs primitives ou caractéristiques extraites d'une image peuvent être utilisées.

Par exemple il est possible de se focaliser sur les parties homogènes ou alors sur des zones non-homogènes de l'image. Dans le premier cas, les primitives utilisées sont des régions, et les caractéristiques peuvent être par exemple liées à la texture. L'utilisation des régions pour modéliser la structure du fond présente deux inconvénients : le manque de précision et le temps de calcul induit par le calcul des différentes caractéristiques. Dans le second cas, les primitives peuvent être des discontinuités présentes dans l'image. Parmi ces discontinuités, on distingue notamment les points caractéristiques, les contours, ou plus particulièrement les contours rectilignes. Les points caractéristiques, quoique pouvant être extraits rapidement d'une image, présentent un certain manque de robustesse. En effet, un point caractéristique utilisé pour modéliser la structure de l'arrière-plan peut faire l'objet d'une occlusion temporaire par un objet en mouvement. Il ne sera plus possible alors d'obtenir une structure correcte de l'arrière-plan, et donc d'interpréter le contenu de la séquence vidéo. Afin d'augmenter la robustesse aux phénomènes d'occlusion, il est possible de considérer un plus grand nombre de points caractéristiques mais cela se traduit par une augmentation notable du temps de calcul. A l'opposé, les contours sont souvent nombreux dans une image. Un modèle de structure de l'arrière-plan basé sur les contours de l'image serait donc beaucoup moins sensible aux occlusions induites par les ob-

jets présents dans la scène. Le nombre de contours détectés dans une image pouvant être très élevé, l'utilisation de cette primitive pose deux problèmes principaux : la complexité du modèle de structure obtenu, et le temps de calcul nécessaire pour calculer le modèle. Finalement les contours rectilignes nous ont semblé un bon compromis puisqu'ils garantissent une précision importante, une robustesse aux occlusions, une structuration du fond basée sur un modèle relativement simple. De plus, selon la méthode utilisée, leur extraction peut être obtenue en temps réel.

Nous utiliserons donc les contours rectilignes pour modéliser la structure de l'arrière-plan d'une scène. Par abus de langage, nous utiliserons dans ce chapitre le terme de **lignes** pour désigner ces contours rectilignes. Outre les différentes qualités enoncées précédemment, les lignes permettent notamment de modéliser la plupart des environnements artificiels, c'est-à-dire créés par l'homme. Plus précisément, les lignes horizontales et verticales sont bien adaptées à la modélisation des constructions humaines. Les figures 4.1 et 4.2 illustrent le bien-fondé de notre approche pour des scènes d'intérieur et d'extérieur respectivement : les lignes détectées dans l'image peuvent être mises en correspondance avec le modèle de la scène connu *a priori*. Il est ensuite possible de connaître la position réelle d'un objet à partir de sa position dans l'image.

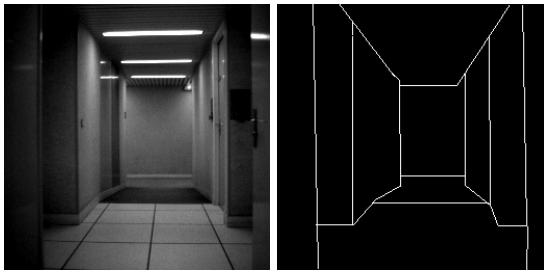

FIG. 4.1 – Modélisation par des lignes de la structure d'une scène intérieure.

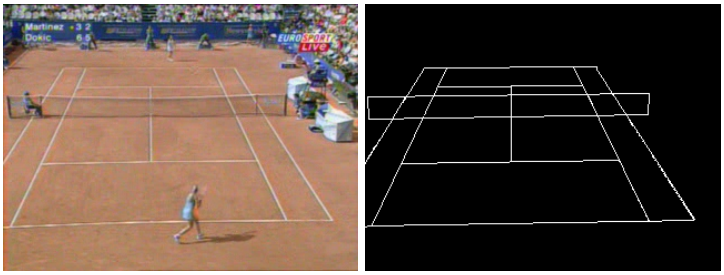

FIG. 4.2 – Modélisation par des lignes de la structure d'une scène extérieure.

Le problème de structuration du fond peut alors être vu comme un problème de détection de lignes, pouvant potentiellement être restreint à la détection des lignes horizontales ou verticales.

Comme nous l'avons énoncé précédemment, deux familles de méthodes permettent de résoudre ce problème : les approches locales et les approches globales. Dans la prochaine section, nous illustrons chacune de ces approches par une méthode représentative.

## 4.2 Approches locales et globales pour la détection de lignes

La détection de lignes est un problème classique de traitement d'images et a été étudié depuis de nombreuses années. La plupart des méthodes proposées peuvent être regroupées en deux familles distinctes selon que l'on considère une analyse au niveau local ou au niveau global. Nous avons préféré illustrer chacune de ces familles par une méthode caractéristique plutôt que de dresser ici un panorama plus exhaustif tel celui de Jeusse [Jeu02].

Afin d'illustrer ces deux familles d'approches, nous détaillons deux méthodes représentatives [Gon92b]. La méthode connue sous le nom d'*edge linking* illustrera le cas des approches locales, et la transformée de Hough celui des approches globales.

### 4.2.1 Une méthode locale : l'approche *edge linking*

Les pixels de contour étant caractérisés par de fortes valeurs de gradient, les informations du gradient (estimé par exemple à l'aide de l'opérateur de Sobel [Sob90]) peuvent être utilisées pour détecter des lignes dans une image. Ainsi, l'approche *edge linking* se base sur le module noté $\|\nabla I\|$ et la direction notée $\angle(\nabla I)$ du gradient, calculés en chaque pixel afin d'extraire les lignes présentes dans l'image. Les conditions pour qu'un pixel $P(x, y)$ appartienne à une ligne sont les suivantes : le pixel doit être associé à une valeur élevée du module de gradient, et il doit exister dans son voisinage un autre point $P'(x', y')$ où le gradient est caractérisé par des modules et directions similaires. Ces deux conditions peuvent être exprimées sous la forme :

$$\|\nabla I(x, y)\| > S_1 \tag{4.1}$$

$$\exists P'(x', y') \in \mathcal{V}(P(x, y)) \quad \text{tel que} \quad \begin{cases} |\angle(\nabla I(x, y)) - \angle(\nabla I(x', y'))| < S_2 \\ |\|\nabla I(x, y)\| - \|\nabla I(x', y')\|| < S_3 \end{cases} \tag{4.2}$$

où $\mathcal{V}(P)$ représente le voisinage du pixel $P$ et $S_1$, $S_2$, et $S_3$ des seuils fixés empiriquement.

### 4.2.2 Une méthode globale : la transformée de Hough

Au lieu de traiter directement les images en niveaux de gris, certaines approches utilisent en entrée les images de contour. Dans ce cas, le problème n'est plus de chercher les contours rectilignes dans l'image mais de chercher, parmi les contours, ceux qui sont rectilignes. Parmi les approches proposées pour résoudre ce problème, la transformée de Hough est probablement la plus connue [Hou62]. Il s'agit d'une méthode globale basée sur une représentation paramétrique des formes recherchées. La transformée consiste en un passage de l'espace de l'image vers un espace de paramètres.

Dans cette méthode, chaque pixel appartenant à un contour vote pour un ensemble de lignes représentées dans un espace paramétrique. Plus précisément, l'espace de représentation $(\rho, \theta)$ est utilisé pour représenter l'espace des droites (c.f. figure 4.3). Les votes sont accumulés dans un tableau (appelé accumulateur) où chaque case est associée à une ligne physique donnée. Finalement, les lignes caractérisées par le plus grand nombre de votes sont considérées. Cette condition se traduit par exemple par la comparaison avec un seuil fixé $S$ qui peut être défini proportionnellement à la taille de l'image. Lorsque l'on souhaite détecter les segments de droites, un post-traitement est nécessaire.

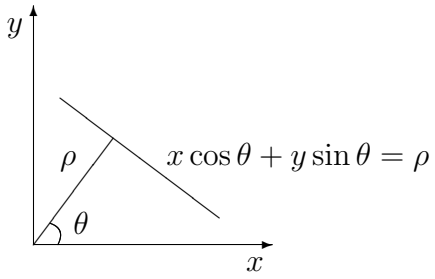


FIG. 4.3 – Représentation d'une droite par un couple $(\rho, \theta)$.

Si, dans un contexte particulier (par exemple des images issues d'un *meeting* d'athlétisme représentant une piste de 100 mètres), nous souhaitons ne conserver qu'un nombre donné $L$ de lignes dans l'image, nous ne considérons que celles caractérisées par $(\rho, \theta)$ associées aux $L$ plus grandes valeurs $A(\rho, \theta)$ de l'accumulateur $A$.

Les inconvénients principaux de cette approche concernent la complexité algorithmique très élevée, le choix du nombre de droites à conserver ou la valeur du seuil à fixer.

La transformée de Hough a donc fait l'objet de nombreuses améliorations. Parmi celles-ci, notons les travaux de Shpilman et Brailovsky [Shp99] qui proposent d'utiliser deux accumulateurs monodimensionnels au lieu de l'accumulateur $A$. La complexité algorithmique de cette

méthode est donc bien plus faible que pour la tranformée de Hough originale.

Après avoir illustré les deux familles de méthodes de détection de lignes par deux approches caractéristiques parmi les plus courantes, la technique d'*edge linking* et la transformée de Hough, nous pouvons remarquer que les approches locales se caractérisent par de faibles temps de calcul mais également par un manque de cohérence au niveau global de l'image. Au contraire, les approches globales fournissent généralement de meilleurs résultats, au détriment de la complexité algorithmique. Afin d'obtenir des résultats de qualité tout en conservant un temps de calcul faible, nous proposons donc dans la section suivante une méthode originale, basée sur l'utilisation d'un accumulateur local tout en tenant compte de plusieurs niveaux de cohérence.

## 4.3 Un accumulateur pour une détection semi-locale

Les méthodes purement locales ou globales ne permettent pas, en général, d'associer qualité des résultats et faible temps de calcul. Nous proposons ici une méthode semi-locale basée sur l'utilisation d'un accumulateur [Lef02c]. La rapidité est assurée par le caractère local de l'étude, tandis que l'accumulateur permet une fusion de points de vue, ce qui fournit un résultat de meilleure qualité.

Deux niveaux d'étude sont considérés successivement. Pour cela, nous divisons l'image en un nombre $B$ de blocs disjoints. La première étape consiste à analyser les différents pixels à l'échelle d'un bloc $b$. Une recherche des segments de lignes, basée sur l'utilisation d'un accumulateur, est effectuée dans chaque bloc de l'image en considérant des directions prédéfinies. La seconde étape permet de s'affranchir du caractère purement local de l'approche. En effet, à l'aide d'une étape de filtrage, nous pouvons assurer la cohérence des résultats à un niveau plus global, le niveau semi-local. La méthode que nous détaillons ici permet de détecter des lignes horizontales ou verticales, mais elle peut facilement être adaptée à d'autres directions. On abordera également le choix des paramètres de l'algorithme dans le but de détecter des lignes caractérisées par des directions différentes des directions horizontale et verticale. Finalement quelques résultats illustreront les qualités de la méthode.

### 4.3.1 Analyse d'un bloc

Puisque nous souhaitons extraire les lignes d'une image pour interpréter le contenu des images en temps réel, le temps de calcul de la méthode à adopter est un critère important. Nous

proposons donc d'analyser l'image d'entrée au niveau local. En outre, la méthode que nous proposons retient le principe de l'accumulation d'indices sur laquelle repose la transformée de Hough. De ce fait, elle présente les mêmes avantages en ce qui concerne la qualité des résultats tout en évitant les inconvénients inhérents à l'approche globale.

Nous considérons que la détection de contours a déjà été effectuée (par exemple par seuillage du gradient calculé avec l'opérateur de Sobel [Sob90]) et qu'il en résulte une image binaire de contours $I_{\text{contours}}$. Dans le cas d'images couleur, des méthodes de détection de contours spécifiques peuvent être utilisées [Fan01]. Parmi celles-ci, certaines considèrent l'espace TSL dont les principaux intérêts ont été donnés dans la section 2.2.2. On peut ainsi citer les travaux de Carron et Lambert [Car94].

Nous considérons l'image $I_{\text{contours}}$ comme un ensemble de blocs disjoints. Chaque bloc est composé de $X_{\text{bloc}} \times Y_{\text{bloc}}$ pixels. Pour chaque bloc $b$, un accumulateur local $A^b$ est créé et analysé pour déterminer la présence et l'orientation d'un potentiel segment de ligne. Au sein de chaque bloc, nous proposons de définir un nombre $K_{\text{h}}$ de zones ou régions horizontales et un nombre $K_{\text{v}}$ de zones ou régions verticales. Ces régions, notées $R_k^b$, peuvent être disjointes ou alors se chevaucher. La figure 4.4 illustre la création de ces régions avec des valeurs de $K_{\text{h}}$ et $K_{\text{v}}$ toutes deux égales à 3, ce qui a pour résultat la création de 3 régions se chevauchant, dans chaque direction. Dans un bloc donné $b$, on considère un nombre de régions $R_k^b$ égal à $K_{\text{h}} + K_{\text{v}}$, où $k$ appartient à l'intervalle $[1, K_{\text{h}} + K_{\text{v}}]$. L'accumulateur local $A^b$ est associé à ces différentes régions et contient un compteur $A^b(k)$ par région $R_k^b$. Il est ainsi composé de $K_{\text{h}}+K_{\text{v}}$ compteurs, notés $\left\{A^b(1), \ldots, A^b(K_{\text{h}} + K_{\text{v}})\right\}$.

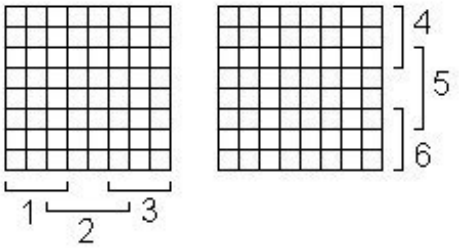


FIG. 4.4 – Disposition de régions $R_k^b$ correspondant à des lignes verticales (à gauche) et des lignes horizontales (à droite).

Nous ne considérons ici que les lignes horizontales et verticales. Celles-ci ont montré leur intérêt dans la modélisation de la scène, notamment quand la nature de l'arrière-plan est artificielle (construite par l'homme). Nous rappelons cependant que la méthode peut être facilement adaptée à d'autres directions.

L'accumulateur est utilisé pour stocker les informations contenues dans l'image de contour $I_{\text{contours}}$. Chaque pixel $P(x, y)$ de contour dans le bloc $b$ vote pour des lignes particulières *via* le compteur associé dans l'accumulateur. Plus précisément, le pixel concerné votera pour au moins deux régions $R_k^b$ du bloc $b$, une dans chaque direction. Le vote dépend uniquement de la position spatiale du pixel, et conduit à l'incrémentation du compteur associé dans l'accumulateur. Il peut être formulé par :

$$A^b(k) \leftarrow A^b(k) + 1 \quad \text{si} \quad P(x, y) \in R_k^b \tag{4.3}$$

Les valeurs contenues dans l'accumateur sont normalisées par rapport à la taille de chaque région afin de ne pas favoriser les zones les plus étendues.

Une fois que tous les pixels du bloc $b$ ont été parcourus, la dernière étape de l'algorithme consiste en l'analyse des compteurs de l'accumulateur. On cherche alors le compteur caractérisé par la valeur la plus élevée. Si cette valeur est inférieure à un seuil $S$, nous considérons que le nombre de votes pour la région associée n'est pas suffisant. La décision est alors qu'aucune ligne n'a été détectée dans le bloc. Dans le cas contraire, la ligne détectée $R_{k'}^b$ est celle caractérisée par le compteur $A^b(k')$ le plus élevé :

$$k' = \underset{\substack{k \in [1, K_{\text{h}} + K_{\text{v}}] \\ A(k) \geq S}}{\arg\max} \left( A(k) \right) \tag{4.4}$$

L'approche décrite ici est basée sur une analyse purement locale des valeurs des pixels, en utilisant un jeu de paramètres $\lambda = \{X_{\text{bloc}}, Y_{\text{bloc}}, K_{\text{h}}, K_{\text{v}}, S\}$. La cohérence des résultats obtenus dans des blocs voisins n'est pas garantie. Nous proposons donc d'utiliser en post-traitement, une étape de filtrage, afin d'introduire un second niveau d'observation et d'améliorer la cohérence des résultats.

### 4.3.2 Etude de la cohérence

L'approche locale décrite précédemment se base sur l'analyse de blocs de $X_{\text{bloc}} \times Y_{\text{bloc}}$ pixels, avec $X_{\text{bloc}} \ll X$ et $Y_{\text{bloc}} \ll Y$. Nous avons choisi dans la pratique une taille de bloc très faible par rapport à la taille de l'image. Il n'est pas sûr que des segments de lignes détectés dans des blocs voisins soient cohérents. Nous avons donc introduit une étape de filtrage décrite ici, et qui permet d'améliorer la robustesse de l'approche proposée précédemment.

Chaque bloc contenant un segment de ligne est comparé à ses voisins pour déterminer si la direction de la ligne détectée est cohérente avec le voisinage. Plus précisément, nous analysons les paires de blocs $b_i b_j$ construites en prenant des blocs voisins de $b$ et symétriques par rapport

à $b$. La figure 4.5 montre des exemples de paires de blocs voisins $b_1b_5$, $b_2b_6$ et $b_3b_7$ dans le cas où une ligne horizontale a été détectée dans $b$. Dans ce cas précis, la ligne horizontale détectée dans $b$ ne sera conservée que si l'une des trois conditions suivantes est respectée :

- les blocs $b_1$ et $b_5$ contiennent tous deux des lignes horizontales,
- les blocs $b_2$ et $b_6$ contiennent tous deux des lignes horizontales,
- les blocs $b_3$ et $b_7$ contiennent tous deux des lignes horizontales.

La vérification dans le cas d'une ligne verticale suit un principe similaire. Enfin, il est possible de réaliser des filtres plus avancés en utilisant cette procédure dans un voisinage étendu de $b$.

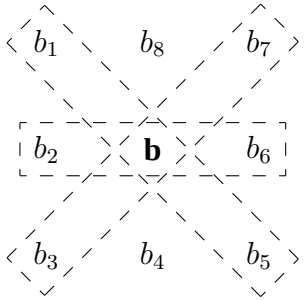


FIG. 4.5 – Paires de blocs voisins $b_1b_5$, $b_2b_6$, et $b_3b_7$ utilisés dans le filtrage d'un bloc $b$ contenant une ligne horizontale.

Le filtrage permet d'améliorer la cohérence des segments de lignes verticaux ou horizontaux détectés au moyen de l'accumulateur local. Il est aussi possible d'adapter cette approche à la détection de lignes caractérisées par d'autres directions.

### 4.3.3 Extension à d'autres directions

Nous n'avons considéré précédemment que les lignes horizontales et verticales (de directions $\theta$ respectivement égales à $0$ ou $\pi$, et $\pi/2$ ou $3\pi/2$), puisqu'elles permettent de modéliser l'arrière-plan de scènes artificielles. La direction de ce type de lignes peut plus généralement être notée $\lambda\pi/2$ avec $\lambda \in \mathbb{N}$.

La méthode peut cependant être adaptée facilement à d'autres directions $\theta = \alpha\pi/\omega$ avec $\alpha, \omega \in \mathbb{N}$. En effet, chaque nouvelle direction $\theta$ à considérer doit faire l'objet de la création d'une région associée $R_k^b$ dans chaque bloc $b$. Cependant, si l'on considère des blocs de petite taille ($X_{\text{bloc}}$ et $Y_{\text{bloc}}$ faibles), il est difficile de définir précisément la direction $\theta$. Il est alors conseillé d'utiliser des blocs plus larges.

L'algorithme 3 page 68 présente la détection de lignes dans le cas général (directions quelconques définies par des régions $R_k^b$).

**Algorithme 3:** Détection de lignes par accumulateur local.

**Entrée** : Image binaire de contours $I_{\text{contours}}$
Paramètres $X_{\text{bloc}}$, $Y_{\text{bloc}}$, $K_{\text{h}}$, $K_{\text{v}}$, et $S$
**Sortie** : Image binaire de lignes $I_{\text{lignes}}$ de même taille que $I_{\text{contours}}$

*Initialisation*
Initialisation de l'image $I_{\text{lignes}}$ à zéro
*Analyse locale des différents blocs de l'image*
**pour** $b \leftarrow 1$ **à** $B$ **faire**
  Initialisation de l'accumulateur $A^b$ à zéro
  *Remplissage de l'accumulateur* $A^b$
  **pour** $x \leftarrow 1$ **à** $X_{\text{bloc}}$ **faire**
    **pour** $y \leftarrow 1$ **à** $Y_{\text{bloc}}$ **faire**
      **si** $I^b_{\text{contours}}(x, y) = 1$ **alors**
        **pour** $k \leftarrow 1$ **à** $K$ **faire**
          **si** $P(x, y) \in R^b_k$ **alors**
            $A^b(k) \leftarrow A^b(k) + 1$
  *Analyse de l'accumulateur* $A^b$
  Obtention du compteur maximal $k'$
  **si** $A^b(k') \geq S$ **alors**
    Afficher la région $R^b_k$ dans l'image $I_{\text{lignes}}$
*Vérification de la cohérence*
**pour** $b \leftarrow 1$ **à** $B$ **faire**
  **si** $b$ contient une ligne **alors**
    Déterminer les paires de blocs voisins selon la direction de la ligne détectée
    **si** $b$ n'est pas cohérent avec ses voisins **alors**
      Eliminer de l'image $I_{\text{lignes}}$ la ligne détectée dans $b$

Les résultats obtenus avec cette méthode vont maintenant être commentés.

### 4.3.4 Résultats

Nous avons testé notre méthode sur des images binaires de contours. Les contours ont été obtenus en utilisant le gradient calculé par l'opérateur de Sobel [Sob90] sur des images en niveaux de gris, comme le montre la figure 4.6.

Nous cherchons les lignes présentes dans une image afin d'effectuer un recalage avec un

FIG. 4.6 – Image originale en niveaux de gris (à gauche) et image binaire de contours calculée par l'opérateur de Sobel (à droite).

modèle de la scène connu *a priori*. Cependant, la détection de lignes peut servir dans de nombreux domaines tels l'interprétation d'images satellites, la compréhension de la structure d'un document, *etc*. Nous devons donc adapter le processus à l'application envisagée. Pour cela nous considérons deux cas extrêmes :

- dans le premier cas, on cherchera à obtenir un résultat aussi précis que possible, en s'assurant que les segments de lignes détectés soient complètement présents dans l'image d'entrée, c'est-à-dire qu'aucun pixel ne manquera dans le segment concerné ; le nombre de lignes détectées sera alors réduit ;
- dans le second cas, le but est d'obtenir les lignes présentes dans l'image d'entrée en considérant une certaine tolérance au bruit ajouté par la détection de contours.

Pour traiter ces différents cas de figure, nous devons choisir des paramètres adéquats.

Dans les résultats présentés ci-dessous, nous utilisons les deux jeux de paramètres décrits dans le tableau 4.1. Ces paramètres ne dépendant pas du type d'image analysée mais plutôt du type de résultat souhaité. Les paramètres $X_{\text{bloc}}, K_{\text{h}}$ et $Y_{\text{bloc}}, K_{\text{v}}$ sont choisis selon la précision spatiale et la tolérance directionnelle (angulaire) souhaitées. Le seuil de tolérance $S$ est lié au bruit admis dans le segment de ligne, et permet de compenser en partie le bruit ajouté par l'étape de détection de contours. Quand on ne désire pas tolérer le moindre bruit dans le segment de ligne, on doit alors utiliser le jeu de paramètres $\lambda_2$. Celui-ci est particulièrement adapté au premier cas décrit dans le paragraphe précédent. Dans ce cas, les régions (qu'elles soient verticales ou horizontales) ne se chevauchent pas. Les paramètres $K_{\text{h}}$, $K_{\text{v}}$ et $S$ peuvent être diminués afin d'obtenir un traitement plus tolérant, donnant par exemple le jeu de paramètres $\lambda_1$. Celui-ci est adapté au second cas décrit dans le paragraphe précédent. Les résultats présentent pour une même image l'utilisation des deux jeux de paramètres $\lambda_1$ et $\lambda_2$.

La figure 4.7 montre les résultats de l'approche proposée ici en utilisant les deux jeux différents de paramètres $\lambda_1$ et $\lambda_2$. On peut vérifier que l'utilisation de $\lambda_1$ permet d'obtenir plus de segments de lignes mais la position spatiale de ces segments est moins précise qu'en utilisant

| Paramètre | Description | $\lambda_1$ | $\lambda_2$ |
|---|---|---|---|
| $X_{\text{bloc}}$ | Largeur d'un bloc | 8 | 8 |
| $Y_{\text{bloc}}$ | Hauteur d'un bloc | 8 | 8 |
| $K_{\text{h}}$ | Nombre de régions horizontales | 3 | 8 |
| $K_{\text{v}}$ | Nombre de régions verticales | 3 | 8 |
| $S$ | Nombre de votes requis | 5 | 8 |

TAB. 4.1 – Description des deux jeux de paramètres utilisés pour obtenir les résultats présentés dans les figures 4.7 et 4.10.

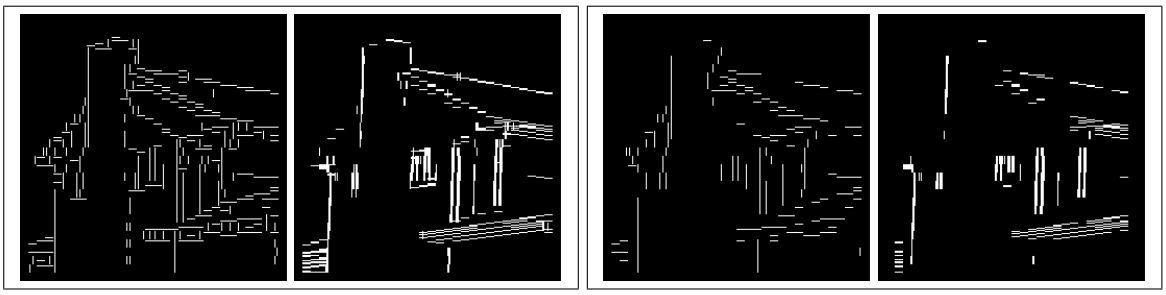

FIG. 4.7 – Résultats de la méthode par accumulateur local sans (cadre de gauche) et avec (cadre de droite) la vérification de cohérence. Dans chaque cas, les jeux de paramètres $\lambda_1$ (image de gauche) et $\lambda_2$ (image de droite) ont été utilisés.

l'autre jeu de paramètres. Au contraire, quand les images sont traitées avec $\lambda_2$, la précision des résultats est améliorée mais le nombre de segments de lignes est réduit. Donc $\lambda_1$ ou $\lambda_2$ sera plus ou moins adapté selon le contexte de l'application. De plus, on peut voir sur la partie gauche de la figure 4.7 que la cohérence des segments de lignes détectés dans des blocs voisins n'est pas assurée. Il est donc nécessaire d'appliquer l'étape de filtrage afin de vérifier la cohérence des résultats obtenus. La partie droite de la figure 4.7 montre les résultats obtenus avec les deux jeux de paramètres en considérant l'étape de filtrage décrite par la figure 4.5.

En considérant les jeux de paramètres $\lambda_1$ et $\lambda_2$, l'image de la figure 4.6 de taille $256 \times 256$ pixels, et une architecture PC Pentium 4 candencé à 1700 Mhz, le temps de calcul nécessaire est égal à 5 millisecondes. Plus la taille des blocs (définie par $X_{\text{bloc}}$ et $Y_{\text{bloc}}$) est importante, plus le temps de calcul requis sera faible, mais moins précis sera le résultat. Afin d'obtenir une précision constante tout en faisant varier la taille des blocs, il est nécessaire de faire également varier le nombre de régions $K_{\text{h}}$ et $K_{\text{v}}$. Nous avons ainsi observé que les temps de calcul obtenus étaient constants si les rapports $\frac{X_{\text{bloc}}}{K_{\text{v}}}$ et $\frac{Y_{\text{bloc}}}{K_{\text{h}}}$ restaient également constants. Néanmoins les longueurs des segments obtenus ne sont pas identiques.

Afin de valider l'approche proposée, nous l'avons comparée à différentes méthodes de la

littérature [Gon92b], notamment celles présentées dans la section 4.2. La comparaison aurait évidemment pu être étendue à un ensemble plus complet de méthodes de la littérature, soit locales [Shp99], soit globales [Kas99]. Nous avons utilisé dans notre protocole de comparaison, outre les techniques présentées dans la section 4.2 et celle que nous avons introduite dans ce chapitre, deux autres approches.

La première consiste en l'utilisation de masques unidirectionnels pour détecter des lignes horizontales ou verticales. Ces masques sont définis respectivement par :

$$M_{\text{horizontal}} = \begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix} \quad \text{et} \quad M_{\text{vertical}} = \begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix} \tag{4.5}$$

Le résultat de la convolution de l'image avec les masques est une image contenant soit des lignes horizontales, soit des lignes verticales. Afin d'obtenir les lignes dans les deux directions, il est nécessaire de combiner ces deux images. En notant $I_{\text{combinaison}}$ l'image obtenue, cette méthode peut s'écrire sous la forme :

$$I_{\text{combinaison}} = (M_{\text{horizontal}} * I) \vee (M_{\text{vertical}} * I) \tag{4.6}$$

où $*$ dénote l'opérateur de convolution.

La seconde, proposée par Frei et Chen [Fre77], consiste tout d'abord à convoluer l'image avec un ensemble de 9 masques orthogonaux. Les différents masques permettent de caractériser l'état d'un pixel selon la réponse impulsionnelle obtenue. Pour cela, les masques sont regroupés en trois catégories permettant de caractériser un pixel comme un point moyen, un point de contour, ou un point de ligne.

La figure 4.8 illustre les résultats de la détection de lignes effectuée en utilisant la méthode *edge linking* décrite dans la section 4.2.1, les masques orthogonaux de Frei et Chen, et finalement la combinaison de masques horizontaux et verticaux. Comme le montre cette figure, la compression des informations n'est pas assez importante pour permettre une compréhension aisée de la scène. En effet, de nombreux points de contours isolés sont conservés, et l'image résultante ne consiste pas uniquement en un ensemble de segments de lignes. Ce type de résultat est cependant nécessaire pour réaliser une modélisation correcte de la scène à l'aide d'une détection de lignes.

Nous avons aussi comparé l'approche par accumulateur local avec une méthode utilisant la transformée de Hough. Cette comparaison se justifie par le fait que l'approche proposée peut être vue comme une adaptation rapide et locale de la transformée de Hough. Le résultat obtenu

à l'aide de la transformée de Hough, tiré de [Kal96], et basé sur l'utilisation de XHoughTool, est présenté dans la figure 4.9. Les lignes sont correctement détectées mais nous pouvons observer que deux lignes sont manquantes. Nous avons également comparé les temps de calcul nécessaires aux deux techniques. L'approche que nous proposons ici est de l'ordre de 1500 fois plus rapide que la transformée de Hough.

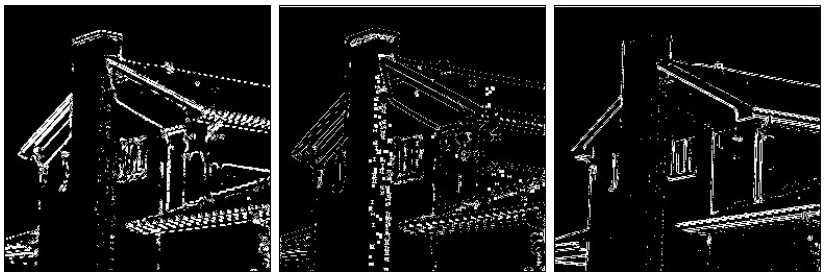

FIG. 4.8 – Lignes détectées par la méthode *edge linking* (à gauche), l'utilisation des masques orthogonaux de Frei et Chen (au centre), et la combinaison de masques horizontaux et verticaux (à droite).

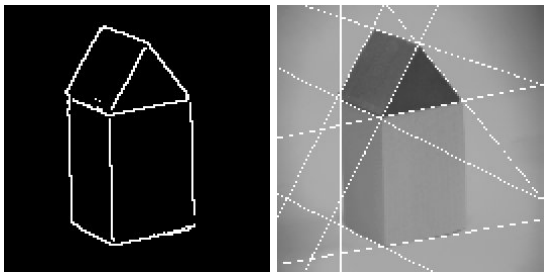

FIG. 4.9 – Image originale de contours (à gauche) et résultat obtenu par la transformée de Hough (à droite) en utilisant XHoughTool comme présenté dans [Kal96].

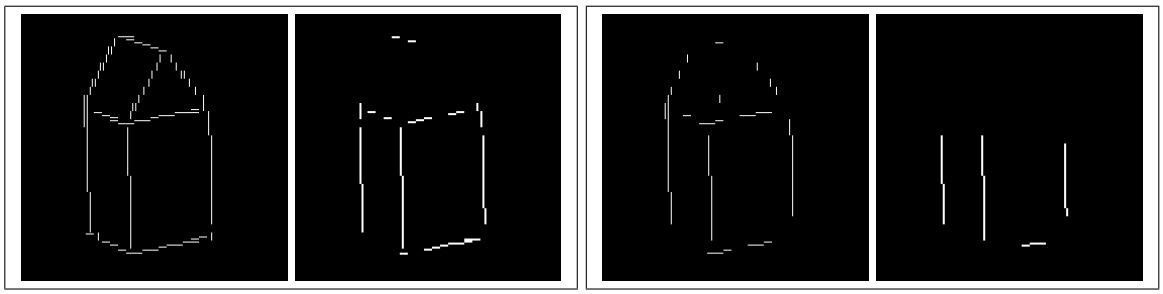

FIG. 4.10 – Résultats de la méthode par accumulateur local sans (cadre de gauche) et avec (cadre de droite) la vérification de cohérence. Dans chaque cas, les jeux de paramètres $\lambda_1$ (image de gauche) et $\lambda_2$ (image de droite) ont été utilisés.

La figure 4.10 illustre les résultats obtenus avec l'accumulateur local en considérant les ensembles de paramètres $\lambda_1$ et $\lambda_2$. Il est ici possible de conserver uniquement les segments de lignes présents dans l'image d'entrée lorsqu'on effectue la détection de lignes avec une certaine tolérance (en utilisant $\lambda_1$). $\lambda_2$ est particulièrement adapté lorsque la détection de lignes a pour but de modéliser des scènes artificielles, composées la plupart du temps de lignes verticales.

## 4.4 Conclusion

Dans ce chapitre, nous avons traité le problème de la structuration du fond. Celle-ci permet de modéliser la scène afin de pouvoir ensuite interpréter les positions des objets en mouvement à un instant donné. Nous avons choisi d'utiliser les lignes (contours rectilignes) verticales ou horizontales comme primitives pour représenter la structure de l'arrière-plan. Celles-ci sont particulièrement adaptées à des environnements artificiels, c'est-à-dire construits par l'homme (c.f. figures 4.1 et 4.2). Le problème que nous avions à résoudre peut donc être assimilé à un problème de détection de lignes (horizontales et verticales) dans une image. Nous devions également prendre en compte les contraintes $\mathcal{C}_{\text{rapidité}}$, $\mathcal{C}_{\text{couleur}}$, $\mathcal{C}_{\text{illumination}}$.

Afin de respecter au mieux la contrainte $\mathcal{C}_{\text{rapidité}}$, nous nous sommes orienté vers une approche locale. Nous souhaitions cependant obtenir des résultats aussi bons qu'avec des approches globales. Nous avons donc retenu le principe de l'accumulation d'indices sur laquelle repose la transformée de Hough. De plus, pour éviter les défauts d'une approche purement locale, nous avons introduit une étape de vérification de la cohérence des résultats à un niveau semi-local. La méthode que nous proposons s'applique sur des images de contour. Afin de respecter les contraintes $\mathcal{C}_{\text{couleur}}$ et $\mathcal{C}_{\text{illumination}}$, il est nécessaire d'utiliser une méthode de détection de contours adaptée.

Nous avons traité dans cette partie le problème de l'étude de l'arrière-plan de la scène. Pour cela, nous avons abordé d'une part la séparation du fond et des objets, et d'autre part la structuration du fond. Nous rappelons que la séparation du fond et des objets permet l'initialisation d'algorithmes de suivi d'objet, tandis que la structuration du fond permet un changement d'espace de représentation spatiale, depuis l'espace de l'image (virtuel) vers l'espace de la scène (réél). Dans la partie suivante, nous abordons le problème de l'étude des objets de la scène. Ce problème est plus connu sous le nom de suivi d'objet.

# Troisième partie

# Suivi des objets

Lorsque la séquence vidéo a été décomposée en plans (c.f. partie I), il est possible d'analyser le contenu de chaque plan afin d'interpréter la séquence vidéo. Les images d'un plan sont généralement composées d'une scène (représentée par son arrière-plan) et de différents objets en mouvement présents dans la scène. Dans la partie II, nous avons étudié l'arrière-plan de la scène. Cette étude a permis, d'une part, de séparer le fond et les objets (c.f. chapitre 3), et d'autre part, de modéliser la structure de l'arrière-plan de la scène (c.f. chapitre 4). Dans cette partie, nous étudierons les objets en mouvement présents dans la scène. Plus précisément, nous aborderons le problème du suivi d'objet, qui peut être énoncé comme la détermination de la position d'un objet à l'instant $t$ connaissant sa position à l'instant $t-1$. Dans la première image d'un plan ($t=1$), le suivi n'est pas possible car la position précédente de l'objet est inconnue. On utilisera dans ce cas les résultats du chapitre 3 sur la séparation des objets et du fond qui nous donnent les positions des objets dans une image. Les algorithmes de suivi d'objet permettant d'obtenir la position dans l'image d'un objet à l'instant $t$, il est alors possible d'interpréter le contenu de la scène en combinant cette information avec la structure connue de l'arrière-plan, et que l'on peut modéliser par l'approche décrite dans le chapitre 4.

Le suivi d'objet a suscité de nombreux travaux. Nous le considérons comme une étape pour la détection d'événements dans des séquences vidéo, et non comme un but en soi. Aussi nous semble-t-il hors de propos de dresser ici un état de l'art des méthodes existantes. On pourra se référer aux articles de Cedras et Shah [Ced95] et de Mitiche et Bouthemy [Mit96]. De plus, plusieurs thèses abordant ce problème fournissent des éléments d'état de l'art [Deu97, Bré97, Cas98, Lac00, Ham01].

Afin d'intégrer un algorithme de suivi d'objet dans un système d'analyse vidéo, nous devons ici aussi tenir compte des aspects $\mathcal{C}_{\text{rapidité}}$, $\mathcal{C}_{\text{couleur}}$, $\mathcal{C}_{\text{illumination}}$, $\mathcal{C}_{\text{mouvement}}$ mais également de :

$\mathcal{C}_{\text{variabilité}}$ : les objets suivis peuvent être de différentes natures (donc de différentes tailles, de différentes formes, *etc.*).

L'algorithme optimal serait donc celui permettant de suivre des objets de différentes natures en temps réel dans des images couleur acquises avec une caméra en mouvement où les changements d'illumination sont fréquents. La mise en place d'un algorithme unique pour résoudre ce problème nous semblant pour le moins difficile, nous nous proposons plutôt d'utiliser différents algorithmes suivant la nature de l'objet suivi. Plus précisément, nous avons déterminé quatre classes d'objets : les objets rigides pouvant faire l'objet d'un apprentissage, les objets rigides pour lesquels on ne dispose pas de données permettant leur apprentissage, les objets de taille trop faible pour être appris, et les objets non-rigides. Cette partie sera donc composée de trois chapitres consacrés au suivi de ces différents types d'objets. Nous n'aborderons pas ici le cas des objets rigides pour lesquels on ne dispose pas de données permettant leur apprentissage.

Dans le chapitre 5, nous traitons le problème du suivi d'objet rigide avec apprentissage. Pour cela, nous allons utiliser les chaînes de Markov cachées multidimensionnelles pour modéliser l'objet à suivre. Après un apprentissage effectué avec l'algorithme GHOSP, l'étape de suivi est modélisée comme un problème relatif aux chaînes de Markov cachées. Ce peut être le problème de l'évaluation, ou celui de la recherche du meilleur chemin d'états. L'algorithme Forward et l'algorithme de Viterbi sont donc respectivement employés et conduisent à deux méthodes de suivi d'objet.

Les objets rigides ne peuvent pas toujours faire l'objet d'un apprentissage préalable au suivi. En effet, on ne dispose pas dans certains cas (par exemple si la taille de l'objet est relativement faible) d'une quantité suffisante d'informations concernant l'objet pour pouvoir effectuer un apprentissage robuste. Nous proposons donc dans le chapitre 6 des méthodes adaptées au suivi d'objets de petite taille, sans apprentissage préalable.

Enfin, nous aborderons le cas des objets non-rigides dans le chapitre 7. Afin de modéliser ce type d'objet, nous utilisons un outil approprié : les contours actifs ou *snakes*. Cet outil, quoique bien adapté aux objets non-rigides, présente un inconvénient majeur : sa forte complexité algorithmique. Nos travaux dans ce domaine ont porté sur la diminution du temps de calcul nécessaire aux algorithmes basés sur les contours actifs. Il en résulte une méthode rapide de suivi par contours actifs.

# Chapitre 5

# Suivi d'objet rigide avec apprentissage

Lorsque les objets à suivre ont une nature rigide, il est possible de les modéliser. Cette modélisation est obtenue par une étape d'apprentissage pouvant être effectuée en ligne ou hors ligne. Le suivi réalisé en ligne consiste alors en la recherche et la reconnaissance de l'objet (par l'utilisation de son modèle créé lors de l'apprentissage) dans les différentes images de la séquence. Cette recherche n'est généralement effectuée que dans une partie de l'image, définie de manière contextuelle en fonction du mouvement et de la position de l'objet dans les images précédentes. Dans ce chapitre, nous proposons d'utiliser les chaînes de Markov cachées multidimensionnelles à processus indépendants (notées CMC-MD/I) comme outil pour modéliser l'objet à suivre. L'étape d'apprentissage permettra de construire des CMC-MD/I modélisant l'objet rigide. L'étape de suivi consistera en la recherche de l'objet dans les images successives par utilisation des CMC-MD/I générées lors de l'apprentissage [Lef02b].

Le plan de ce chapitre est donc le suivant. Tout d'abord nous présenterons la problématique associée au suivi d'objet rigide. Puis nous détaillerons l'outil utilisé, les CMC-MD/I. Plus précisément, après avoir introduit les chaînes de Markov cachées (notées CMC) et décrit les principaux problèmes liés à ces outils, nous présenterons les CMC-MD/I qui sont une extension des CMC au cas multidimensionnel. De plus nous dresserons un bref panorama des travaux utilisant ces outils en analyse d'images. Le problème de l'apprentissage sera en outre abordé et l'algorithme utilisé, appelé GHOSP, sera notamment décrit. Les deux principaux problèmes associés aux CMC sont liés à l'évaluation de l'adéquation du modèle à l'observation et à la détermination du meilleur chemin d'états. Ces deux problèmes peuvent être respectivement résolus par l'algorithme Forward et l'algorithme de Viterbi. Nous proposons donc ensuite de ramener la recherche de l'objet appris dans les images à ces deux problèmes. Deux méthodes de suivi, basées respectivement sur l'algorithme Forward et sur l'algorithme de Viterbi, découlent

de ce principe et seront décrites dans les sections suivantes. Une adaptation de ces méthodes à la détection de l'objet dans une image sera ensuite étudiée. Elle est valable même lorsqu'aucune information de mouvement n'est présente. Finalement, nous comparerons le suivi d'objet par CMC-MD/I avec une approche classique de la littérature, basée sur le *template matching*.

## 5.1 Problématique

Un objet caractérisé par une nature rigide présente des apparences assez similaires dans les différentes images d'une séquence vidéo. En effet, sa forme n'étant pas modifiée, sa représentation dans l'image est généralement identique tout au long d'une séquence. Les deux principales raisons d'une modification de l'apparence de l'objet dans l'image sont liées au changement d'éclairage de l'objet par des sources lumineuses et à la rotation 3-D de l'objet dans la scène. Dans de nombreuses applications, la rotation de la représentation de l'objet entre deux images s'effectue en première approximation dans le plan de l'image, comme par exemple dans le cas du suivi de véhicules pour l'analyse de trafic routier. Dans d'autres applications, la nature de l'objet permet d'assimiler un mouvement 3-D à une rotation 2-D. Ainsi, lors du suivi de balles dans des retransmissions sportives, la nature sphérique de l'objet suivi permet de ne considérer un mouvement rotationnel que dans le plan de l'image. Dans ce chapitre, nous supposerons que les rotations de l'objet suivi ne sont effectuées qu'en 2-D, ou alors qu'elles peuvent être approximées par des rotations apparentes en 2-D.

Il est donc possible d'apprendre l'objet, de le modéliser, dans un premier temps pour le reconnaître et le suivre par la suite dans une séquence. L'apprentissage s'effectue à partir de plusieurs images représentant l'objet à apprendre. Afin d'obtenir un modèle robuste aux changements d'illumination et aux rotations 2-D de l'objet, plusieurs images seront utilisées lors de l'étape d'apprentissage. Ces images représenteront l'objet dans différentes positions et sous différentes conditions d'éclairage.

L'apprentissage de l'objet à suivre permet d'obtenir un modèle de l'objet. Ce modèle est alors utilisé lors du suivi dans les différentes images. Le suivi consiste donc dans ce cas à rechercher une forme associée au modèle, en employant les informations disponibles *a priori* sur la position et le mouvement de l'objet, plutôt qu'à détecter des objets en mouvement dans la séquence. Nous avons choisi de résoudre ce problème de reconnaissance des formes à l'aide des chaînes de Markov cachées.

Les chaînes de Markov cachées (CMC) sont des outils mathématiques permettant de modéliser des phénomènes physiques connus par des variables aléatoires représentant les états d'un

processus stochastique discret. Dans le domaine de l'analyse d'images, la partie observable de la chaîne correspond aux données présentes dans l'image (intensité, gradient, *etc.*). Les CMC sont bien adaptées aux phénomènes monodimensionnels, aux phénomènes temporels. Elles sont fréquemment utilisées dans le domaine du suivi d'objet pour modéliser le mouvement de l'objet en fonction du temps. Les CMC sont alors employées comme un outil pour estimer ou prédire le mouvement. Une fois le mouvement estimé, la position de l'objet peut être affinée en utilisant un traitement supplémentaire.

Dans ce chapitre, nous proposons d'utiliser les CMC pour le suivi d'objet dans un cadre résolument différent des approches traditionnelles de la littérature. En effet, nous décomposons le problème du suivi d'objet en deux problèmes distincts devant être résolus successivement pour chaque image : la prédiction d'une position approximative de l'objet, et la recherche de la position exacte de l'objet. Contrairement aux autres approches basées sur les CMC, la prédiction de la position approximative de l'objet est obtenue non pas à l'aide des CMC, mais en utilisant un estimateur de mouvement relativement simple. La recherche de la position exacte de l'objet dans l'image courante est ensuite limitée dans une zone construite autour de la position prédite de l'objet. Cette recherche, qui peut être considérée comme une étape de détection d'objet dans une zone limitée de l'image, est effectuée en utilisant les CMC. Contrairement aux approches traditionnelles utilisant les CMC, nous employons les CMC pour modéliser la variation spatiale des intensités des pixels dans une image donnée (c'est-à-dire les caractéristiques spatiales de l'objet telles que la couleur, la texture, ou la forme) et non pour modéliser la variation temporelle des pixels des objets dans les images successives (c'est-à-dire le mouvement des objets). L'architecture choisie pour la CMC est une architecture ergodique contrairement à celle gauche-droite utilisée pour les modèles temporels. Afin de gérer des images couleurs, nous utilisons en outre un modèle Markovien particulier, les chaînes de Markov cachées multidimensionnelles à processus indépendants, introduites par Brouard dans sa thèse [Bro99].

## 5.2 Les chaînes de Markov cachées multidimensionnelles

Dans cette section, nous débuterons par une introduction aux chaînes de Markov cachées (CMC) en précisant les notations utilisées. Afin de traiter des données multidimensionnelles, nous utilisons un modèle markovien particulier : les chaînes de Markov cachées multidimensionnelles [Bro99] que nous décrirons en insistant sur la différence avec les chaînes de Markov cachées classiques.

### 5.2.1 Les chaînes de Markov cachées

Les chaînes de Markov cachées sont des outils statistiques fréquemment utilisés en reconnaissance des formes. Une introduction relativement complète de ces outils et de leurs applications peut être consultée dans le tutorial de Rabiner [Rab89]. Nous nous limiterons ici à une brève présentation de ces outils.

Une chaîne de Markov cachée est définie comme un ensemble de deux variables aléatoires permettant de représenter les états d'un processus stochastique discret. La première, appelée la partie cachée, définit un automate tandis que la seconde, appelée la partie visible ou observable, définit comment un état donné de l'automate génère un symbole donné.

Une CMC est caractérisée par :

– $S$ : l'ensemble des états cachés de la CMC (de cardinalité $N$),

$$S = \{s_1, s_2, \ldots, s_N\} \tag{5.1}$$

– $V$ : l'ensemble des symboles générables par la CMC (de cardinalité $M$),

$$V = \{v_1, v_2, \ldots, v_M\} \tag{5.2}$$

– $B$ : la matrice $(N \times M)$ de distribution des probabilités de génération des symboles,

$$B = \{b_j(v_k)\}_{\substack{j\in[1,N]\\k\in[1,M]}} \tag{5.3}$$

où $b_j(v_k)$ désigne la probabilité que la CMC qui se trouve dans l'état $s_j$ génère le symbole $v_k$, ce qui donne à l'instant $t$ (en notant $o_t$ l'observation et $q_t$ l'état à un instant $t$ quelconque) :

$$\forall j \in [1, N] \quad \forall k \in [1, M] \qquad b_{jk} = b_j(v_k) = P(o_t = v_k \mid q_t = s_j) \tag{5.4}$$

– $A$ : la matrice de distribution des probabilités de transition entre les états,

$$A = \{a_{ij}\}_{i,j\in[1,N]} \tag{5.5}$$

avec :

$$\forall i \in [1, N] \quad \forall j \in [1, N] \qquad a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i) \tag{5.6}$$

– $\Pi$ : le vecteur de distribution des probabilités de l'état initial,

$$\Pi = \{\pi_i\}_{i\in[1,N]} \tag{5.7}$$

où $\pi_i$ est la probabilité que la CMC soit dans l'état $s_i$ à l'instant initial :

$$\forall i \in [1, N] \qquad \pi_i = P(q_1 = s_i) \tag{5.8}$$

En utilisant ces notations, une CMC peut alors être modélisée par le triplet $\lambda = (A, B, \Pi)$.

### 5.2.2 Principaux problèmes liés aux CMC

Lors de l'utilisation des CMC, trois problèmes principaux doivent être résolus. Ils concernent respectivement :

– l'évaluation de l'adéquation du modèle à l'observation ;
– la détermination du meilleur chemin d'états ;
– l'apprentissage (supervisé ou non) du modèle associé au phénomène étudié.

Le problème de l'évaluation consiste à déterminer la probabilité qu'une CMC $\lambda$ génère une observation $O$, soit $P(O \mid \lambda)$. Afin de résoudre ce problème, plusieurs méthodes sont possibles. La plus courante est une méthode itérative, l'algorithme Forward [Rab89].

Etant données une observation $O$ et une CMC $\lambda$, un autre problème est de déterminer le chemin d'états $Q$ le plus probablement suivi par la CMC $\lambda$ lorsqu'elle engendre l'observation $O$. L'algorithme de Viterbi [Vit67] permet de réaliser cette recherche. La similitude entre deux observations peut être mesurée en comparant les deux meilleurs chemins d'états associés.

Le troisième problème à résoudre est celui de l'apprentissage. Il peut être réalisé de manière supervisée ou non. Dans le premier cas, on suppose que l'architecture (c'est-à-dire le nombre d'états $N$ et les liens entre états) de la CMC optimale est connue *a priori* ainsi qu'un ensemble d'observations associées au processus. Le résultat attendu est une CMC $\overline{\lambda}$ caractérisée par la plus forte probabilité d'engendrer les observations. $\overline{\lambda}$ constitue un modèle du processus. Ce problème est résolu par l'algorithme de Baum-Welch [Bau67]. Dans le cas non supervisé, le but est de déterminer la CMC $\lambda$ ayant la plus forte probabilité d'engendrer les observations, sans aucune autre information *a priori* que l'ensemble d'observations. Pour résoudre ce problème, nous utiliserons l'algorithme GHOSP (Genetic Hybrid Optimization & Search of Parameters) introduit par Brouard dans sa thèse [Bro99].

### 5.2.3 Extension au cas multidimensionnel

Nous avons décrit précédemment les chaînes de Markov cachées et les problèmes qui leur sont associés. Ces outils statistiques sont particulièrement adaptés pour modéliser des phénomènes physiques 1-D. Dans notre cas, un parcours des pixels de l'image selon une règle prédéfinie est nécessaire pour obtenir une représentation 1-D de l'image. Lorsque chaque pixel est caractérisé par plusieurs valeurs (comme dans le cas des images couleur), la nature multidimensionnelle des données nécessite un traitement particulier. Dans la littérature, chaque symbole est généralement associé à une région dans l'espace multidimensionnel, ce qui nécessite une classification ou une segmentation préalable des données et implique de plus la perte de l'aspect complémentaire fourni par l'ensemble des différentes composantes. Afin d'éviter ces limitations, et d'utiliser des données à plusieurs composantes sans définir de symboles globaux, nous avons fait le choix d'utiliser des CMC multidimensionnelles, pour lesquelles la distribution des symboles est définie pour chaque dimension. Ce type d'approche a été proposé indépendamment par Yang *et al.* [Yan94] et par Brouard [Bro99]. Nous utiliserons ici la modélisation introduite dans [Bro99] sous l'appelation chaînes de Markov cachées multidimensionnelles à processus indépendants.

Dans cette approche, on considère qu'à un instant donné $t$ la CMC génère $R$ symboles différents (avec $R = 3$ dans le cas d'images couleur) et non plus un seul symbole. Ici chacune des composantes peut être de nature différente des autres et aucune normalisation n'est nécessaire. L'observation décrit ici l'évolution de $R$ processus, ce qui signifie que nous disposons de différents aspects d'un même phénomène. Le processus observé étant unique, on fait l'hypothèse que les $R$ processus partagent les mêmes états avec les mêmes transitions, ce qui signifie dans le modèle que le phénomène n'est caractérisé que par une seule matrice $A$. Par contre, les symboles sont spécifiques à chaque processus, ce qui amène à définir plusieurs matrices $B$ (une pour chacun des processus observables simultanément). La définition de la CMC-MD/I est donc plus complète que celle d'une CMC classique. En effet, elle contient les éléments supplémentaires suivants :

- $R$ : le nombre de processus gérés par la CMC-MD/I ;
- $V^r$ : l'ensemble (de cardinalité $M^r$) des symboles associés au processus $P_r$ ;
- $B^r$ : la matrice de distribution des probabilités de génération des symboles associés au processus $P_r$ ;
- $V$ : l'ensemble (de cardinalité $R$) des dictionnaires de symboles $V^r$ associés à chaque processus ;
- $B$ : l'ensemble (de cardinalité $R$) des matrices $B^r$ de distribution des probabilités de génération des symboles associés à chaque processus.

Une modification des algorithmes utilisés pour résoudre les différents problèmes liés aux CMC (évaluation, détermination du chemin d'états, apprentissage supervisé ou non) a donc été proposée dans [Bro99] pour s'adapter aux spécificités des CMC-MD/I.

### 5.2.4 Les CMC et l'analyse d'images

Les chaînes de Markov cachées sont fréquemment utilisées dans le domaine d'analyse de la parole. Leur emploi en analyse d'images est plus rare sauf dans le domaine de la reconnaissance de l'écrit où les codages utilisés permettent l'utilisation de chaînes [EY99]. Aas *et al.* présentent dans [Aas99] différentes applications des chaînes de Markov cachées, telles que la reconnaissance du texte, la vision industrielle, la classification d'images IRM, ou encore la détection de véhicules dans une application de surveillance. Brouard *et al.* [Bro98] proposent une méthode de segmentation d'images basée sur les chaînes de Markov cachées. Caelli *et al.* [Cae01] appliquent les chaînes de Markov cachées au problème de la génération et du suivi de frontières dans des images aériennes. Tao *et al.* [Tao99] utilisent les chaînes de Markov cachées pour modéliser leur problème de suivi d'objets multiples.

Le caractère monodimensionnel des chaînes de Markov cachées amène de nombreux chercheurs à proposer leurs propres modèles Markoviens. Ainsi, Gong [Gon92a] introduit la notion de VAHMM *(Visual Augmented Hidden Markov Models)* afin de prédire les mouvements de véhicules dans un aéroport. Rigoll *et al.* présentent dans [Rig00] les P2HMM *(Pseudo 2-D Hidden Markov Models)* qui, utilisés conjointement avec des filtres de Kalman, permettent de suivre des personnes dans des scènes intérieures ou extérieures. Wilson et Bobick [Wil99] définissent des PHMM *(Parametric Hidden Markov Models)* linéaires et non-linéaires utilisés dans une application de reconnaissance de gestes. Bui *et al.* décrivent dans [Bui01] des AHMM *(Abstract Hidden Markov Models)* pour suivre et prédire les trajectoires de personnes dans des scènes intérieures.

Nous avons pensé pouvoir utiliser de manière originale le concept en analyse d'images. D'une part, nous utilisons les CMC pour modéliser la variation spatiale et non la variation temporelle des informations. D'autre part, les CMC-MD/I, utilisées pour gérer des données multidimensionnelles telles que les images couleur, permettent de modéliser l'évolution d'un processus caractérisé par des symboles spécifiques mais partageant les mêmes états et les mêmes transitions.

Dans la section suivante, nous abordons le problème de l'apprentissage.

## 5.3 Apprentissage par l'algorithme GHOSP

L'efficacité du suivi dépend directement de la qualité de l'apprentissage. Afin d'assurer une bonne qualité d'apprentissage, nous avons choisi de ne pas considérer de contrainte de temps de calcul $\mathcal{C}_{\text{rapidité}}$ pour ce procédé, et l'apprentissage est effectué hors ligne. Cet apprentissage n'est effectué qu'une seule fois par objet. Le modèle de l'objet obtenu peut alors être utilisé dans différentes séquences vidéo pour effectuer l'étape de suivi.

Le but de l'étape d'apprentissage est de fournir une CMC représentant au mieux cet objet. Pour cela, nous faisons appel à l'algorithme GHOSP. Dans un premier temps il sera décrit. Nous détaillerons également les prétraitements effectués sur les images d'apprentissage et qui sont nécessaires pour utiliser l'algorithme GHOSP. Nous décrirons ensuite les différentes étapes du processus d'apprentissage. Finalement, nous verrons à l'aide d'un exemple comment les images d'apprentissage sont traitées et quels sont les paramètres de GHOSP utilisés.

### 5.3.1 L'algorithme GHOSP

L'algorithme GHOSP (Genetic Hybrid Optimization & Search of Parameters) [Bro97] résulte de l'hybridation d'un algorithme génétique [Hol75] et de l'algorithme de Baum-Welch [Bau67] permettant l'optimisation d'une CMC.

Les algorithmes génétiques sont connus pour leurs capacités en apprentissage qui leur permettent notamment de s'adapter à différents environnements et de s'échapper de minima locaux.

L'algorithme de Baum-Welch [Bau67] est un algorithme d'optimisation de type gradient. En particulier, il a une tendance à se laisser piéger par des optima locaux. L'utilisation simultanée des algorithmes génétiques et de l'algorithme de Baum-Welch permet de s'adapter à différents environnements et de s'échapper de minima locaux. On améliore ainsi la CMC qui représente le mieux une observation.

Dans le cas des CMC, le génotype traduit le triplet $\lambda = (A, B, \Pi)$. La qualité d'un individu est liée à la probabilité (évaluée par l'algorithme Forward) que la CMC puisse engendrer l'observation. La figure 5.1 résume comment les deux algorithmes AG et BW interagissent. Dans une première étape, l'utilisation de l'algorithme de Baum-Welch permet une première optimisation de la chaîne. Dans ce cadre, les gènes, les chromosomes, la fonction d'évaluation, et les opérateurs de croisement et de mutation sont définis de la manière suivante :

– les gènes sont définis dans l'espace des probabilités ; un chromosome correspond simplement à la réorganisation sous forme d'un vecteur, des coefficients contenus dans $A$, $B$, et

$\Pi$ ;

- l'évaluation est effectuée par l'algorithme Forward (meilleur est l'individu, plus élevé est le taux de vraisemblance de l'observation) ;
- le croisement consiste en la combinaison de deux individus selon certaines contraintes (puisque les matrices des CMC sont stochastiques), ces contraintes s'appliquent aussi à l'opérateur de mutation ; plus précisément, après croisement ou mutation, chaque individu est normalisé, afin de rendre les matrices $A$, $B$, et $\Pi$ stochastiques ; de plus, lors de la génération aléatoire de la population initiale, le nombre d'états des individus peut varier entre des bornes prédéterminées ; le croisement doit donc permettre d'obtenir un codage cohérent avec la structure d'une CMC.

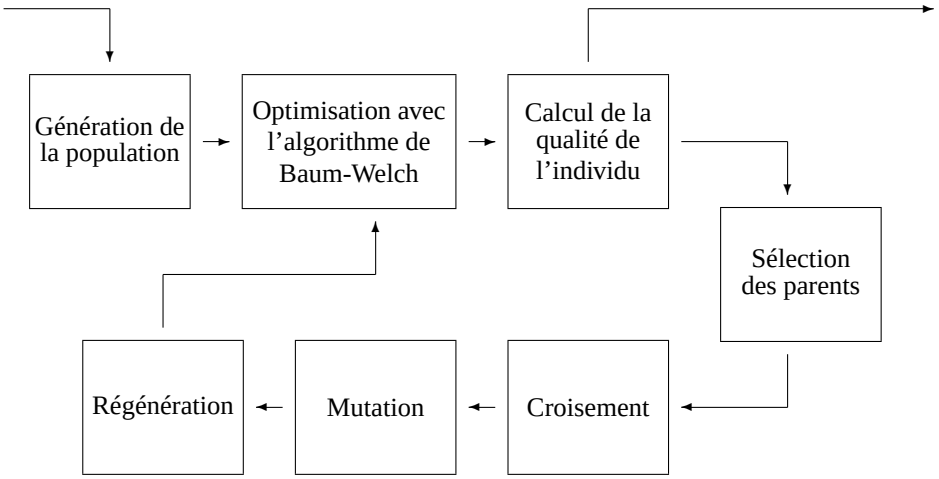


FIG. 5.1 – Hybridation des algorithmes génétiques et de l'algorithme de Baum-Welch : l'algorithme GHOSP.

L'utilisation de l'algorithme génétique conduit à une architecture de CMC. Cette approche peut être vue comme une évolution Lamarckienne où les individus sont capables de modifier leurs gènes durant leur vie (par l'algorithme de Baum-Welch) et de transmettre leurs gènes modifiés à leur descendance [Gre91]. L'algorithme est désigné sous le nom de GHOSP par Brouard [Bro97].

### 5.3.2 Prétraitement des images d'apprentissage

L'image est une représentation particulière de l'objet qui doit être suivi. Pour faciliter l'utilisation de l'algorithme GHOSP, nous avons choisi de n'avoir qu'un faible nombre $M^r$ de symboles dans la CMC pour créer une séquence d'observations associée à l'image et qui sera traitée par les CMC.

Les images utilisées lors de l'étape d'apprentissage sont des images couleur codées dans l'espace couleur RGB. Elles peuvent donc être décomposées en 3 images en niveaux de gris, une pour chaque composante couleur (Rouge, Vert, et Bleu). Chaque pixel des images en niveaux de gris est caractérisé par une valeur entière $n$ appartenant à l'intervalle $[0, N-1]$, avec $N = 256$. En notant $\lfloor x \rfloor$ la partie entière de $x$, l'échantillonnage est effectué de la manière suivante :

$$n' = \left\lfloor n \cdot \frac{M^r - 1}{N} \right\rfloor \tag{5.9}$$

où $n'$, $N$, et $M^r$ correspondent respectivement au symbole obtenu, au nombre de valeurs de $n$ possibles, et au nombre de symboles associés au processus $P_r$.

Après avoir effectué cet échantillonnage, nous disposons d'observations multidimensionnelles. Chaque observation peut se représenter par trois vecteurs (de tailles identiques) ou séquences de points. Chacune des séquences correspond à une composante couleur et chaque point représente donc l'intensité d'un pixel pour une composante couleur donnée. D'autre part une image correspond à un ensemble 2-D de données multidimensionnelles alors qu'une observation est représentée par une séquence 1-D de données multidimensionnelles. Une technique de parcours de l'image doit être utilisée pour obtenir une séquence 1-D à partir d'une image 2-D. Nous avons étudié expérimentalement différents parcours plus ou moins complexes. Il en résulte que l'apport fourni par l'utilisation de parcours complexes tels que celui de Peano, qui a été utilisé dans de nombreux travaux [Ste83, See97, Gio97], est relativement faible.

### 5.3.3 Processus d'apprentissage

A l'issue de ce prétraitement nous disposons d'une observation associée à chaque image et il est possible de générer une CMC-MD/I appropriée. Le résultat obtenu consiste en une matrice $A$, une matrice $\Pi$, et un ensemble de trois matrices $B^r$. Afin de résoudre un problème lié au manque de données dans le corpus d'apprentissage, la présence de valeurs nulles dans les matrices $B^r$, nous utilisons une technique présentée dans [Lev83]. Après addition d'une faible constante $(\varepsilon = 0,1)$ à chacun des termes de ces trois matrices, une normalisation est opérée afin de conserver la propriété de stochasticité de ces matrices.

Evidemment, les données utilisées en entrée de l'algorithme d'apprentissage ne sont pas issues d'une seule image représentant l'objet à suivre. Afin d'obtenir une qualité d'apprentissage aussi élevée que possible, nous utilisons un ensemble d'images contenant l'objet à suivre dans différentes positions et tailles, sous différents points de vue et conditions d'éclairage.

Ainsi, l'algorithme GHOSP peut fournir une CMC-MD/I plus générique qui représentera

plus fidèlement l'objet à suivre. En considérant l'exemple du suivi de ballon dans un match de football, des exemples d'images d'apprentissage utilisées sont donnés dans la figure 5.2.

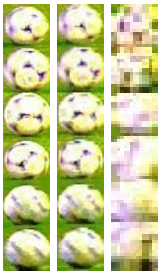

FIG. 5.2 – Images utilisées lors de la phase d'apprentissage. De gauche à droite (à différentes échelles), les images sont composées respectivement de $34 \times 34$, $26 \times 26$, et $15 \times 15$ pixels.

Les paramètres utilisés par l'algorithme GHOSP sont décrits dans le tableau 5.1. Ils permettent d'obtenir un apprentissage automatique d'un objet (ici un ballon de football) à partir des données d'apprentissage. Comme nous l'avons souligné, la dimension est liée à l'utilisation d'images couleur. La simplicité des images conduit à un nombre relativement faible d'états. Le nombre de symboles est le même pour chaque composante couleur, chacune d'entre elles étant considérée avec la même importance et analysée de la même manière. La taille de la population représente le nombre de solutions explorées par l'AG à chaque itération, tandis que le nombre de parents correspond au nombre de meilleures solutions utilisées comme point de départ pour l'exploration suivante. La méthode est relativement robuste à un changement de la valeur de ces paramètres.

| Paramètre | Valeur |
|---|---|
| Nombre d'états minimum | 4 |
| Nombre d'états maximum | 6 |
| Nombre de symboles | 11 |
| Nombre d'itérations de l'algorithme de Baum-Welch | 3 |
| Taille de la population dans l'algorithme génétique | 100 |
| Nombre de parents dans la population | 90 |
| Nombre d'itérations de l'algorithme génétique | 80 |
| Nombre de dimensions | 3 |

TAB. 5.1 – Paramètres de l'algorithme d'apprentissage GHOSP.

Après avoir étudié la phase d'apprentissage, deux autres problèmes peuvent être abordés : l'évaluation de l'adéquation du modèle à l'observation et à la détermination du meilleur chemin d'états. Ces deux problèmes peuvent être résolus respectivement par l'algorithme Forward et l'algorithme de Viterbi. Nous proposons dans ce chapitre de modéliser le problème du suivi

d'objet se basant sur chacun de ces deux aspects. Dans les deux sections suivantes, nous présentons donc deux méthodes de suivi d'objet basées d'une part sur l'algorithme Forward et d'autre part sur l'algorithme de Viterbi.

## 5.4 Suivi basé sur l'algorithme Forward

Le suivi d'objet rigide comporte ici deux étapes : la prédiction d'une position approximative à l'aide d'un estimateur de mouvement simple, puis la recherche de la position exacte de l'objet en utilisant la CMC-MD/I générée lors de l'étape d'apprentissage.

On peut considérer la probabilité avec laquelle une CMC $\lambda$ génère une observation $O$, soit $P(O \mid \lambda)$. Ce problème peut être résolu à l'aide de l'algorithme Forward [Rab89]. Dans cette section, nous développons cette approche.

La méthode proposée consiste en trois étapes réalisées de manière itérative pour chaque image. Il est tout d'abord nécessaire de déterminer si l'objet a été perdu ou non. Une position approximative de l'objet est ensuite prédite. Finalement, la recherche de la position exacte de l'objet est effectuée.

### 5.4.1 Détermination de l'état de l'objet suivi

La première étape de l'algorithme consiste à vérifier si à l'instant $t$ l'objet a été perdu ou non. Pour cela, nous analysons les résultats de l'algorithme Forward pour une image $I_t$ en fonction des résultats obtenus à l'image $I_{t-1}$.

La zone analysée dans la trame $I_t$ est découpée en plusieurs fenêtres $f_i(I_t)$ de taille $X_{\text{fenêtre}} \times Y_{\text{fenêtre}}$. L'algorithme Forward [Rab89] est appliqué sur chaque fenêtre $f_i(I_t)$ en considérant la CMC-MD/I générée lors de l'apprentissage. Il fournit un score global $P_i(I_t)$. On définit alors un indice $p_i(I_t)$ que le contenu de la fenêtre $f_i(I_t)$ soit généré par la CMC. Si on considérait que les indices $p_i(I_t)$ étaient identiques, on obtiendrait :

$$p_i(I_t) \approx e^{\frac{\ln(P_i(I_t))}{R \times X_{\text{fenêtre}} \times Y_{\text{fenêtre}}}} \tag{5.10}$$

où $R$ est la dimension de la CMC-MD/I. L'indice $p_i(I_t)$ peut être considéré comme une normalisation par rapport à la taille et la dimension de l'observation. Plus la valeur de $p_i(I_t)$ est élevée, mieux la fenêtre $f_i(I_t)$ représente l'objet et plus elle a de chances de contenir l'objet suivi.

Dans une image, nous cherchons donc la fenêtre $f_{\max}(I_t)$ de plus fort indice $p_{\max}(I_t)$ :

$$p_{\max}(I_t) = \max_i p_i(I_t) \tag{5.11}$$

Idéalement, la valeur $p_{\max}(I_t)$ devrait être proche de 1 et tous les indices $p_i(I_t)$ éloignés de cette valeur seraient associés à des fenêtres ne contenant pas l'objet à suivre. Cependant la représentation de l'objet par une image numérique s'accompagne d'erreurs liées à la qualité de l'acquisition (bruit, sensibilité au mouvement, *etc.*). Le processus doit donc considérer une certaine tolérance à ces erreurs. Ainsi, dans une image donnée $I_t$, l'indice de référence (noté $p_{\text{ref}}(I_t)$) ne sera pas considéré comme égal à 1, mais comme égal à l'indice associé à la fenêtre contenant l'objet dans la trame précédente, soit $p_{\max}(I_{t-1})$.

A partir de cette mesure, il est possible de définir un intervalle $[s_{\text{inf}}(I_t), s_{\text{sup}}(I_t)]$ en dehors duquel on considère que les indices n'assurent pas la présence de l'objet suivi :

$$s_{\text{inf}}(I_t) = \alpha_{\text{inf}} \times p_{\max}(I_{t-1}) \quad ; \quad s_{\text{sup}}(I_t) = \alpha_{\text{sup}} \times p_{\max}(I_{t-1}) \tag{5.12}$$

où $\alpha_{\text{inf}}$ et $\alpha_{\text{sup}}$ sont deux coefficients permettant de qualifier la tolérance aux erreurs. Ces valeurs sont actuellement définies de manière empirique mais elles pourraient être obtenues par apprentissage hors ligne ou analyse de l'erreur en ligne. Cet intervalle sera utilisé pour déterminer si un objet a été perdu.

Le traitement de la première image $I_1$ diffère du traitement des autres images puisqu'il est impossible de se baser sur les informations contenues dans les images précédentes pour déterminer l'intervalle $[s_{\text{inf}}(I_t), s_{\text{sup}}(I_t)]$. Dans ce cas, cinq fenêtres sont créées autour de la position initiale de l'objet, comme le montre la figure 5.3.

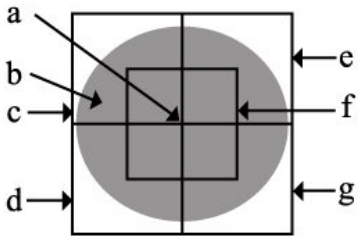


FIG. 5.3 – Fenêtres utilisées pour déterminer $p_{\max}(I_1)$ sur la première image $I_1$. La position *a* de l'objet *b* est considérée comme connue. Les 5 fenêtres sont représentées par les symboles *c* à *g*.

Actuellement, la taille de ces fenêtres est constante tout au long de la séquence d'images. On

peut remarquer qu'une des fenêtres recouvre partiellement chacune des 4 autres. Pour chaque fenêtre $f_i(I_1)$, nous calculons l'indice associé $p_i(I_1)$. Nous sélectionnons ensuite l'indice maximal $p_{\max}(I_1)$ afin de définir l'intervalle $[s_{\inf}(I_2), s_{\sup}(I_2)]$.

### 5.4.2 Prédiction d'une position approximative

Le positionnement des fenêtres repose sur une étape de prédiction de position. Cette étape est nécessaire pour éviter la perte d'un objet en déplacement ou une recherche trop coûteuse (car initialisée depuis une position erronée). Nous utilisons ici une méthode basée sur un modèle simple de mouvement afin de limiter le temps de calcul. De plus, le résultat obtenu n'est pas considéré comme définitif mais est utilisé comme initialisation pour le processus de recherche ultérieur.

Le modèle utilisé est donc basé sur les positions de l'objet dans les deux fenêtres précédentes :

$$\begin{cases} x_t = x_{t-1} + c\,(x_{t-1} - x_{t-2}) \\ y_t = y_{t-1} + c\,(y_{t-1} - y_{t-2}) \end{cases} \tag{5.13}$$

La position $(x_t, y_t)$ est alors utilisée lors de l'étape de recherche de la position décrite plus loin. L'hypothèse de vitesse constante se traduit par $c = 1$. Dans le cas où l'objet n'aurait pas été trouvé (notamment si sa vitesse n'est pas constante), plusieurs cas sont envisagés : soit une décélération ($c = \frac{1}{2}$), soit une accéleration ($c = 2$). Finalement, il est aussi possible que l'objet se soit arrêté. Ce mouvement nul entre les deux dernières images se traduit par $c = 0$.

Si l'objet est perdu dans tous les cas, on suppose qu'un phénomène d'occlusion temporaire est en cours et sa position estimée reste inchangée. Si le phénomène d'occlusion persiste sur un certain nombre d'images, le processus de suivi est arrêté et il est nécessaire d'effectuer à nouveau une détection des objets.

### 5.4.3 Recherche de la position exacte

En utilisant un de ces modèles de mouvement, nous obtenons une prédiction de la position approximative de l'objet à partir des images obtenues aux instants $t-1$ et $t-2$. La recherche de la position exacte de l'objet est limitée à une fenêtre locale dont le centre est situé à la position $(x_t, y_t)$.

Nous souhaitons utiliser l'algorithme Forward. Cet algorithme fournit un score qui permet de déterminer si l'objet est représenté ou non par l'observation analysée, mais ne permet pas

de localiser précisément l'objet dans une fenêtre. Nous avons développé une technique en deux étapes où l'algorithme Forward est appliqué sur des observations associées à chacune des sous-fenêtres considérées. Nous utilisons la notation $f_i(I_t)$ introduite précédemment pour représenter les différentes sous-fenêtres analysées dans la trame $I_t$ de la séquence vidéo.

Plus précisément, la fenêtre de recherche est divisée en 64 sous-fenêtres de dimension $X_{\text{fenêtre}} \times Y_{\text{fenêtre}}$ pixels. Ensuite chaque sous-fenêtre est transformée en observation multidimensionnelle composée de trois composantes contenant un nombre $X_{\text{fenêtre}} \times Y_{\text{fenêtre}}$ de valeurs. On obtient donc 64 chaînes d'observations.

La deuxième étape consiste à utiliser les observations pour déterminer la position exacte de l'objet. Pour cela on applique à chaque chaîne d'observations l'algorithme Forward dans sa version multidimensionnelle. Dans le but de s'assurer de l'indépendance des probabilités à la taille et à la dimension des observations, on utilise les valeurs $p_i(I_t)$ définies par l'équation (5.10) page 88. Cela permet notamment de gérer le cas où la taille des fenêtres d'analyse $f_i(I_t)$ et celle des images du corpus d'apprentissage sont différentes. Parmi les fenêtres caractérisées par une valeur $p_i(I_t)$ comprise dans l'intervalle $[s_{\text{inf}}, s_{\text{sup}}]$, on ne conserve que celles associées aux quatre valeurs $p_i(I_t)$ les plus élevées. Si aucune valeur $p_i(I_t)$ n'appartient à l'intervalle défini par $[s_{\text{inf}}, s_{\text{sup}}]$, on considère que l'objet recherché n'a pas été trouvé et la recherche de la position exacte est initialisée à partir d'une autre estimation de la position. Si une seule valeur $p_i(I_t)$ appartient à l'intervalle $[s_{\text{inf}}, s_{\text{sup}}]$, la position exacte de l'objet est déterminée comme le centre de la fenêtre pour laquelle cette valeur $p_i(I_t)$ a été obtenue. Enfin, si plusieurs valeurs $p_i(I_t)$ sont comprises entre les deux seuils, la position $C$ de l'objet est calculée comme une combinaison linéaire des centres des fenêtres pour lesquelles les valeurs $p_i(I_t)$ ont été obtenues :

$$C(I_t) = \frac{\sum_i p_i(I_t) c_i(I_t)}{\sum_i p_i(I_t)} \tag{5.14}$$

où $c_i(I_t)$ représente le centre de la fenêtre $f_i(I_t)$ associée à l'indice $p_i(I_t)$.

### 5.4.4 Résultats

La méthode présentée dans cette section a été testée sur des séquences d'images issues de retransmissions télévisées de matchs de football. Le but est de suivre le ballon sur toute la durée d'un plan. La plupart des tests ont été réalisés sur des séquences contenant plus d'une centaine d'images, comme le montre la figure 5.4. Les temps de calcul nécessaires sont de l'ordre de 80 millisecondes par image en considérant une architecture PC Pentium 4 candencé à 1700 MHz.

Même si la méthode utilise un modèle simple de mouvement à vitesse constante, les objets se déplaçant selon un mouvement non linéaire peuvent être suivis de manière précise, comme le montre la figure 5.5.

L'occlusion est un des problèmes les plus importants dans le domaine du suivi d'objet. Notre méthode est capable de suivre des objets temporairement cachés. Un exemple d'occlusion est présenté en figure 5.6.

En cas de présence dans la scène d'objets similaires à l'objet suivi, la méthode peut manquer de robustesse. Dans la figure 5.7, le ballon et la chaussette du joueur sont relativement similaires, et la position obtenue est donc incorrecte. Une étape de vérification basée sur des méthodes morphologiques serait nécessaire pour résoudre ce problème.

La méthode est également incapable de réaliser le suivi correctement lorsque la taille de l'objet à suivre est trop faible. Dans ce cas, des méthodes adaptées doivent être utilisées. Elles feront l'objet du chapitre 6.

Dans cette section, nous avons présenté une méthode de suivi par évaluation d'une CMC à l'aide de l'algorithme Forward. Au vu des résultats obtenus et des limitations de la méthode présentée dans cette section, l'utilisation d'une méthode plus complexe semble justifiée. Nous avons tenté de modéliser le suivi d'objet comme un problème de recherche du meilleur chemin d'états en utilisant l'algorithme de Viterbi.

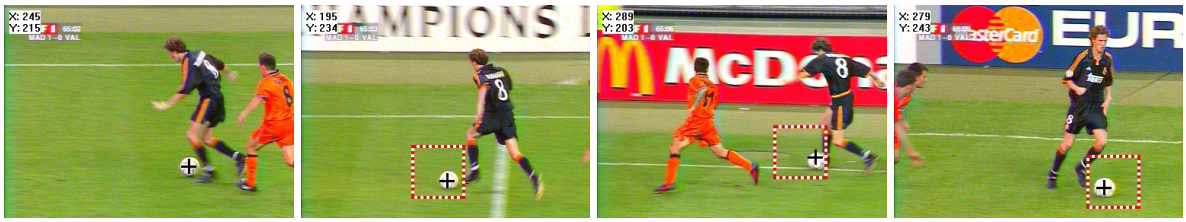

FIG. 5.4 – Suivi d'un objet dans une séquence contenant plus de 100 images. La première image contient la position initiale de l'objet. Les autres images sont extraites à intervalles réguliers de la séquence vidéo.

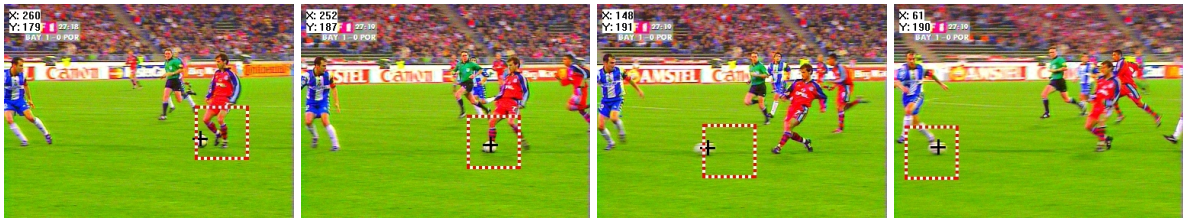

FIG. 5.5 – Suivi d'un objet avec un mouvement non linéaire.

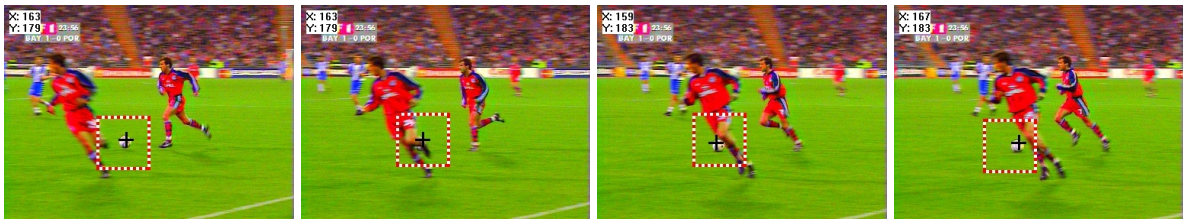

FIG. 5.6 – Suivi d'un objet en présence de phénomènes d'occlusion.

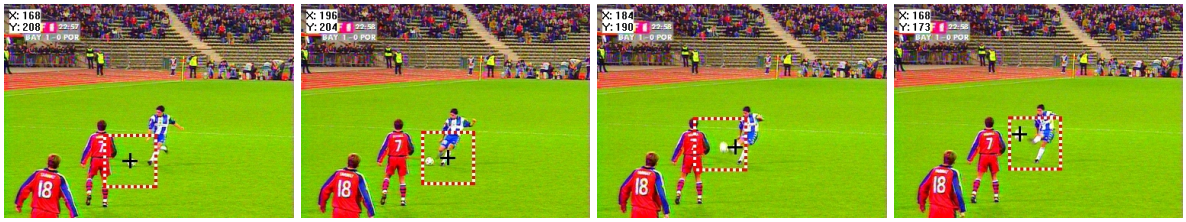

FIG. 5.7 – Suivi incorrect d'un objet dans une scène contenant des objets similaires.

## 5.5 Suivi basé sur l'algorithme de Viterbi

Outre le problème de l'évaluation, un problème classique lié aux chaînes de Markov cachées est la détermination du meilleur chemin d'états. Etant données une observation $O$ et une CMC $\lambda$, on peut déterminer le chemin d'états $Q$ le plus probablement suivi par la CMC $\lambda$ lorsqu'elle engendre l'observation $O$. Cette recherche peut être réalisée en utilisant l'algorithme de Viterbi [Vit67]. Nous proposons dans cette section de modéliser la recherche locale de la position exacte de l'objet par ce problème de recherche du chemin d'états.

Cette méthode nécessite un traitement supplémentaire lors de l'apprentissage. Pour le suivi, elle est composée, de même que la méthode précédente, de trois étapes réalisées de manière itérative pour chaque image. Après avoir obtenu une estimation de la position de l'objet, l'algorithme de Viterbi est utilisé pour calculer des chemins d'états. La dernière étape consiste en l'analyse des chemins d'états obtenus pour déterminer la position exacte de l'objet. Finalement, quelques résultats illustreront l'approche proposée.

### 5.5.1 Traitement supplémentaire lors de l'apprentissage

L'apprentissage décrit dans la section 5.3 a conduit à fabriquer une CMC-MD/I. Ici nous souhaitons modéliser un objet par la CMC-MD/I mais également par la suite des chemins d'états associés à l'observation.

Pour cela, nous appliquons l'algorithme de Viterbi sur les observations utilisées pour l'apprentissage en considérant la CMC-MD/I générée par l'algorithme GHOSP. L'algorithme de Viterbi nous fournit des chemins d'états $Q(I_{\text{app}})$ de référence pour chaque image $I_{\text{app}}$ du corpus d'apprentissage. Ces chemins d'états sont considérés comme des modèles de l'objet appris et seront utilisés lors de l'étape de suivi.

### 5.5.2 Prédiction d'une position approximative

Afin d'initialiser la recherche de l'objet suivi dans une zone particulière de l'image, nous calculons comme précédemment une estimation de la position de l'objet. La fenêtre de recherche est alors centrée sur la position prédite de l'objet. Cependant, elle ne sera pas divisée ici en sous-fenêtres.

### 5.5.3 Calcul du chemin d'états

Afin de déterminer la position exacte de l'objet dans la fenêtre de recherche, il est nécessaire d'obtenir le chemin d'états pour cette fenêtre. La partie de l'image analysée est transformée en observation contenant trois composantes. L'algorithme de Viterbi est ensuite appliqué sur l'observation afin d'obtenir le chemin d'états $Q$ représentant la fenêtre de recherche.

### 5.5.4 Analyse du chemin d'états

Il est ensuite possible de chercher les occurrences de chaque chemin d'états $Q(I_{\text{app}})$, défini comme un modèle de l'objet, dans ce chemin d'états $Q$ généré par l'algorithme de Viterbi. Ces occurrences vont être utilisées pour déterminer la position de l'objet suivi.

Les chemins d'états $Q(I_{\text{app}})$ et $Q$ doivent être comparés. On définit une zone $Q'$ de la même taille que les modèles $Q(I_{\text{app}})$ et on décale cette zone le long du chemin d'états $Q$ obtenu à partir de la fenêtre de recherche, afin de couvrir toutes les positions possibles.

Il est alors possible de comparer les chemins d'états $Q'$ et $Q(I_{\text{app}})$ en calculant une distance d'édition entre les deux séquences d'états. Nous avons choisi d'utiliser l'algorithme de Wagner et Fischer [Wag74]. Les coûts de substitution, d'insertion, et de suppression ont été fixés à 0 ou 1. On obtient une mesure de dissimilarité entre deux chaînes, deux chemins d'états dans notre cas. Cet algorithme permet de comparer deux chaînes de longueur différente. Nous aurions donc pu l'utiliser pour comparer le chemin d'états $Q$ avec chaque chemin d'états de référence $Q(I_{\text{app}})$.

Cependant le résultat obtenu aurait consisté en une mesure de distance ne nous permettant pas de localiser précisément l'objet suivi. C'est pourquoi nous comparons $Q'$ et $Q(I_{\text{app}})$, et si la mesure obtenue est inférieure à un seuil proportionnel à la longueur de $Q(I_{\text{app}})$, une occurrence possible de $Q(I_{\text{app}})$ est trouvée dans $Q$.

Si le nombre d'occurrences possibles de chemin d'états $Q(I_{\text{app}})$ (considérant toutes les images $I_{\text{app}}$ du corpus d'apprentissage) est inférieur à un seuil $S_{\text{app}}$, on considère le nombre d'états représentant l'objet dans l'image comme insuffisant. Le seuil $S_{\text{app}}$ permet de quantifier la tolérance que l'on souhaite imposer au processus, et de déterminer le compromis désiré entre rappel et précision. Si le nombre d'occurrences est supérieur au seuil $S_{\text{app}}$, on considère que l'objet a été trouvé dans la zone de recherche. Une analyse spatiale est alors effectuée afin d'obtenir la position exacte de l'objet. Cette position est déterminée comme le barycentre des positions des occurrences du modèle de l'objet dans le chemin d'états associé à l'image analysée. La position spatiale de l'objet suivi est finalement obtenue en effectuant la transformation depuis l'espace 1-D du chemin d'états vers l'espace 2-D de la zone de recherche.

### 5.5.5 Résultats

La méthode présentée dans cette section a été testée sur les mêmes séquences d'images que la méthode décrite dans la section précédente. Il est également possible de suivre le ballon sur une séquence contenant plus de 100 images comme le montre la figure 5.8.

La méthode est capable de gérer les phénomènes d'occlusion temporaire. Un exemple de suivi dans ces circonstances est donné en figure 5.9.

Les limites de la méthode sont relativement similaires à celles de la méthode présentée dans la section précédente. Ainsi, si plusieurs objets aux caractéristiques similaires sont présents dans la scène, le suivi peut échouer (figure 5.10).

Comme le montrent les différents résultats illustrés ici, les deux approches présentées dans

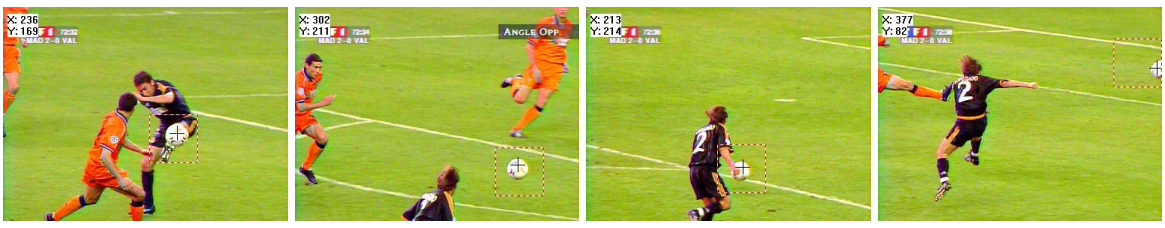

FIG. 5.8 – Suivi d'un objet dans une séquence contenant plus de 100 images.

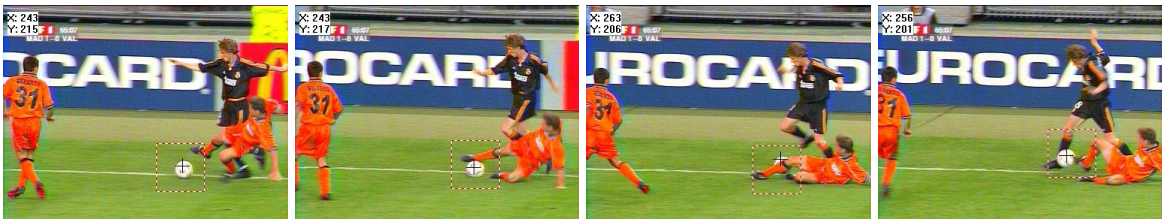

FIG. 5.9 – Suivi d'un objet en présence de phénomènes d'occlusion.

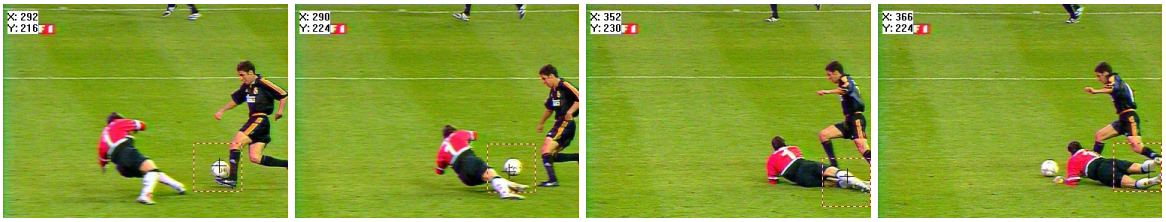

FIG. 5.10 – Suivi incorrect d'un objet dans une scène contenant des objets aux caractéristiques similaires.

ce chapitre fournissent des résultats relativement similaires. Elles sont toutes deux capables de suivre des objets dans des séquences vidéo composées de plus de 100 trames mais échouent lorsque des objets aux caractéristiques similaires sont présents dans la scène. Contrairement à nos intuitions, la seconde méthode (basée sur l'algorithme de Viterbi), quoique plus complexe, ne permet pas d'améliorer les résultats obtenus. Par contre, elle est caractérisée par des temps de calcul plus élevés.

## 5.6 Détection de l'objet appris

Dans les deux sections précédentes, nous avons utilisé le résultat de l'apprentissage pour réaliser un suivi de l'objet appris dans une séquence vidéo. Ce suivi suppose que la position de l'objet soit connue dans la première image. Pour cela, nous proposons une étape de détection de l'objet à suivre, qui ne sera effectuée que sur la première image de chaque plan.

Nous présentons ici encore deux méthodes, basées respectivement sur les algorithmes Forward et de Viterbi. Ces deux méthodes peuvent être assimilées à des adaptations à l'analyse de l'image complète, des méthodes de suivi précédentes.

### 5.6.1 Détection basée sur l'algorithme Forward

La méthode de suivi par l'algorithme Forward est basée principalement sur le découpage de l'image en sous-fenêtres $f_i$ de taille $X_{\text{fenêtre}} \times Y_{\text{fenêtre}}$, puis sur l'application de l'algorithme Forward sur chaque sous-fenêtre $f_i$. La méthode peut s'appliquer sur l'image entière. Le nombre de fenêtres de taille $X_{\text{fenêtre}} \times Y_{\text{fenêtre}}$ créées est alors plus élevé.

Pour chaque fenêtre créée, l'algorithme Forward est appliqué à l'observation associée afin de fournir une probabilité $P(O \mid \lambda)$. On ne s'intéresse qu'aux $k$ plus élevées probabilités fournies par l'algorithme Forward. La valeur de $k$ dépend du rapport entre la taille des images du corpus d'apprentissage et la taille des fenêtres d'analyse. Plus ce rapport est élevé, plus le nombre de fenêtres pouvant contenir l'objet sera important. Le paramètre $k$ sera alors ajusté en conséquence. Si on considère des sous-fenêtres dont la taille est proche de la taille des images du corpus d'apprentissage, on peut affirmer qu'au maximum 4 fenêtres peuvent être associées à un objet à détecter et on choisit $k = 4$. Dans ce cas, les 4 probabilités associées seront relativement faibles. Evidemment, il est possible qu'une unique fenêtre contienne l'objet à suivre. La probabilité associée à cette fenêtre sera alors élevée.

Cependant, le choix d'une valeur $k$ si faible se traduit par une sensibilité importante au bruit. Nous choisissons donc une valeur plus élevée ($k = 20$ déterminé empiriquement) afin de garantir que l'objet est représenté par certaines des fenêtres $f_i$ considérées. Il nous faut maintenant analyser ces fenêtres pour déterminer précisément la position de l'objet. En partant de connaissances *a priori* sur la taille de l'objet, nous éliminons les ensembles connexes de 5 fenêtres ou plus parmi les fenêtres conservées, considérant que la taille de l'objet détecté est trop élevée. Cette approche permet de ne considérer que des objets dont la taille est relativement similaire à celle de l'objet appris. Elle est particulièrement efficace pour supprimer les grands ensembles connexes de fenêtres. Cependant elle n'est pas aussi robuste dans le cas d'ensembles connexes contenant un faible nombre de fenêtres.

Nous avons donc envisagé une seconde technique, plus robuste, qui effectue une analyse locale des fenêtres associées aux $k$ probabilités les plus élevées. Un processus de vérification est effectué itérativement, depuis la probabilité la plus élevée jusqu'à la $k^{\text{ième}}$ probabilité. Ce processus s'arrête lorsqu'une fenêtre (ou observation) $O$ associée à la probabilité concernée vérifie la condition suivante. Dans le voisinage $3 \times 3$ de la fenêtre $O$, on note $O_v$ celle qui est caractérisée par la probabilité la plus élevée. Puisque l'objet ne peut occuper qu'un ensemble spatial de $2 \times 2$ fenêtres, nous analysons le symétrique de $O$ par rapport à $O_v$. Si cette fenêtre est associée à une probabilité appartenant aux $k$ plus élevées, alors la taille de l'objet détecté ne correspond pas à celle de l'objet appris.

La méthode de détection présentée ici a été testée sur différentes images. Malgré l'analyse de toute l'image, le temps de calcul reste relativement faible. Cependant, la qualité de la détection dépend fortement de la nature de l'environnement, comme le montre la figure 5.11.

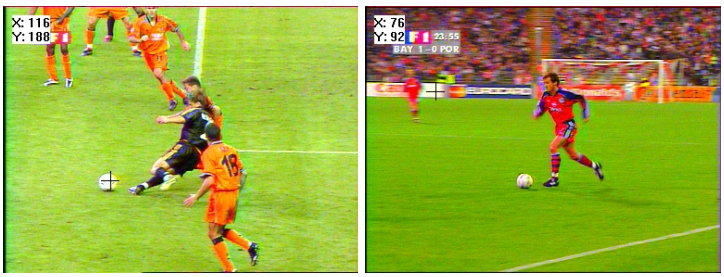


FIG. 5.11 – Détection d'un objet par l'algorithme Forward : correcte au sein d'un environnement simple (à gauche), incorrecte au sein d'un environnement complexe (à droite).

Nous avons donc envisagé de représenter la détection d'un objet appris par un problème plus complexe, celui de la détermination du meilleur chemin d'états, résolu par l'algorithme de Viterbi.

### 5.6.2 Détection basée sur l'algorithme de Viterbi

La méthode est une adaptation de la méthode de suivi à la recherche dans une image entière. Nous utiliserons ici les chemins d'états $Q(I_{\text{app}})$ de référence. Ces chemins d'états ne sont calculés qu'une seule fois, hors ligne. Nous découpons l'image en différentes fenêtres (ou observations) et sur chacune d'entre elles nous appliquons l'algorithme de Viterbi avec la CMC-MD/I générée lors de l'apprentissage.

Le chemin d'états $Q$ associé à chaque fenêtre est alors comparé avec les chemins d'états $Q(I_{\text{app}})$ de référence. Contrairement à la section 5.5, les chemins d'états $Q$ et $Q(I_{\text{app}})$ sont ici de même longueur. La comparaison, effectuée à l'aide de l'algorithme de Wagner et Fischer, se traduit par un coût de passage d'un chemin d'états à l'autre. Pour chaque chemin d'états $Q$, le nombre de distances calculées est égal au nombre d'images $I_{\text{app}}$ présentes dans le corpus d'apprentissage. Nous ne conservons alors pour chaque chemin d'états $Q$ que celui de coût minimal.

La recherche de la position de l'objet s'effectue comme dans le cas précédent, mais ici les probabilités (décroissantes) sont remplacées par les mesures de distance (croissantes).

De même que dans le cas du suivi, les résultats obtenus avec cette méthode (figure 5.12) sont relativement similaires à ceux obtenus avec la méthode basée sur l'algorithme Forward.

Cependant le temps de calcul est ici plus important puisque chaque zone est comparée à tous les modèles de l'objet appris, c'est-à-dire les chemins d'états de référence.

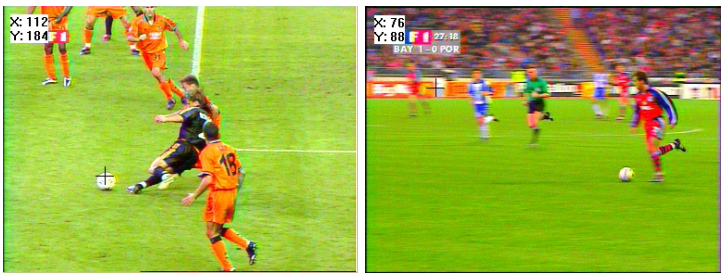


FIG. 5.12 – Détection d'un objet par l'algorithme de Viterbi : correcte au sein d'un environnement simple (à gauche), incorrecte au sein d'un environnement complexe (à droite).

## 5.7 Comparaison avec un algorithme de *template matching*

Dans ce chapitre, nous avons proposé une nouvelle manière d'utiliser les chaînes de Markov cachées pour le suivi d'objet. En effet, les CMC ne sont pas ici utilisées pour modéliser le mouvement des objets au cours du temps mais plutôt la variation des pixels des objets au sein de l'image. Les résultats obtenus avec ce type d'approche sont encourageants. Cependant il est nécessaire de comparer ces résultats avec ceux obtenus avec des méthodes de suivi traditionnelles, afin de vérifier de manière pratique l'apport de notre contribution.

Afin d'évaluer l'intérêt de l'approche proposée dans ce chapitre, nous avons choisi de la comparer avec deux techniques différentes (*template matching* et *template updating*) reposant sur des principes similaires. Le choix de ce type de technique s'explique par le fait que l'utilisation des chaînes de Markov cachées proposée ici peut être considérée comme une méthode pour chercher des motifs dans les images d'une séquence vidéo à partir d'un modèle appris au préalable. Cette approche est donc en quelque sorte une méthode de type *template matching*.

Le *template matching* est une technique fréquemment utilisée dans le domaine du suivi d'objet. Elle consiste en l'appariement d'une forme apprise au préalable (ou modèle, *template*, de l'objet) avec une zone de l'image analysée. Le but est donc de trouver dans chaque image de la séquence vidéo les zones qui correspondent le mieux au modèle considéré dans le processus d'apprentissage. La correspondance entre une zone de l'image analysée et l'image apprise peut être évaluée en utilisant diverses mesures de similarité. La plus simple est définie entre deux

images $I_{t_1}$ et $I_{t_2}$ par :

$$d(I_{t_1}, I_{t_2}) = 1 - \frac{1}{255 \times X \times Y \times C} \sum_{x=1}^{X} \sum_{y=1}^{Y} \sum_{c=1}^{C} |I_{t_1}(x, y, c) - I_{t_2}(x, y, c)| \tag{5.15}$$

Afin de garantir une certaine tolérance au bruit, nous considérons de plus une comparaison avec un seuil $S_{\text{min}}$. On peut également prendre en compte l'information temporelle, puisque le mouvement de l'objet fournit une indication *a priori* sur la position présumée de l'objet. La recherche dans l'image courante est donc limitée dans un voisinage rectangulaire (de taille $\Delta_x \times \Delta_y$) localisé autour de la position prédite de l'objet suivi. Cette prédiction est basée sur la position de l'objet dans les précédentes images, en considérant différents modèles de mouvement possibles.

Les paramètres utilisés sont donnés dans le tableau 5.2 où $X_{\text{app}}$ et $Y_{\text{app}}$ représentent respectivement la largeur et la hauteur de l'image d'apprentissage $I_{\text{app}}$.

| Paramètre | Description | Valeur |
|---|---|---|
| $\Delta_x$ | Largeur de la zone de recherche | $3 \times X_{\text{app}}$ |
| $\Delta_y$ | Hauteur de la zone de recherche | $3 \times Y_{\text{app}}$ |
| $S_{\text{min}}$ | Similarité minimale pour que la correspondance soit validée | $0.7$ |

TAB. 5.2 – Paramètres utilisés lors du suivi d'objet par template matching.

Afin de disposer d'une méthode plus robuste aux évolutions temporelles de l'objet suivi, nous avons également comparé notre approche avec une technique de *template updating*. Cette dernière technique diffère du *template matching* par le fait que le modèle $I_{\text{app}}$ de l'objet suivi est modifié au cours de la séquence en fonction d'un critère donné. Parmi les critères régulièrement utilisés, on peut citer un nombre d'images donné depuis la dernière modification du modèle, ou encore une mesure d'erreur supérieure à un seuil dans le calcul de la similarité entre le modèle et l'objet suivi [Kan02a].

Les résultats obtenus sont relativement similaires, quelle que soit la méthode utilisée. En particulier, les approches par chaînes de Markov cachées présentées précédemment ne permettent pas d'obtenir des résultats de meilleure qualité. Les limites des différentes approches sont similaires, et liées notamment à la présence dans la scène d'objets d'aspect proche de l'objet suivi. Les temps de calcul sont, quant à eux, du même ordre de grandeur pour les différentes approches. L'approche proposée dans ce chapitre, quoique théoriquement originale, ne permet donc pas dans la pratique un apport, comparativement aux techniques traditionnelles.

## 5.8 Conclusion

Dans ce chapitre, nous cherchions à résoudre le problème du suivi d'objet rigide après apprentissage. Pour cela, nous avons proposé d'utiliser les chaînes de Markov cachées, qui sont des outils appropriés pour apprendre et reconnaître un motif (ici l'objet à suivre). Mais contrairement aux approches classiques, nous utilisons ici les CMC pour modéliser les variations spatiales des intensités des pixels dans une image donnée, et non pour modéliser les variations temporelles (c'est-à-dire le mouvement) des objets au cours de la séquence vidéo. De plus, afin de gérer des images couleurs, contenant des données multidimensionnelles, nous utilisons une extension des CMC au cas multidimensionnel, appelée Chaîne de Markov Cachée Multidimensionnelle à Processus Indépendants.

Lorsque les éléments à suivre ne peuvent faire l'objet d'un apprentissage, nous devons nous tourner vers d'autres méthodes plus adaptées. Nous détaillons celles-ci dans le prochain chapitre.

# Chapitre 6

# Suivi d'objet rigide sans apprentissage

Dans le chapitre précédent, nous avons traité le problème du suivi d'objet lorsqu'un apprentissage peut être effectué au préalable. Cependant il n'est pas toujours possible d'effectuer un apprentissage de l'objet à suivre. En effet, lorsque l'on ne dispose d'aucune information *a priori* sur l'objet à suivre, ou lorsque la quantité d'informations disponible est trop faible, l'apprentissage ne pourra être réalisé. Dans ce chapitre, nous proposons d'aborder ce type de problème en nous restreignant au cas des objets de petite taille.

Après avoir présenté la problématique du suivi d'objet de petite taille et dressé un bref panorama de la littérature dans ce domaine, nous traiterons le problème de la détection de ces objets dans des images. Il pourra être résolu par différentes approches basées sur une analyse globale ou locale de l'image. Nous décrirons ensuite l'aspect temporel, c'est-à-dire la méthode de suivi. Finalement nous présenterons quelques résultats de suivi d'un ballon de football dans des plans larges ou éloignés.

## 6.1 Problématique

Le problème du suivi de ce type d'objet présente quelques particularités par rapport au problème général du suivi d'objet. Nous détaillerons ici les différentes particularités dont on doit tenir compte.

Tout d'abord, il nous semble nécessaire de donner une définition plus formelle à la qualification subjective "petite taille". Nous considérons qu'un objet est de petite taille si sa taille (c'est-à-dire le nombre de pixels le représentant dans l'image) est trop faible pour obtenir des mesures statistiques fiables. Dans la pratique, un objet de petite taille sera représenté dans une

image par quelques pixels seulement (de 10 à 100 environ). Dans chaque image, nous ne disposerons donc que de peu d'informations sur l'objet à suivre.

Le faible nombre de pixels représentant l'objet dans l'image peut être dû soit à la distance importante qui le sépare de la source d'acquisition (c'est-à-dire la caméra), soit à la faible résolution des images fournies par le système d'acquisition. Dans les deux cas, la représentation de l'objet dans les images ne contiendra que peu (voire pas) de détails sur celui-ci. L'objet sera plutôt caractérisé par un ensemble de pixels de couleur relativement uniforme. Cette cohérence n'apparaît que dans le domaine spatial. En effet, la couleur de l'objet pourra varier d'une image à l'autre et il ne sera donc pas possible de suivre un objet en se basant sur la constance de sa couleur tout au long de la séquence.

Contrairement à l'objet de petite taille faisant l'objet du suivi, les autres objets et l'arrière-plan de la scène pourront être observés dans l'image avec de nombreux détails. La détection et le suivi de l'objet sera donc difficile. Des exemples d'images analysées sont donnés dans la figure 6.1.

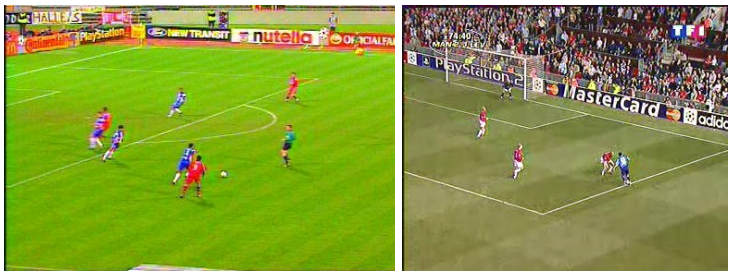


FIG. 6.1 – Exemple d'images analysées pour détecter et suivre un objet de petite taille (ici un ballon).

Comme dans les chapitres précédents, la caméra effectuant l'acquisition des images est considérée mobile. De plus, l'objet de petite taille est caractérisé par un mouvement plus difficile à prédire du fait de ses interactions avec les autres objets de la scène, comme le montre la figure 6.2. Cette remarque est particulièrement fondée lorsque l'objet suivi est une balle de tennis ou un ballon de football dans des séquences vidéo représentant des retransmissions télévisées d'événements sportifs.

Le cumul d'une caméra mobile et du faible nombre de pixels représentant l'objet entraîne une contrainte supplémentaire (qui sera assimilée à $\mathcal{C}_{\text{variabilité}}$) liée à la forme de l'objet. En effet, même si l'objet est de nature rigide, son apparence dans les images successives ne pourra pas être considérée comme constante. La figure 6.3 illustre ce constat.

Du fait de ses caractéristiques, un objet de petite taille peut facilement être assimilé à un

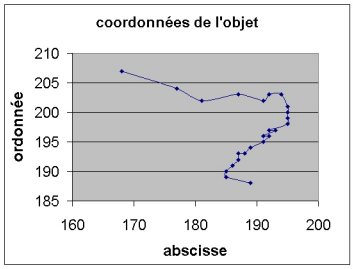


FIG. 6.2 – Exemple de la trajectroire suivie par un objet de petite taille (ici un ballon) au cours d'une séquence d'images.

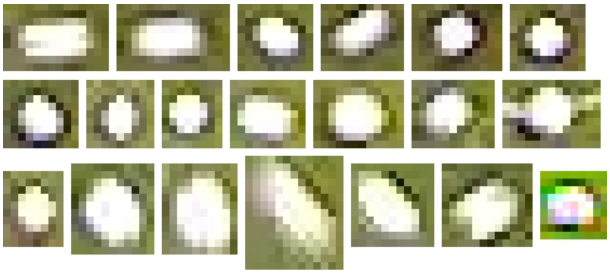

FIG. 6.3 – Apparence d'un même objet de petite taille (ici un ballon) dans différentes images.

bruit présent dans l'image. Toute la difficulté du problème réside dans la distinction entre les pixels représentant le bruit engendré par le mécanisme d'acquisition et ceux représentant les objets de petite taille que nous cherchons effectivement à détecter et suivre.

Dans ce chapitre nous proposons une solution pour réaliser le suivi d'objet de petite taille en temps réel dans des images couleur, en considérant les contraintes spécifiques énoncées ci-dessus. Ces spécificités ont entraîné plusieurs chercheurs à proposer des méthodes de suivi adaptées à ce type de suivi d'objet. Quelques-unes de ces méthodes vont être maintenant présentées.

## 6.2 Le suivi d'objet de petite taille dans la littérature

Le problème spécifique du suivi d'objet de petite taille peut être rattaché à d'autres problèmes pour lesquels des solutions sont connues, tel que le problème de la détection d'objets dans des images radar. Cependant, dans ce type d'images où le but est de séparer les objets du bruit, les caractéristiques de l'image, c'est-à-dire des objets et du bruit, sont généralement connues *a priori*.

Ce n'est pas le cas du suivi d'objet de petite taille dans des séquences vidéo quelconques. Nous nous limitons donc à ce problème et dressons ici un bref panorama des approches proposées dans la littérature, en considérant successivement le cas d'une ou plusieurs caméras fixes, celui d'une caméra mobile, et enfin celui d'une caméra mobile avec traitement en temps réel.

Dans le cas le plus simple, la caméra effectuant l'acquisition des images est considérée comme fixe ou statique. Ohno *et al.* [Ohn00] effectuent un suivi du ballon dans un match de football en deux étapes. Tout d'abord, la différence entre deux images successives permet de détecter toutes les régions mobiles. La recherche du ballon parmi ces régions repose sur l'hypothèse que le ballon correspond à une région de petite taille dont la couleur ne correspond pas à celle d'un joueur. Lipton *et al.* [Lip98] analysent des séquences d'images en niveaux de gris en considérant deux types d'objets (humains ou véhicules). Là encore, la différence entre deux images successives ou avec une image de référence permet d'isoler les régions candidates. Une mise en relation de modèles par comparaison des apparences des régions dans les images successives est ensuite effectuée. Pingali *et al.* [Pin98] se proposent de suivre une balle de tennis dans des séquences vidéo couleur. La détection des régions mobiles est toujours effectuée par différence entre deux images successives mais un post-traitement (fermeture morphologique) est nécessaire. Ensuite, la balle est distinguée des joueurs par analyse de couleur, en considérant la couleur de la balle (jaune) par des valeurs de teinte et de saturation spécifiques dans l'espace TSL. Finalement, Davies *et al.* [Dav98] utilisent des ondelettes et des filtres de Kalman pour suivre des objets dans des séquences d'images infrarouges en niveaux de gris.

En considérant la caméra mobile et non plus fixe, on augmente la complexité du problème. Cohen *et al.* [Coh98] analysent des images acquises par une caméra embarquée sur un avion et compensent le mouvement de cette caméra entre les différentes images. Les régions mobiles sont composées des pixels dont le mouvement n'est pas cohérent avec le mouvement global, c'est-à-dire celui de la caméra. En supposant une apparence relativement constante durant quelques images, chaque objet est alors suivi à l'aide d'un modèle dynamique, c'est-à-dire généré sans apprentissage, qui lui est associé. Deux approches ( [Seo97] et [Gon95b]) concernent le suivi du ballon dans des retransmissions de matchs de football. Dans les deux cas, le suivi s'effectue en trois étapes. La première étape consiste à supprimer le fond, c'est-à-dire la pelouse, par recherche des pixels caractérisés par la couleur la plus fréquente dans l'image [Seo97] ou la couleur verte [Gon95b]. Il est ensuite possible de déterminer la position de l'occurrence du ballon dans une image, soit par *template matching* [Seo97], soit par recherche de régions circulaires de couleur blanche [Gon95b]. Le suivi dans les différentes images est basé finalement sur l'utilisation de filtres de Kalman [Seo97] ou sur la recherche dans une zone localisée autour de la position précédente [Gon95b].

Enfin, il est possible de suivre des petits objets dans une scène dynamique (caméra mobile) en temps réel. Ainsi, Sanderson *et al.* [San99] traitent le problème de la détection et du suivi de navires en haute mer. Un modèle fréquentiel de l'arrière-plan, c'est-à-dire la mer, est obtenu en utilisant une transformée de Fourier rapide (FFT). En supprimant les fréquences associées à l'arrière-plan, puis en effectuant la transformée inverse, il est possible de supprimer l'arrière-plan. Les différentes régions détectées sont alors décrites par leur taille et leur centre de gravité. Ces informations sont enfin utilisées pour effectuer le suivi des objets dans les différentes images.

Parmi les méthodes de la littérature présentées ici, aucune ne permet de résoudre le problème du suivi d'objet de petite taille en respectant toutes les contraintes énoncées : les contraintes générales $\mathcal{C}_{\text{mouvement}}$, $\mathcal{C}_{\text{couleur}}$, $\mathcal{C}_{\text{rapidité}}$, $\mathcal{C}_{\text{illumination}}$, ainsi que des contraintes spécifiques relatives aux nombreux objets présents dans la scène, à l'adaptabilité de la méthode à différents objets, *etc*.

Afin de résoudre le problème posé, nous l'avons divisé en deux sous-problèmes : celui de la détection des objets de petite taille dans une image donnée, et celui du suivi de ces objets dans les différentes images de la séquence.

Nous avons étudié le premier problème en cherchant tout d'abord une solution basée sur l'analyse globale de l'image. Nous avons tenté de supprimer les détails (dont les objets de petite taille) d'une image puis de comparer le résultat avec l'image originale. La différence entre les deux images permet de localiser les objets de petite taille. Cependant, cette approche originale ne permet pas de traiter la contrainte $\mathcal{C}_{\text{rapidité}}$.

Nous avons alors dû envisager une analyse locale de l'image, connue pour être généralement plus rapide que les analyses globales. Pour cela, la répartition des couleurs des pixels est mesurée localement. Il est donc possible d'effectuer localement une segmentation en deux classes (détails et autres régions) *via* l'utilisation d'histogrammes. Cette approche, décrite dans la section 6.4, permet de respecter les différentes contraintes énoncées.

Il est finalement possible de résoudre le problème du suivi à l'aide des solutions proposées pour le problème de la détection en fusionnant les résultats obtenus sur les différentes images. L'approche utilisée sera décrite dans la section 6.5.

## 6.3 Détection des objets par analyse globale de l'image

Les objets de petite taille sont représentés dans les images par des détails. La détection des objets peut alors s'effectuer au travers d'une recherche des détails dans l'image.

Ce problème peut être assimilé au problème de séparation du fond et des objets : ici nous cherchons à séparer les zones de détails des autres zones. En reprenant les notations introduites dans la section 3.1, nous notons $\mathcal{I}$ le domaine sur lequel est définie l'image et $\mathcal{R}$ le domaine des détails de l'image. La partie complémentaire $\overline{\mathcal{R}} = \complement_{\mathcal{I}}\mathcal{R}$ représente alors les zones qui ne sont pas considérées comme des détails. Dans la section 3.3, nous avons montré que nous pouvions séparer $\mathcal{R}$ et $\overline{\mathcal{R}}$ à l'aide d'une analyse multirésolution. Nous utilisons ici cette même approche multirésolution pour détecter les objets de petite taille.

### 6.3.1 Principe

La suppression des détails présents dans une image s'effectue par changements successifs de résolution. Dans cette méthode multirésolution, l'image peut donc être représentée par une pyramide dont la base serait l'image originale. Deux étapes sont nécessaires pour supprimer les détails. Dans un premier temps, la résolution de l'image est diminuée jusqu'à atteindre la résolution $r_{\text{max}}$. Cette diminution a pour effet de supprimer les détails, notamment les objets de petite taille. Dans un second temps, le processus inverse est réalisé et la résolution est augmentée jusqu'à obtenir une image de taille identique à l'image originale. Il est alors possible de comparer les deux images en calculant leur différence.

Différentes techniques peuvent être utilisées pour effectuer les changements de résolution. Lorsque la résolution décroît, chaque nouveau pixel est caractérisé à partir des caractéristiques d'un ensemble de pixels de l'image à la résolution précédente. Nous utilisons la moyenne de l'ensemble pour générer la nouvelle valeur. Lorsque la résolution augmente, un ensemble de pixels est créé à partir d'un seul pixel de l'image à la résolution précédente. Nous utilisons, dans ce cas, une recopie des pixels d'origine pour générer les nouvelles valeurs. Nous avons respectivement préféré la moyenne et la recopie à la médiane et l'interpolation du fait de leur plus faible coût de calcul. De plus, nous avons vérifié expérimentalement que les mesures plus complexes (médiane et interpolation) n'apportaient pas de meilleurs résultats.

Le traitement précédent permet la suppression des détails présents dans l'image. Les objets de petite taille ne sont alors plus représentés dans l'image obtenue. Afin de localiser ces objets, une différence entre l'image originale et l'image filtrée est donc effectuée. La différence obtenue pour chaque pixel est finalement comparée à un seuil de tolérance $S_{\text{diff}}$. Une valeur de différence

supérieure au seuil correspond à la disparition d'un détail à la position analysée. Il est ainsi possible d'étiqueter les différents pixels de l'image de différence en "détail" (pouvant appartenir aux objets de petite taille) ou "autre région".

A l'issue du traitement nous disposons d'une image binaire. Les objets de petite taille étant composés des pixels étiquetés comme "détail", une analyse de l'image binaire est alors effectuée pour déterminer la position exacte de ces objets. Cette analyse consiste en la recherche, dans une fenêtre de taille prédéfinie $N \times N$, des régions ou ensembles de pixels de détail connexes dont la taille (c'est-à-dire le nombre de pixels appartenant à la région) est comprise dans un intervalle fixé selon l'objet recherché. Afin d'éviter les régions appartenant à des fenêtres voisines, nous ne considérons pas les régions appartenant à plusieurs fenêtres. Les fenêtres utilisées pour l'analyse se chevauchent donc pour ne pas perdre de région pouvant correspondre à un objet de petite taille. Le décalage utilisé est égal à $\frac{N}{2}$.

### 6.3.2 Résultats

Afin de valider l'approche proposée, nous avons effectué des tests sur des images couleur RGB issues de retransmissions télévisées de matchs de football. Le petit objet considéré est ici le ballon. Les paramètres utilisés ont été déterminés expérimentalement et sont donnés dans le tableau 6.1.

| Paramètre | Valeur |
|---|---|
| Nombre de pixels minimum de l'objet recherché | 10 |
| Nombre de pixels maximum de l'objet recherché | 70 |
| Taille $N \times N$ de la fenêtre analysée | $50 \times 50$ |
| Hauteur de la pyramide $r_{\text{max}}$ | 4 |
| Seuil de différence $S_{\text{diff}}$ | 60 |

TAB. 6.1 – Paramètres utilisés pour la détection globale de petits objets (ballon dans un match de football).

La figure 6.5 illustre les résultats obtenus en considérant l'image originale donnée dans la figure 6.4. La colonne de gauche représente les résultats obtenus à l'aide de la moyenne et de la recopie tandis que la colonne de droite montre l'utilisation de la médiane et de l'interpolation. Les différentes lignes, quant à elles, illustrent le nombre de changements de résolutions effectués. Il est possible de noter que plus ce nombre (qui est équivalent à la hauteur de la pyramide) est important, plus la qualité de la détection est importante. Cependant, le temps de calcul étant proportionnel au nombre d'opérations effectuées, cette amélioration de la qualité s'accompagne d'un temps de calcul plus important. En considérant le jeu de paramètres décrit dans le tableau

6.1 et une architecture PC Pentium 4 candencé à 1700 MHz, le temps de calcul requis est égal à 65 millisecondes pour une image de taille $384 \times 288$ pixels.

FIG. 6.4 – Exemple d'image utilisée pour la détection des objets de petite taille.

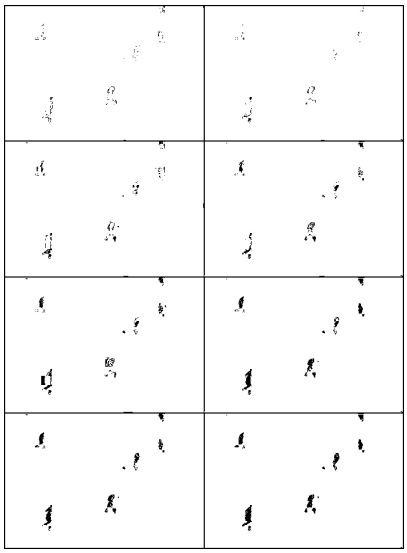

FIG. 6.5 – Détection des objets de petite taille par approche globale : les changements successifs de résolution sont effectués en considérant la moyenne et la recopie (à gauche) ou la médiane et l'interpolation (à droite), pour une hauteur de pyramide variant de $r_{\text{max}} = 2$ (en haut) à $r_{\text{max}} = 5$ (en bas).

Afin de supprimer les détails présents dans les images, nous avons également mené des expériences basées sur l'utilisation d'opérateurs de granulométrie (ouverture et fermeture morphologiques). Les résultats obtenus, quoiqu'également intéressants, n'étaient pas suffisants pour accepter le surcoût nécessaire en temps de calcul.

D'après les résultats obtenus, cette méthode globale ne permet pas de respecter la contrainte $\mathcal{C}_{\text{rapidité}}$. Nous avons donc envisagé une approche locale pour résoudre le problème dans un laps de temps plus court.

## 6.4 Détection des objets par analyse locale de l'image

Nous avons vu précédemment une approche globale pour résoudre le problème de la détection des objets de petite taille dans une image. Cette approche donne des résultats intéressants mais ne permet pas de respecter la contrainte $\mathcal{C}_{\text{rapidité}}$. Nous avons alors cherché une solution basée sur une analyse plus locale de l'image. Plus précisément, ici les objets de petite taille ne sont pas détectés à la suite d'une suppression globale des détails de l'image originale mais directement par une segmentation locale basée sur une répartition des couleurs des pixels.

### 6.4.1 Principe

La segmentation est effectuée sur des fenêtres de taille $N \times N$ décalées de $\frac{N}{2}$ afin de détecter tous les objets de petite taille présents dans la scène. En observant les histogrammes des images associées à chaque composante couleur, nous remarquons la présence d'une gaussienne représentant l'arrière-plan de la scène ou les objets de taille importante. Les valeurs des pixels correspondant aux détails, quant à elles, sont réparties de manière plus uniforme sur la plage disponible.

Pour chaque composante couleur, un seuil proportionnel à la valeur maximale dans l'histogramme est donc calculé. En notant $H_{\text{max}}$ la valeur maximale de l'histogramme, ce seuil est défini comme étant égal à $\alpha \times H_{\text{max}}$ et est utilisé pour binariser l'image correspondante. Le résultat final est obtenu par fusion des résultats associés à chacune des trois composantes couleurs.

L'image binaire obtenue est alors analysée d'une manière similaire à celle utilisée dans la méthode précédente. Les régions retenues respectent deux conditions. D'une part, elles sont strictement contenues dans la fenêtre considérée. D'autre part, leur taille est comprise dans un intervalle prédéfini en fonction du type d'objet recherché.

### 6.4.2 Résultats

Afin d'évaluer l'apport de cette approche locale, nous avons utilisé le même ensemble de tests que précédemment. Les paramètres communs avec l'approche précédente sont également identiques (c.f. tableau 6.2).

Les résultats obtenus, illustrés par les figures 6.6 et 6.7, montrent l'efficacité de la méthode proposée dans cette section. Les temps de calcul, quant à eux, sont relativement faibles puisque

| Paramètre | Valeur |
|---|---|
| Nombre de pixels minimum de l'objet recherché | 10 |
| Nombre de pixels maximum de l'objet recherché | 70 |
| Taille $N \times N$ de la fenêtre analysée | $50 \times 50$ |
| Coefficient $\alpha$ utilisé pour l'analyse des histogrammes | 0.025 |

TAB. 6.2 – Paramètres utilisés pour la détection locale de petits objets.

nous les avons estimés à une durée de l'ordre de 15 à 30 millisecondes par image. Nous considérons ici encore une architecture PC Pentium 4 candencé à 1700 MHz. A l'aide d'une analyse locale, nous pouvons donc résoudre le premier problème posé en prenant notamment en compte la contrainte $\mathcal{C}_{\text{rapidité}}$.

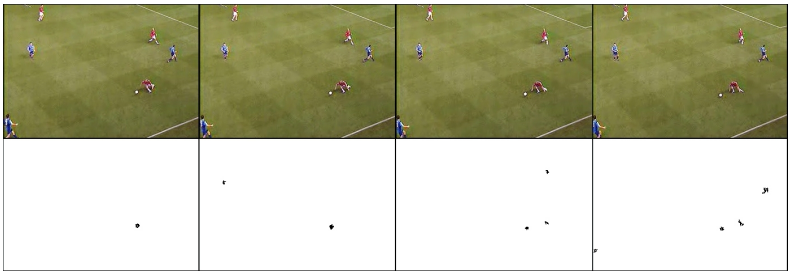

FIG. 6.6 – Détection des objets de petite taille par segmentation locale dans une scène simple : images originales (en haut) et résultats (en bas).

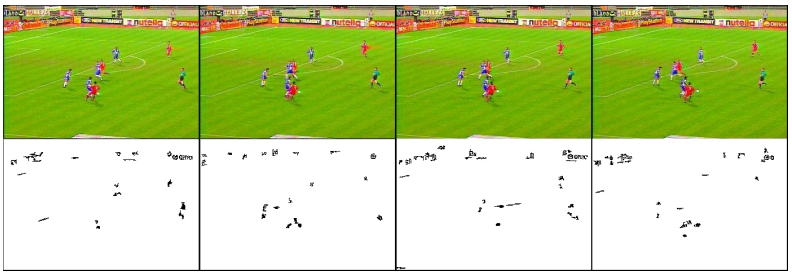

FIG. 6.7 – Détection des objets de petite taille par segmentation locale dans une scène plus complexe : images originales (en haut) et résultats (en bas).

De même que pour la méthode globale décrite dans la section précédente, nous n'avons pas abordé ici l'évaluation quantitative de la qualité des méthodes en termes de précision et rappel. En effet, ces deux méthodes de détection ont pour but, non pas de détecter les objets de petite taille sans erreur, mais de fournir des candidats potentiels (qui pourront être éliminés par la suite) pour l'étape de suivi. Cette étape fait l'objet de la prochaine section.

## 6.5 Suivi de l'objet sur plusieurs images

Après avoir détecté les objets de petite taille dans une image, nous devons maintenant réaliser leur suivi dans la séquence vidéo. Ce problème difficile va notamment permettre de séparer les régions qui sont réellement des objets de petite taille des régions qui correspondent au bruit présent dans les images. Le but est donc double : suivre les objets d'une image à l'autre, et éliminer les candidats obtenus qui correspondent au bruit.

### 6.5.1 Principe

Afin de suivre ces objets tout au long d'une séquence vidéo, il est possible de fusionner les résultats obtenus par la détection des objets sur les différentes images de la séquence analysée. Cependant, il n'est pas toujours possible de prédire la trajectoire d'un objet de petite taille lorsque la source d'acquisition est elle-même en mouvement. Nous n'effectuerons donc pas ici d'estimation de mouvement de l'objet suivi ni de prédiction de sa position dans les trames suivantes de la séquence vidéo.

La détection des objets dans une image $I_t$ nous fournit des régions candidates que nous noterons $R_i(I_t)$ avec $i$ l'indice du candidat dans la liste $\mathcal{L}(I_t)$ des candidats détectés dans l'image $I_t$. Chaque candidat $R_i(I_t)$ peut être représenté par un vecteur de caractéristiques :

$$R_i = \begin{pmatrix} c_i \\ f_i \end{pmatrix} \tag{6.1}$$

où $c_i$ et $f_i$ sont la position et la forme de la région $R_i$.

Le suivi, quant à lui, doit permettre de connaître la position d'un objet $O$ tout au long de la séquence. Dans une image donnée $I_t$, l'objet $O$ sera associé à une des régions $R_i(I_t)$. Nous pouvons alors définir le prédicat $\Phi$ qui associe deux régions de deux images différentes si elles correspondent au même objet $O$ :

$$\Phi(R_i(I_{t_1}), R_j(I_{t_2})) = \begin{cases} \text{vrai} & \text{si } R_i(I_{t_1}) \text{ et } R_j(I_{t_2}) \text{ représentent le même objet } O \\ \text{faux} & \text{sinon} \end{cases} \tag{6.2}$$

Deux régions $R_i(I_{t_1})$ et $R_j(I_{t_2})$ représentent le même objet $O$ si elles vérifient :

$$\begin{cases} \text{aire}(f_i) \approx \text{aire}(f_j) \\ \|c_i - c_j\| < \Delta_s \\ (i,j) = \underset{\substack{i \in \mathcal{L}(I_{t_1}) \\ j \in \mathcal{L}(I_{t_2})}}{\arg\min} \|c_i - c_j\| \end{cases} \tag{6.3}$$

La technique utilisée pour suivre un petit objet consiste à mettre en relation les régions détectées d'une image à l'autre, c'est-à-dire déterminer les régions $R_i(I_{t-1})$ et $R_j(I_t)$ telles que le prédicat $\Phi(R_i(I_{t-1}), R_j(I_t))$ soit vrai. L'objet suivi sera validé si plusieurs images consécutives contiennent une occurrence de celui-ci. Cette hypothèse se justifie par le fait que le résultat obtenu par le processus de détection des objets de petite taille consiste en de nombreux candidats parmi lesquels l'objet recherché et d'autres régions relatives au bruit. Or ces dernières régions ne seront pas détectées dans toutes les images, contrairement à l'objet que l'on cherche à suivre. Nous pouvons ainsi éliminer des régions candidates les régions correspondant au bruit. Le processus de mise en relation est alors le suivant. Pour chaque région détectée $R_j(I_t)$, nous cherchons (dans un voisinage de taille fixée) la région $R_i(I_{t-1})$ qui vérifie le prédicat $\Phi(R_i(I_{t-1}), R_j(I_t))$, c'est-à-dire l'occurrence de l'objet concerné parmi les résultats obtenus sur l'image précédente, en considérant l'équation (6.3).

Nous associons à chaque objet un poids qui représente le nombre d'images dans lesquelles il apparaît. A chaque instant, l'objet de plus fort poids correspond, selon notre hypothèse, à l'objet suivi.

Lorsqu'une région $R_i(I_t)$ ne peut être appariée avec les résultats $R_j$ obtenus sur un nombre d'images donné $\Delta_t$, nous ne prenons plus en compte l'objet $O$ correspondant. Cela nous permet de gérer correctement les occlusions temporaires des objets à suivre. De plus, lorsque le nombre d'occurrences de l'objet suivi au cours du temps s'accroît, la confiance donnée au résultat est plus importante. Nous limitons alors la détection des objets à une zone de l'image centrée autour de cette position de l'objet suivi dans l'image précédente. Le temps de calcul se trouve ainsi diminué.

### 6.5.2 Résultats

Nous considérons ici que la détection des objets a été obtenue par l'approche locale décrite dans la section 6.4. Cette approche a évidemment été préférée à la première solution étudiée (analyse globale) car elle respecte toutes les contraintes formulées, et tout particulièrement la

contrainte $\mathcal{C}_{\text{rapidité}}$.

La figure 6.8 illustre le résultat obtenu sur un extrait d'une séquence d'images. La ligne horizontale associée à chaque région représente le nombre d'apparitions de l'objet dans les différentes images. L'objet à suivre correspond bien à la région la plus fréquente dans la séquence à un instant donné.

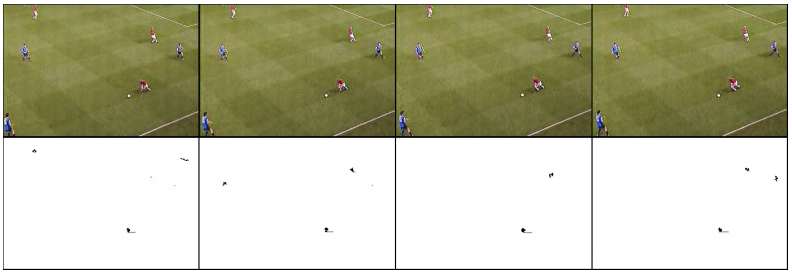

FIG. 6.8 – Suivi des objets de petite taille dans les images successives d'une séquence vidéo : images originales (en haut) et résultats (en bas).

Les paramètres utilisés pour obtenir le résultat précédent sont donnés dans le tableau 6.3.

| Paramètre | Description | Valeur |
|---|---|---|
| $\Delta_t$ | Nombre d'images considérées pour le suivi | 4 |
| $\Delta_s$ | Déplacement maximal (en pixels) d'un objet suivi entre deux images consécutives | 15 |

TAB. 6.3 – Paramètres utilisés pour la détection et le suivi de petits objets.

## 6.6 Conclusion

Dans ce chapitre, nous avons traité le problème du suivi d'objets de petite taille. Nous étions à la recherche d'une solution considérant les différentes contraintes imposées : $\mathcal{C}_{\text{couleur}}$, $\mathcal{C}_{\text{mouvement}}$, $\mathcal{C}_{\text{rapidité}}$, $\mathcal{C}_{\text{variabilité}}$, scène complexe, mouvement non-linéaire, *etc.*

Afin de prendre en compte la contrainte $\mathcal{C}_{\text{mouvement}}$, nous avons choisi de ne pas effectuer d'estimation de mouvement, puisque celui-ci peut être non-linéaire. Nous avons déterminé les caractéristiques des objets à suivre, ce qui nous a permis de gérer plus facilement l'aspect $\mathcal{C}_{\text{variabilité}}$. Pour résoudre le problème posé, nous l'avons tout d'abord modélisé comme un problème de séparation du fond et des objets en considérant que, dans ce cas, les objets étaient de petite taille et que toutes les autres régions pouvaient être assimilées à l'arrière-plan. Nous

avons donc cherché à réutiliser les concepts et solutions énoncés dans le chapitre 3 et nous avons proposé une approche multirésolution pour résoudre le problème. Les résultats obtenus ne permettant pas de considérer avec exactitude la contrainte $\mathcal{C}_{\text{rapidité}}$, nous avons ensuite proposé une solution par analyse locale de l'image, qui consiste à segmenter localement l'image pour y détecter les objets de petite taille. La solution obtenue permet de respecter toutes les contraintes formulées.

Cependant, lorsque l'objet à suivre est non-rigide, la solution proposée ici n'est pas adaptée. Ce nouveau problème sera abordé dans le prochain chapitre.

# Chapitre 7

# Suivi d'objet non-rigide par contours actifs

Dans les chapitres précédents, nous avons traité le problème du suivi d'objet rigide, que ce soit avec ou sans apprentissage. Dans ce chapitre, nous abordons le problème du suivi d'objet non-rigide. Pour cela, nous proposons d'utiliser les contours actifs, ou *snakes*, particulièrement adaptés pour segmenter des objets non-rigides. Les algorithmes utilisant les contours actifs sont souvent caractérisés par des temps de calcul importants, rendant impossible un suivi des objets en temps réel. Nos travaux ont donc principalement porté sur l'amélioration de la complexité algorithmique de ces outils [Lef00a, Lef01b, Lef02d].

Après avoir présenté la problématique du suivi d'objet non-rigide, nous expliquerons comment les contours actifs peuvent être utilisés pour suivre des objets dans des séquences vidéo. Pour cela, nous présenterons différents modèles et implémentations de contours actifs. Nous détaillerons aussi quelques méthodes de suivi par contours actifs publiées dans la littérature. Dans un second temps, nous décrirons la méthode de suivi proposée. Celle-ci est composée de deux étapes effectuées sur chaque image : l'initialisation du contour actif puis sa déformation en utilisant des énergies données. Une extension de la méthode au cas du suivi d'objets multiples sera ensuite présentée. Afin de diminuer le temps de calcul nécesssaire, nous proposons d'analyser les images suivant une approche multirésolution. Finalement des résultats seront présentés afin de montrer l'intérêt de notre contribution.

## 7.1 Problématique

Les méthodes de suivi d'objet rigide se basent sur la recherche d'un même motif dans des images successives. Les objets non-rigides (comme les corps humains), de par leur nature déformable, ne peuvent être suivis avec les mêmes outils. Il est donc nécessaire d'utiliser des approches appropriées, tels les modèles déformables ou les contours actifs. Les modèles déformables [Jai98b] sont particulièrement intéressants lorsque l'on peut modéliser la forme de l'objet que l'on cherche à suivre. Dans le cas contraire, on utilisera plutôt les contours actifs. Les contours actifs sont notamment utilisés en segmentation d'image et en suivi d'objet [Bla99]. Nous cherchons des outils les plus génériques possibles pour permettre la mise en place de systèmes d'indexation spécifiques. Nous avons donc choisi d'utiliser les contours actifs plutôt que les modèles déformables pour suivre les objets non-rigides.

Différents modèles de contours actifs ont été proposés dans la littérature. Ainsi, on doit à Kass *et al.* [Kas88] les travaux originaux dans ce domaine. Le modèle utilisé, appelé *snakes*, a cependant montré différentes limites, liées notamment à l'initialisation, au paramétrage, et à l'impossibilité de changement de topologie du *snake*. Certains auteurs ont donc proposé d'autres modèles. Parmi eux, les contours géodésiques [Par00] (de la famille des approches *level set* [Set96, Set99]), préconisés actuellement dans la littérature, permettent de gérer des topologies quelconques, mais se traduisent par des temps de calcul importants. D'après Aubert et Blanc-Féraud [Aub99], les *snakes* et les contours géodésiques peuvent être considérés dans une certaine mesure comme équivalents.

Puisque nous souhaitons effectuer le suivi des objets non-rigides en temps réel, nous avons préféré les *snakes* aux autres approches, plus puissantes mais aussi plus coûteuses en temps de calcul. Cependant, les *snakes* présentent différentes limites que nous précisons :

**initialisation :** afin d'obtenir un résultat correct, il est souvent nécessaire d'initialiser le *snake* proche du contour de l'objet recherché,

**paramétrage :** le modèle de *snake* comporte de nombreux paramètres, souvent difficiles à régler,

**topologie :** les *snakes* ne permettent pas un changement de topologie (comme une scission si le contour regroupe deux objets),

**temps de calcul :** même si les *snakes* sont moins coûteux en ressources que les approches par *level set*, ils restent néanmoins difficiles à exécuter en temps réel.

Nous avons développé une technique de *snakes* pour effectuer un suivi d'objet non-rigide en temps réel et envisagé une méthode de scission. Nous présenterons tout d'abord une introduction aux contours actifs et décrirons leur utilisation pour le suivi d'objet.

## 7.2 Contours actifs et suivi d'objet

Les contours actifs ont été introduits par Kass *et al.* [Kas88] et font l'objet de nombreux travaux récents [Bla99]. Dans cette section, nous présenterons les principaux modèles de contours actifs. Nous détaillerons ensuite les différentes énergies utilisées. Enfin, nous dresserons un bref panorama des méthodes de suivi d'objet basées sur les contours actifs.

### 7.2.1 Modèles de contour actif

Un contour actif peut être représenté sous la forme d'une courbe, fermée ou non, et évoluant dans le temps :

$$\begin{array}{rccl} v: & [0,1] & \longrightarrow & \mathbb{R}^2 \\ & s & \longmapsto & v(s) = [x(s), y(s)] \end{array} \tag{7.1}$$

On confondra ici la courbe et sa représentation paramétrique. La courbe est déformée de manière itérative afin de minimiser une fonctionnelle d'énergie. Cette fonction peut être définie dans le domaine continu comme :

$$E(v) = \int_0^1 \left[\alpha_{\text{int}} E_{\text{int}}(v(s)) + \alpha_{\text{ext}} E_{\text{ext}}(v(s))\right] ds \tag{7.2}$$

où $E_{\text{int}}$ et $E_{\text{ext}}$ représentent les énergies internes et externes. Les coefficients $\alpha_{\text{int}}$ et $\alpha_{\text{ext}}$ sont utilisés pour donner plus ou moins d'influence à chaque énergie.

Plusieurs approches ont été proposées pour l'implémentation du modèle de contour actif : le calcul variationnel [Kas88], la programmation dynamique [Ami90], ou encore l'algorithme greedy [Wil92]. Dans une étude comparative [Den95], il a été montré que l'algorithme greedy est plus rapide (d'un facteur 10 à 80) que les méthodes par calcul variationnel ou programmation dynamique.

Nous utilisons donc cette approche, qui consiste en une implémentation discrète et locale du modèle de contour actif. La définition, dans le domaine discret, de la fonction d'énergie du *snake* est alors la suivante :

$$E = \sum_{i=1}^{m} E(V(i)) \tag{7.3}$$

où $V$ représente le contour actif discret et $V(i)$ son i$^{\text{ème}}$ point. Ce contour actif est constitué par un nombre $m$ de points. Le principe de la méthode est de faire évoluer itérativement les points du contour actif $V$. Nous utilisons la notation $V^\gamma$ pour représenter le contour actif $V$ à l'itération $\gamma$.

En chaque point $V(i)$ de ce contour, la fonction d'énergie $E(V^\gamma(i))$ est calculée pour tous les points $M_j$ appartenant au voisinage de $V(i)$, soit $M_j \in \mathcal{V}(V(i))$. Le point $M_{j'}$ caractérisé par l'énergie minimale, alors noté $V^{\gamma+1}(i)$, remplace le point $V^\gamma(i)$ si $E(M_{j'}) < E(V^\gamma(i))$. Dans le cas contraire, le point de contour n'est pas modifié $V^{\gamma+1}(i) = V^\gamma(i)$. Ce principe traduit l'effet d'une force dont la direction est $\overrightarrow{V^\gamma(i)V^{\gamma+1}(i)}$. L'énergie pour un point $V(i)$ est donnée par :

$$E(V(i)) = \alpha_{\text{int}} E_{\text{int}}(V(i)) + \alpha_{\text{ext}} E_{\text{ext}}(V(i)) \tag{7.4}$$

Ce processus itératif de déformation est effectué pour chaque point de la courbe, jusqu'à convergence en utilisant un critère d'arrêt donné. A ce sujet, Wong *et al.* [Won98] proposent une revue critique des critères d'arrêt existants. Le critère que nous utilisons repose sur la comparaison du contour obtenu à l'itération $\gamma$ avec celui obtenu à l'itération $\gamma - 1$. Si les deux contours sont identiques, le processus s'arrête.

### 7.2.2 Définition des énergies

Dans les équations (7.2) à (7.4), le modèle de contour actif est basé sur le calcul en chaque point d'une énergie $E$, composée d'une énergie interne $E_{\text{int}}$ et d'une énergie externe $E_{\text{ext}}$. Nous détaillons ici le calcul de ces deux énergies, composées elles-mêmes de différentes énergies. Chacune de ces énergies est liée à une force, dont le comportement permettra de comprendre l'intérêt et l'influence de l'énergie dans l'évolution du contour actif. Parmi les énergies que nous avons considérées, certaines ($E_{\text{cont}}$, $E_{\text{cour}}$, et $E_{\text{grad}}$) sont issues de la littérature [Kas88], tandis que d'autres sont plus originales ($E_{\text{coul}}$) ou adaptées de la littérature [Coh91] à notre problème ($E_{\text{ball}}$).

L'énergie interne représente les propriétés physiques du contour, et n'est donc pas liée au contenu de l'image. Elle est calculée souvent comme la somme de trois énergies :

$$E_{\text{int}} = \alpha_{\text{cont}} E_{\text{cont}} + \alpha_{\text{cour}} E_{\text{cour}} + \alpha_{\text{ball}} E_{\text{ball}} \tag{7.5}$$

où $E_{\text{cont}}$, $E_{\text{ball}}$, et $E_{\text{cour}}$ définissent respectivement les énergies de continuité, de ballon, et de courbure. Les coefficients permettent le paramétrage du *snake*.

La force associée à l'énergie de continuité influe sur le rayon de courbure du contour en conduisant les points du contour à se positionner de manière à être équidistants. La forme d'un contour fermé tend alors vers un cercle. L'énergie $E_{\text{cont}}(M(j))$ est définie comme :

$$E_{\text{cont}}(M(j)) = \left|\overline{V} - V(i-1)M(j)\right| \tag{7.6}$$

où $AB$ représente la longueur du segment $[AB]$ et $\overline{V}$ la distance moyenne entre deux points successifs du contour. Cette dernière est calculée à chaque itération du contour.

La seconde force interne, associée à l'énergie de courbure, a pour but d'éviter que le contour ne contienne des points isolés qui ne seraient pas cohérents avec la forme globale et régulière supposée du contour. Elle peut s'exprimer de la manière suivante :

$$E_{\text{cour}}(M(j)) = \frac{V(i-1)M(j) + M(j)V(i+1)}{V(i-1)V(i+1)} \tag{7.7}$$

La dernière force interne utilisée a été introduite par Cohen [Coh91] sous le nom de force ballon. Puisqu'un *snake* composé uniquement des deux premières énergies aurait tendance à se contracter, cette nouvelle force lui permet de compenser l'effet des deux premières forces. L'énergie est définie comme le produit scalaire de deux vecteurs :

$$E_{\text{ball}}(M(j)) = \overrightarrow{N(i)} \cdot \overrightarrow{V(i)M(j)} \tag{7.8}$$

où $\overrightarrow{N(i)}$ représente le vecteur normal extérieur au contour V au point $V(i)$.

L'énergie externe permet d'associer le contour actif au contenu de l'image. De même que l'énergie interne, l'énergie externe est composée de plusieurs énergies, chacune associée à une force :

$$E_{\text{ext}} = \alpha_{\text{grad}} E_{\text{grad}} + \alpha_{\text{coul}} E_{\text{coul}} \tag{7.9}$$

où $E_{\text{grad}}$ et $E_{\text{coul}}$ sont deux énergies liées respectivement aux informations de gradient et de couleur.

La première force externe est basée sur le gradient $\nabla I$ calculé sur l'image couleur $I$. Elle a pour but de fixer le contour sur les contours réels des objets de l'image. Nous avons choisi d'estimer le gradient d'une image couleur par la somme des gradients calculés sur les différentes composantes en utilisant l'opérateur de Sobel [Sob90]. Afin d'éliminer la plupart des contours non significatifs et de limiter la sensibilité au bruit, le résultat obtenu est ensuite seuillé. L'énergie $E_{\text{grad}}$ est donc définie par :

$$E_{\text{grad}}(M_j) = \begin{cases} -\|\nabla I(M_j)\| & \text{si } \|\nabla I(M_j)\| > S_{\text{grad}}, \\ 0 & \text{sinon} \end{cases} \tag{7.10}$$

où $\|\nabla I(M_j)\|$ représente le module du gradient dans l'image $I$ au point $M_j$. Comme il a été précisé dans la section 4.3.1, il est possible d'utiliser pour cela différentes méthodes adaptées aux images couleur [Car94].

La seconde force externe permet au *snake* de rester sur les frontières de l'objet suivi. Pour cela, elle est définie en utilisant des informations *a priori* sur les couleurs présentes dans l'arrière-plan de l'image. En particulier, il est possible de calculer la couleur moyenne de l'arrière-plan. L'énergie $E_{\text{coul}}$ est ici définie en chaque point de l'image comme la différence entre la couleur du point en question et la couleur moyenne du fond. Afin de limiter la sensibilité au bruit, la valeur obtenue est seuillée. La définition de l'énergie liée à la couleur de l'arrière-plan est donc la suivante :

$$E_{\text{coul}}(M_j) = \begin{cases} \|I(M_j) - I_{\text{fond}}\| & \text{si } \|I(M_j) - I_{\text{fond}}\| > S_{\text{coul}}, \\ 0 & \text{sinon} \end{cases} \tag{7.11}$$

où $I_{\text{fond}}$ représente la couleur de l'arrière-plan. Le seuil utilisé pour limiter la sensibilité au bruit est noté $S_{\text{coul}}$.

Pour une description plus complète des différentes énergies utilisées habituellement dans les contours actifs et de leur interprétation, on pourra se référer à [Dav00]. Le choix des coefficients utilisés pour pondérer les différentes énergies est généralement effectué de manière empirique. Davatzikos et Prince proposent dans [Dav99] une étude sur l'influence de ces coefficients dans l'obtention de solutions qui seraient des minimaux locaux et non globaux. Vincent et Rousselle décrivent dans [Vin02] des techniques permettant de déterminer les coefficients optimaux. Nous n'aborderons pas ce problème dans le cadre de cette thèse.

### 7.2.3 Suivi d'objet par contours actifs

Les *snakes* sont particulièrement adaptés pour le suivi d'objet non-rigide. Nous dressons ici un bref panorama de ces méthodes proposées dans la littérature.

Les premiers travaux utilisant les *snakes* pour le suivi d'objet sont ceux de Kass *et al.* [Kas88] et ceux de Leymarie et Levine [Ley93], qui utilisent le *snake* final de l'image précédente comme *snake* initial sur l'image courante. Afin de gérer des mouvements importants ou des fréquences de trame faibles, Terzopoulos et Szeliski [Ter92] et Peterfreund [Pet99] proposent d'utiliser un filtre de Kalman pour estimer le mouvement. Dellandrea *et al.* présentent dans [Del00] une technique similaire pour des séquences d'images médicales. Vieren *et al.* [Vie95] placent un *snake* global sur les bords de l'image afin de détecter et de suivre les nouveaux objets entrant dans la scène. A l'opposé, Ge et Tian [Ge02] initialisent les *snakes* par les centres de divergence détectés dans l'image. Les *snakes* s'agrandissent alors pour se fixer sur les contours des différents objets présents dans les images. Lorsque l'arrière-plan est complexe,

certains auteurs ont ajouté un nouveau terme d'énergie, comme l'énergie de texture introduite par Delagnes *et al.* [Del95] ou le terme d'erreur de compensation de mouvement utilisé par Pardas et Sayrol [Par01]. Venegas-Martinez introduit dans sa thèse [VM01] une méthode de suivi qui combine les résultats d'une segmentation des images par front de propagation avec les informations relatives à la compensation du mouvement entre deux images successives. Lam et Yuen [Lam98] proposent une modification de la formulation de l'énergie. La méthode de suivi obtenue donne alors de meilleurs résultats que celle basée sur l'algorithme greedy de Williams et Shah [Wil92]. Ronfard [Ron94] intègre dans la définition des forces des attributs de région obtenus par des statistiques locales. Chesneaud [Che00] utilise également des techniques statistiques afin de disposer d'une méthode de suivi robuste, même en l'absence de contour net entre les objets et le fond. Bonnet *et al.* [Bon95, Bon97] utilisent des statistiques robustes pour définir leur modèle de contour actif (appelé *snake robuste*). Leur modèle comprend, outre des classiques énergies intra-trames, des énergies inter-trames permettant de prendre en compte les informations spatio-temporelles de la séquence d'images. Rendon Mancha [Ren02] intègre des techniques de morphologie mathématique au modèle de contour actif. Une modélisation radiale du contour a été proposée par Denzler et Nieman dans [Den99] et par Chen *et al.* dans [Che01a] afin d'éviter des croisements dans le contour et de réduire le problème de minimisation d'énergie depuis le plan image 2-D vers un espace 1-D. Comme de nombreux auteurs ont proposé des énergies basées sur les caractéristiques des objets suivis, Marques [Mar98] a étudié le lien entre ces énergies et les énergies basées sur le contenu de l'image. Il en résulte la possibilité d'estimer des énergies liées à l'image par des énergies liées aux caractéristiques des objets. Finalement, l'ouvrage de Blake et Isard [Bla99] propose différentes méthodes de suivi, notamment en temps réel.

Dans cette section nous avons présenté le modèle de contours actifs utilisé, les différents termes d'énergie employés, ainsi que mentionné quelques méthodes de suivi par contour actif. Nous allons maintenant décrire les mises en œuvre rapides ainsi que les solutions techniques que nous proposons.

## 7.3 Méthode de suivi

Le but de la méthode présentée est de suivre des objets non-rigides en temps réel dans des images couleur acquises avec une caméra en mouvement. Afin de minimiser le temps de calcul, nous avons été amené à faire différents choix qui différencient notre contribution des approches de la littérature. Tout d'abord, nous choisissons de n'appliquer aucun prétraitement avant la déformation du contour actif, contrairement à la plupart des méthodes présentées dans la littéra-

ture [Van97]. Ensuite, aucune compensation du mouvemement de la caméra n'est effectuée. De plus, nous n'estimons pas non plus le mouvement des différents objets suivis. Enfin, le calcul du gradient est limité à une zone restreinte autour de la position initiale de l'objet suivi.

L'approche que nous avons proposée [Lef00a, Lef01b] est composée de deux étapes réalisées successivement sur chaque trame de la séquence vidéo. Tout d'abord le *snake* est initialisé en utilisant le résultat obtenu à l'image précédente. Il est ensuite déformé en utilisant les énergies décrites dans la section 7.2.2. Ces deux étapes sont décrites ci-dessous. Nous donnerons ensuite quelques résultats préliminaires nous permettant d'illustrer les qualités et les limites de l'approche proposée.

### 7.3.1 Initialisation

La première étape consiste en l'initialisation du *snake* sur l'image courante. Cette initialisation est effectuée là encore en deux étapes. Dans un premier temps, le rectangle $R[0]$ à côtés parallèles aux contours de l'image et circonscrit au *snake* final obtenu à l'image précédente est créé. La taille de ce rectangle est ensuite augmentée de sorte qu'il englobe *a priori* le *snake* final que nous allons obtenir pour l'image courante. Puisque nous utilisons une implémentation discrète et locale du contour actif, nous construisons le *snake* initial en plaçant les points le composant uniformément sur le contour du rectangle. Le nombre de points utilisé dépend évidemment de la précision souhaitée pour le résultat.

Toutes les images d'un même plan ne sont pas considérées de la même manière. En effet, une initialisation spécifique est nécessaire pour la première trame d'un plan, où aucune information sur le *snake* n'est disponible *a priori*. Dans ce cas, l'initialisation est effectuée de manière automatique en utilisant le résultat d'une étape de séparation du fond et des objets, comme celle décrite dans la section 3.3.

### 7.3.2 Déformation

Une fois le *snake* initialisé, un processus de déformation intervient jusqu'à convergence en utilisant les forces décrites précédemment. Il va nous permettre de déterminer la position d'un objet $O_i(t)$ à l'instant $t$ en se basant sur sa position précédente $O_i(t-1)$.

Contrairement à son emploi usuel, la force ballon est ici utilisée pour contracter le *snake* et non le dilater. Ce choix est justifié lorsque les objets sont caractérisés par des zones hétérogènes (gradients de forte amplitude et couleurs variées) tandis que l'arrière-plan est représenté princi-

palement par des zones homogènes. La déformation d'un contour étant plus facile au travers de zones homogènes qu'au travers de zones hétérogènes, une contraction du contour depuis la zone correspondant à l'arrière-plan est plus efficace qu'un accroissement du contour depuis une zone correspondant à l'intérieur de l'objet suivi. Ce choix nous permet, de plus, d'éviter l'estimation de mouvement de l'objet suivi.

L'approche décrite ci-dessus a fait l'objet d'un ensemble de tests afin d'une part d'être validée, et d'autre part de mettre en évidence ses qualités aussi bien que ses défauts.

### 7.3.3 Résultats préliminaires

La méthode de suivi décrite ici supporte le cas d'une caméra en mouvement. Néanmoins, la sensibilité à l'environnement de l'objet suivi est importante. Ainsi le suivi peut échouer si plusieurs objets mobiles ont des positions spatiales proches. En effet, on peut se trouver dans le cas où :

$$\exists j \quad / \quad O_j(t) \subset R[O_i(t-1)] \tag{7.12}$$

c'est-à-dire lorsque l'objet suivi est proche d'un autre objet. Le *snake* initial englobera alors les deux objets. Lorsque les deux objets $O_i$ et $O_j$ s'éloignent l'un de l'autre, le modèle de contour actif décrit précédemment est incapable de se fixer sur un seul des objets suivis et continue à suivre les deux objets comme un seul. De plus, ce modèle est sensible aux fortes valeurs de gradient des pixels pouvant appartenir à l'arrière-plan (comme par exemple les lignes blanches dans des scènes de football). Ces deux problèmes sont illustrés dans la figure 7.1.

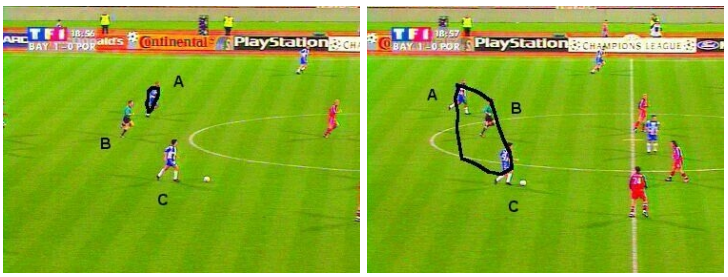


FIG. 7.1 – Incapacité du *snake* à gérer des objets proches ou un arrière-plan contenant des pixels de fort gradient. Le suivi d'objet peut être correct à un instant donné (à gauche) mais échouer après des phénomènes d'occlusion, le joueur B passant ici devant le joueur A (à droite).

Il n'est donc pas possible de traiter des séquences où plusieurs centres d'intérêt d'importance égale sont en mouvement de manière concurrente. Aussi, afin de gérer le cas d'objets multiples dans la séquence vidéo, il est nécessaire d'intégrer au modèle la possibilité d'un chan-

gement de topologie quand les trajectoires des différents objets se croisent. Cet apport est décrit maintenant.

## 7.4 Extension au cas d'objets multiples

Nous proposons une extension [Lef02d] du modèle initial de contour actif afin de résoudre les problèmes illustrés dans la figure 7.1. Cette extension permettra également de suivre plusieurs objets simultanément. Après avoir justifié la solution proposée pour répondre au problème posé, nous détaillerons l'algorithme utilisé.

### 7.4.1 Justifications

Formalisons le problème à résoudre ici afin d'y trouver une solution. Nous rappelons que le but est de suivre une forme dans chaque image. Notons $F_t$ la forme d'intérêt à l'instant $t$. Celle-ci est suivie par un *snake*, noté $V_t$. Lorsqu'un phénomène d'occlusion intervient, la forme $F_t$ représente deux objets et non un seul (car l'un des deux est caché par l'autre). Considérons l'occlusion totale terminée à l'instant $t+1$. Nous sommes alors en présence, non plus d'une seule forme $F_{t+1}$, mais de plusieurs formes. Nous nous limiterons par la suite au cas de deux formes disjointes $F^1_{t+1}$ et $F^2_{t+1}$, chacune représentant un objet. Le même raisonnement peut néanmoins s'appliquer lorsque plus de deux formes sont présentes.

Les propriétés de ces deux formes $F^1_{t+1}$ et $F^2_{t+1}$ sont les suivantes :

$$F^1_{t+1} \cap F^2_{t+1} = \varnothing \tag{7.13}$$

$$F^1_{t+1} \cup F^2_{t+1} \subseteq F_{t+1} \tag{7.14}$$

Cependant, le *snake* $V_{t+1}$, n'ayant pas connaissance de cette nouvelle information, continue de considérer la forme $F_{t+1}$ comme référentiel. C'est pourquoi le résultat illustré par la figure 7.1 est obtenu. D'après le modèle de *snake* que nous utilisons, un *snake* $V$ a pour but de suivre une et une seule forme $F$. Puisque nous sommes maintenant en présence de deux formes $F^1_{t+1}$ et $F^2_{t+1}$, nous devons utiliser deux *snakes* appropriés $V^1_{t+1}$ et $V^2_{t+1}$.

Le problème à résoudre peut alors se formuler comme la recherche de la transformation $T$ qui à un *snake* $V_{t+1}$ modélisant une forme $F_{t+1}$ associe deux *snakes* $V^1_{t+1}$ et $V^2_{t+1}$ modélisant respectivement $F^1_{t+1}$ et $F^2_{t+1}$. D'après les équations (7.13) et (7.14), il est possible de scinder $V_{t+1}$ en plusieurs *snakes* distincts afin d'obtenir $V^1_{t+1}$ et $V^2_{t+1}$.

Cependant, l'équation (7.13) n'est pas une égalité, ce qui signifie que certaines parties de la forme $F_{t+1}$ peuvent n'appartenir ni à $F^1_{t+1}$, ni à $F^2_{t+1}$. En effet, la forme $F_{t+1}$ peut représenter également l'arrière-plan visible entre les deux formes disjointes $F^1_{t+1}$ et $F^2_{t+1}$. A l'issue du processus de scission, il est alors possible que certains contours $V^i_{t+1}$ ne modélisent aucune des deux formes d'intérêt $F^1_{t+1}$ ou $F^2_{t+1}$. Il est nécessaire de ne pas prendre en considération ces contours dans le processus de suivi d'objet. Pour cela, nous devons identifier les caractéristiques des contours $V^i_{t+1}$ que nous noterons $\mathcal{Q}$. A l'aide de ces caractéristiques, nous pourrons finalement décider de conserver ou non un contour donné $V$ dans le processus de suivi.

### 7.4.2 Principe

D'après le formalisme précédent, deux étapes supplémentaires dans l'algorithme de suivi sont nécessaires : une étape de scission qui va diviser (si besoin) le *snake* en plusieurs contours et une étape de décision qui permettra de conserver uniquement les contours intéressants. L'étape de scission est basée ici sur les caractéristiques internes du *snake*, et non pas sur ses caractéristiques externes comme dans [Cho01]. Afin de limiter le temps de calcul, ces deux étapes ne sont réalisées qu'une seule fois par image, lorsqu'un contour final a été obtenu à l'aide de l'algorithme précédent.

L'étape de scission a pour but de diviser le *snake* en plusieurs contours. Pour cela, chaque couple de points successifs du contour est analysé. Si la longueur $V(i)V(i+1)$ du segment reliant deux points successifs du contour $V(i)$ et $V(i+1)$ est supérieure à un seuil $S_{\text{scission}}$ (proportionnel à la distance moyenne $\overline{V}$ entre deux points successifs du contour), le segment $[V(i)V(i+1)]$ est découpé en trois nouveaux segments de longueurs égales. Ce processus est illustré par la figure 7.2. Soient $V(i_1)$ et $V(i_2)$ les deux points intermédiaires situés entre $V(i)$ et $V(i+1)$. Les trois segments créés sont $[V(i)V(i_1)]$, $[V(i_1)V(i_2)]$, et $[V(i_2)V(i+1)]$. Supposons que la division du contour soit intervenue sur les segments $[V(i)V(i+1)]$, $[V(j)V(j+1)]$, et $[V(k)V(k+1)]$ avec $i < j < k \pmod m$ ; le *snake* sera alors divisé en définissant trois contours, qui contiendront respectivement les ensembles $\Xi$ de segments suivants :

$$\{[V(i)V(i_1)], [V(i_1)V(k_2)], [V(k_2)V(k+1)], [V(k+1)V(k+2)], \ldots, [V(i-1)V(i)]\} \tag{7.15}$$

$$\{[V(j)V(j_1)], [V(j_1)V(i_2)], [V(i_2)V(i+1)], [V(i+1)V(i+2)], \ldots, [V(j-1)V(j)]\} \tag{7.16}$$

$$\{[V(k)V(k_1)], [V(k_1)V(j_2)], [V(j_2)V(j+1)], [V(j+1)V(j+2)], \ldots, [V(k-1)V(k)]\} \tag{7.17}$$

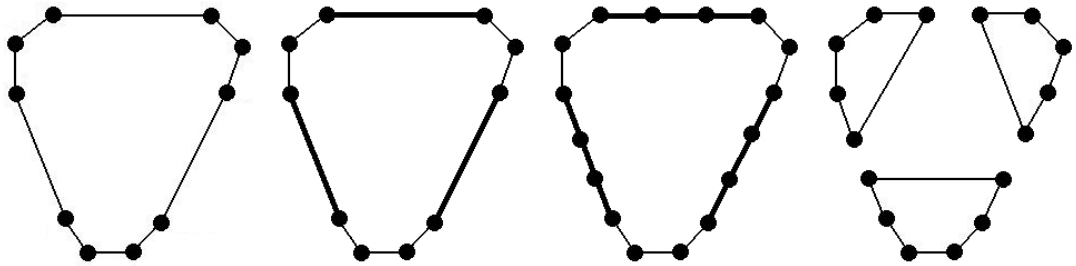

FIG. 7.2 – Processus de scission (de gauche à droite) : le contour initial est analysé pour déterminer les segments à diviser, puis pour chaque segment concerné, deux nouveaux points sont créés et finalement, les ensembles de segments sont réorganisés pour créer les contours finaux.

L'étape de scission permet de définir à partir d'un *snake* initial, plusieurs nouveaux contours. Mais l'ensemble des nouveaux contours peut contenir des *snakes* qui se sont fixés sur des pixels correspondant au bruit ou à l'arrière-plan. L'étape de scission est donc suivie par une étape de décision dont le but est de déterminer quels sont les contours intéressants parmi ceux créés. En nous basant sur la taille et la forme des nouveaux contours, nous décidons de supprimer les contours vérifiant l'une des caractéristiques suivantes :

$(\mathcal{Q}_1)$ : le contour $V$ consiste en un seul point,

$(\mathcal{Q}_2)$ : le contour $V$ consiste en une ligne,

$(\mathcal{Q}_3)$ : la surface du contour $V$ est très faible,

$(\mathcal{Q}_4)$ : la surface du contour $V$ est très importante.

qui peuvent se simplifier en une caractéristique relative à l'aire du contour $V$ :

$(\mathcal{Q})$ : l'aire délimitée par le contour $V$ notée $\mathrm{aire}(V)$ n'appartient pas à l'intervalle de définition d'un *snake* valide, soit $\mathrm{aire}(V) \notin [s_{\text{aire minimale}}, s_{\text{aire maximale}}]$ .

Nous avons décrit ici comment une étape de scission du contour actif permet de gérer les changements de topologie, d'augmenter la robustesse du suivi, et d'assurer un suivi simultané de plusieurs objets.

Dans la section suivante, nous présentons comment une analyse multirésolution des trames de la séquence vidéo peut être effectuée pour limiter le temps de calcul nécessaire à l'algorithme de suivi par contours actifs.

## 7.5 Analyse multirésolution des images

Afin de limiter le temps de calcul, la méthode proposée dans la section 7.3 peut être adaptée pour analyser les trames de la séquence vidéo selon une approche multirésolution [Lef02d].

L'analyse multirésolution n'est pas effectuée jusqu'à la résolution originale mais s'arrête lorsqu'un critère d'arrêt robuste est vérifié. De plus, nous proposons une adaptation automatique de certains des paramètres du modèle à la résolution à laquelle l'image est analysée.

### 7.5.1 Un processus multirésolution incomplet basé sur un critère d'arrêt robuste

Plusieurs auteurs ont proposé de modéliser les contours actifs selon un cadre multirésolution [Gra96, Ler96, Ray01]. L'évolution du *snake* est alors effectuée selon une approche *coarse-to-fine*. Le *snake* est tout d'abord déformé à une résolution grossière $r_{\text{max}}$, puis le résultat obtenu est utilisé comme le *snake* initial déformé alors à une résolution plus fine (égale à $r_{\text{max}} - 1$). Ce processus est répété jusqu'à ce que la résolution courante soit la résolution originale ($r = 0$).

Le cadre multirésolution est utilisé ici pour l'image et non pas pour le *snake*. Chaque image de la séquence vidéo est analysée à différentes résolutions, en commençant par la résolution la plus faible, c'est-à-dire $r = r_{\text{max}}$. Si la méthode précédente ne permet pas d'obtenir un contour acceptable relativement au critère de décision $\mathcal{Q}$, l'image est alors analysée à une résolution plus fine, soit $r \leftarrow r - 1$. La taille de l'image à la nouvelle résolution est $p^2$ fois supérieure à celle de la résolution précédente.

La définition et l'utilisation d'un critère d'arrêt lié à la qualité des résultats obtenus à une résolution donnée, limitent ici le nombre de résolutions analysées. Ce choix est particulièrement intéressant lorsque le contour obtenu à une résolution faible est suffisant pour traiter correctement les trames suivantes.

FIG. 7.3 – Représentation d'une image à différentes résolutions, depuis $r = 0$ (gauche) jusqu'à $r = 5$ (droite). Les images ont été rééchantillonnées pour observer les différences de détail.

La figure 7.3 montre différentes représentations d'une image à différentes résolutions. L'algorithme proposé ici est capable de traiter des images à différentes résolutions, depuis la résolution originale $r = 0$ jusqu'à la résolution la plus faible $r_{\text{max}} = 5$. Pour la résolution la plus faible, la taille de l'image a été réduite d'un facteur $2^{5\times 2} = 1024$. En moyenne, le suivi est effectué correctement tout au long de la séquence si $r \leq 3$, sur des images réduites d'un facteur

$2^{3\times 2} = 64$. Cependant il est possible d'obtenir dans certains cas des résultats corrects avec des résolutions plus grossières $3 < r \leq 5$.

### 7.5.2 Robustesse des paramètres vis-à-vis des changements de résolution

Afin d'assurer la robustesse de l'algorithme face aux changements de résolution, on a fait dépendre certains paramètres de la résolution de l'image. D'autres, au contraire, restent fixes tout au long du processus d'analyse multirésolution. Ainsi, les coefficients $\alpha$ pondérant les énergies ainsi que la taille $\Delta_s$ du voisinage ne dépendent pas du niveau de résolution. Nous décrivons ici les différents paramètres évoluant selon la résolution, ainsi que les fonctions d'évolution associées.

La taille $X_{V_0} \times Y_{V_0}$ du rectangle pour le *snake* initial $V_0$ ne doit évidemment pas être fixe, puisqu'une diminution de résolution s'accompagne d'une diminution de la taille des objets dans l'image. En conservant les notations utilisées dans ce mémoire, et en notant $\underset{0}{X_{V_0}} \times \underset{0}{Y_{V_0}}$ la taille du rectangle à la résolution originale $r = 0$, on a :

$$\underset{r}{X_{V_0}} = \frac{\underset{0}{X_{V_0}}}{2^r} \tag{7.18}$$

$$\underset{r}{Y_{V_0}} = \frac{\underset{0}{Y_{V_0}}}{2^r} \tag{7.19}$$

La même fonction d'évolution s'applique pour le nombre $m$ de points du *snake*, lui aussi dépendant du nombre de pixels dans l'image :

$$\underset{r}{m} = \frac{m}{2^r} \tag{7.20}$$

Puisque l'on considère un voisinage $\Delta_s$ constant alors que la taille de l'espace des données (c'est-à-dire le nombre de pixels dans l'image) est variable, le processus de déformation convergera plus ou moins rapidement vers la solution selon la résolution utilisée. Le nombre $\Gamma$ d'itérations peut alors être réduit ou augmenté lorsque la résolution change : la valeur du coefficient $\underset{r}{\Gamma}$ est donc variable selon la résolution donnée $r$.

Finalement, nous observons que les propriétés du calcul du gradient ne sont pas indépendantes de la résolution. En effet, le moyennage successif des pixels pour obtenir une image à plus faible résolution, entraîne également un lissage de l'image. Les pixels sont alors caractérisés par des modules de gradient plus faibles. Le seuil $S_{\text{grad}}$ utilisé pour la comparaison avec les

modules de gradient des pixels doit être adapté à la résolution utilisée.

La méthode de suivi décrite dans la section 7.3, avec ses extensions liées au changement de topologie et à l'analyse multirésolution des images, est résumée par l'algorithme 4.

La méthode de suivi présentée dans ce chapitre a été testée sur différentes séquences vidéo. Les résultats obtenus, permettant de caractériser son efficacité, sont décrits dans la section suivante.

## 7.6 Résultats

Nous avons présenté précédemment un modèle de contour actif permettant le suivi rapide d'objets non-rigides en mouvement dans des images couleur acquises avec une caméra en mouvement. Nous avons aussi introduit différentes extensions qui consistent en la gestion des changements de topologie et en l'accéleration du traitement par une analyse multirésolution des images. Dans cette section, nous décrirons tout d'abord les différents paramètres utilisés dans la méthode et discuterons la manière de les régler. Ensuite nous présenterons quelques résultats obtenus avec ces paramètres sur des séquences vidéo de matchs de football, afin d'illustrer la méthode de suivi décrite précédemment. Finalement nous commenterons les temps de calcul obtenus.

La méthode proposée a été testée sur des séquences vidéo représentant des scènes d'extérieur caractérisées par un arrière-plan relativement uniforme. La taille des images couleur est égale à $384 \times 284$ pixels et la fréquence d'acquisition est de 15 Hz. Afin de limiter le temps de calcul, nous n'effectuons pas ici de conversion depuis l'espace couleur RGB vers un autre espace de représentation.

Le *snake* est composé initialement de $m$ points à la résolution originale ($r = 0$). Ce paramètre $m$ a une influence directe sur le résultat obtenu d'une part et le temps de calcul nécessaire d'autre part. Lorsque l'application ne nécessite que la position de l'objet, et que la précision de la forme de l'objet n'est pas importante, le nombre de points peut être réduit, ce qui conduit à un temps de calcul moindre.

A la résolution initiale ($r = 0$), le nombre maximum d'itérations est égal à 30. Cependant, le contour converge la plupart du temps avant ce nombre d'itérations. En effet, la convergence a été obtenue pour plus de 70 % des images. La taille $\Delta_s$ du voisinage utilisé pour la recherche locale du point minimisant la fonctionnelle d'énergie est égale à $5 \times 5$ pixels. Afin de diminuer une nouvelle fois le temps de calcul, il est possible de réduire les valeurs de $\Gamma$ et de $\Delta_s$.

**Algorithme 4:** Suivi d'objet non-rigide par contours actifs.

**Entrée** : Plan d'une séquence vidéo composé de $T$ trames codées en RGB
Position initiale de l'objet suivi modélisé par un *snake* $V_0$
Paramètres $m$, $\Gamma$, $\Delta_s$, coefficients $\alpha$, $r_{\text{max}}$, et $p^2$

**Sortie** : Position de l'objet suivi dans chaque trame modélisé par un *snake* $V$

*Initialisation*
$V_{\text{initial}} \leftarrow V_0$
*Parcours des images du plan*
**pour** $t \leftarrow 1$ **à** $T$ **faire**
  *Analyse multirésolution*
  $r \leftarrow r_{\text{max}} + 1$
  **répéter**
    $r \leftarrow r - 1$
    Détermination des extrémités du *snake* initial $V_{\text{initial}}$
    Création du *snake* $V$ le long d'un rectangle proportionnel aux extrémités de $V_{\text{initial}}$
    Calcul du gradient et de $\|\nabla I\|$ sur une zone limitée autour de $V$
    $\gamma \leftarrow 0$
    *Déformation itérative du* snake
    **répéter**
      Calcul de la distance moyenne $\overline{V}$
      *Déplacement de chaque point du* snake
      **pour** $i \leftarrow 1$ **à** $m$ **faire**
        **pour chaque** point $M(j)$ voisin de $V(i)$ vérifiant $M(j) \in \mathcal{V}(V(i))$ **faire**
          Calcul de l'énergie $E(M(j))$ par l'équation (7.4)
        *Recherche du point voisin minimisant l'énergie*
        $j_{\text{min}} \leftarrow \underset{M(j)\in\mathcal{V}(V(i))}{\arg\min} (E(M(j)))$
        **si** $M(j_{\text{min}}) \neq V(i)$ **alors** $V(i) \leftarrow M(j_{\text{min}})$
      $V_\gamma \leftarrow V$
      $\gamma \leftarrow \gamma + 1$
    **jusqu'à** $\gamma > \Gamma$ **ou** $V_\gamma = V_{\gamma-1}$
    *Processus de changement de topologie*
    Recherche des segments $[V(i)V(i+1)]$ vérifiant $V(i)V(i+1) > S_{\text{scission}}$
    Découpage des segments $[V(i)V(i+1)]$ et génération des ensembles $\Xi$
    Création des nouveaux *snakes* contenant les ensembles $\Xi$
    **pour chaque** nouveau *snake* $V'$ créé **faire**
      **si** $V'$ n'est pas valide **alors** Supprimer $V'$
  **jusqu'à** $\exists V'$ **ou** $r = 0$
  Afficher le contour $V$
  $V_{\text{initial}} \leftarrow V$

Les coefficients utilisés pour pondérer les différentes énergies ont tous été fixés à 1. Les résultats obtenus sont satisfaisants tout en limitant le nombre d'opérations (les multiplications des coefficients et de l'énergie correspondante ne sont pas à effectuer). Le seuil $S_{\text{grad}}$ utilisé pour le calcul du gradient est quant à lui fixé à 500 pour la résolution originale.

Le tableau 7.1 récapitule les différents paramètres nécessaires à la méthode et les valeurs que nous avons utilisées pour ces paramètres.

| Paramètre | Description | Valeur |
|---|---|---|
| $m$ | Nombre de points du *snake* (pour $r = 0$) | 16 |
| $\Gamma$ | Nombre maximal d'itérations lors de la déformation (pour $r = 0$) | 30 |
| $S_{\text{grad}}$ | Seuil utilisé dans le calcul du gradient (pour $r = 0$) | 500 |
| $\Delta_s$ | Taille du voisinage analysé pour le calcul local de l'énergie | $5 \times 5$ |
| $\alpha_{\text{int}}$ | Coefficient de l'énergie interne | 1 |
| $\alpha_{\text{ext}}$ | Coefficient de l'énergie externe | 1 |
| $\alpha_{\text{cont}}$ | Coefficient de l'énergie continuité | 1 |
| $\alpha_{\text{cour}}$ | Coefficient de l'énergie courbure | 1 |
| $\alpha_{\text{ball}}$ | Coefficient de l'énergie ballon | 1 |
| $\alpha_{\text{grad}}$ | Coefficient de l'énergie gradient | 1 |
| $\alpha_{\text{coul}}$ | Coefficient de l'énergie couleur | 1 |
| $r_{\text{max}}$ | Nombre de couches de la pyramide | 5 |
| $p^2$ | Nombre de pixels utilisés à la résolution $r$ pour générer un pixel à la résolution $r + 1$ | 4 |

TAB. 7.1 – Paramètres utilisés lors du suivi d'objet non-rigide par contours actifs.

La figure 7.4 illustre le suivi d'un objet (joueur de football) au cours d'une séquence contenant 100 trames. Dans ce cas, le suivi a été effectué à la résolution originale. L'algorithme permet de suivre un objet mobile dans un environement également mobile, sans effectuer d'estimation de mouvement de l'objet ni de compensation de mouvement de la caméra.

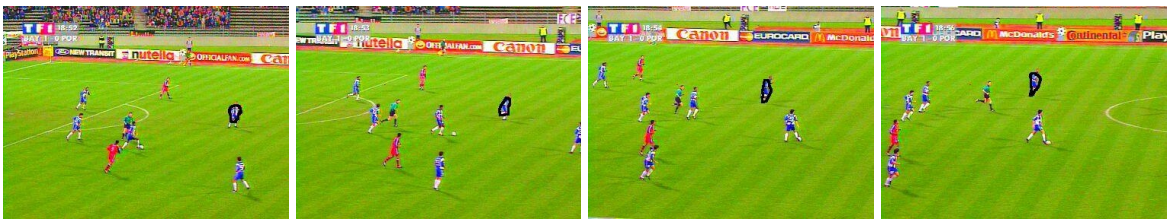

FIG. 7.4 – Suivi d'un objet non-rigide dans une séquence de 100 trames.

La figure 7.5 illustre le principe de scission. Ainsi, il est possible de suivre indépendamment

les différents objets présents dans la scène. La répartition précise des différents points du *snake* est donnée dans la figure 7.6. Nous pouvons ainsi observer l'apport du principe de scission. Cependant, la sensibilité du modèle de contour actif à un arrière-plan complexe (contenant des pixels caractérisés par des valeurs de gradient élevées) reste forte.

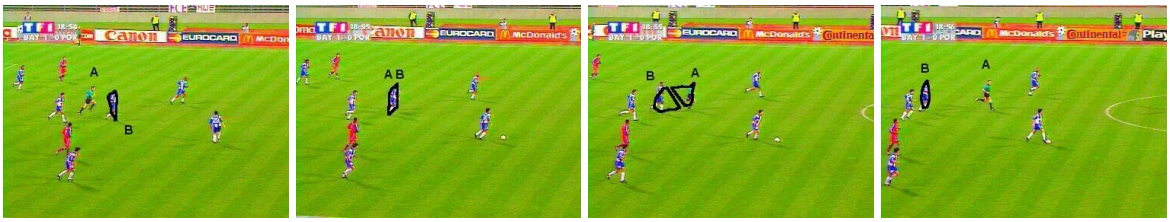


FIG. 7.5 – Intérêt de l'étape de scission dans le cas d'objets proches. Les deux objets mobiles ont des trajectoires croisées (l'occlusion intervient dans la seconde image). L'étape de scission permet d'obtenir des contours corrects après l'occlusion (image de droite).

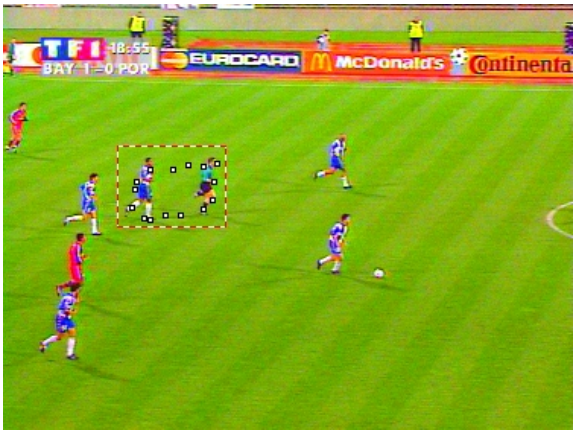


FIG. 7.6 – Répartion précise des points du *snake* avant l'exécution de la procédure de scission.

L'analyse multirésolution décrite dans la section 7.5 est illustrée dans la figure 7.7. La résolution utilisée est égale à 4, ce qui résulte en une taille d'image 256 fois inférieure à la taille originale. On peut observer le manque de précision de la forme du *snake*, qui est due à la faible résolution à laquelle l'image est analysée.

Le temps de calcul de la méthode de suivi proposée est d'environ 35 millisecondes par image. L'architecture utilisée est toujours un PC Pentium 4 cadencé à 1700 MHz. Il est donc possible de suivre en temps réel des objets en mouvement dans une scène en mouvement, en utilisant des informations de gradient et de couleur.

Contrairement aux justifications théoriques, l'analyse monorésolution s'effectue quasiment aussi rapidement que l'analyse multirésolution (l'écart n'est que de quelques millisecondes).

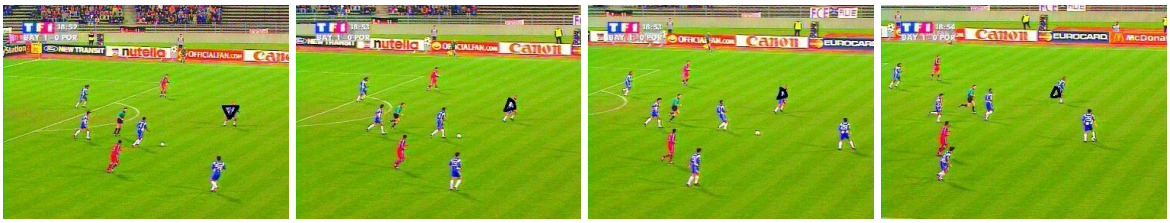

FIG. 7.7 – Suivi d'objet non-rigide dans un cadre multirésolution. Les images ont été analysées à une résolution 256 fois inférieure à la résolution originale.

Cela est dû en partie au temps nécessaire pour créer la représentation multirésolution de l'image. Ce traitement peut être évité si l'on analyse des séquences vidéo compressées (comme celles au format MPEG ou M-JPEG). Dans ce cas, les coefficients DC peuvent être utilisés directement (sans décompression) pour obtenir des images à faible résolution. La décompression complète des trames de la séquence vidéo n'est alors pas nécessaire si l'algorithme converge à faible résolution. Cependant, après avoir analysé le temps nécessaire pour construire la représentation multirésolution de l'image, il s'avère que celui-ci reste faible (de l'ordre de quelques millisecondes). L'absence de différence dans les temps de calcul des images monorésolution et multirésolution se justifie donc par une raison différente.

On peut remarquer que l'observation des temps de calcul similaires pour les analyses monorésolution et multirésolution est due au fait que l'algorithme de suivi a déjà été optimisé (par exemple le gradient n'est calculé qu'une seule fois par image, et sur une zone limitée autour du *snake* initial) et est déjà caractérisé par une complexité algorithmique faible.

## 7.7 Conclusion

Dans ce chapitre, nous avons traité le problème du suivi d'objet non-rigide. Ce type d'objet ne peut être suivi avec les méthodes dédiées aux objets rigides décrites dans les chapitres 5 et 6, et nécessite d'utiliser un outil adapté. C'est pourquoi notre choix s'est porté sur les *snakes* introduits dans [Kas88]. Quoiqu'efficaces, les *snakes* présentent différents inconvénients liés à l'initialisation, aux nombreux paramètres, au changement possible de topologie, et au temps de calcul. Dans ce chapitre, notre contribution a donc consisté à résoudre les différents problèmes posés par les *snakes*, afin d'obtenir une méthode de suivi d'objet non-rigide par contours actifs respectant les différentes contraintes énoncées précédemment.

Afin de limiter la sensibilité du modèle à l'initialisation, nous avons proposé une approche originale qui consiste à intialiser le *snake* sous la forme d'un rectangle, puis à le réduire au-

tour de l'objet suivi. De ce fait, le suivi est robuste aux conditions d'initialisation. Pour gérer les changements de topologie, nous avons introduit un processus de scission, ce qui permet de suivre des objets même s'ils sont proches les uns des autres. Finalement, la contrainte la plus difficile à prendre en compte dans le cas des *snakes* est $\mathcal{C}_{\text{rapidité}}$. Pour résoudre le problème tout en prenant en compte cette contrainte, nous avons combiné différentes techniques d'optimisation : calcul du gradient une seule fois par image et sur une zone limitée de celle-ci, absence de filtrage ou de prétraitement global, absence d'estimation de mouvement de l'objet suivi, absence d'estimation et de compensation du mouvement de la caméra, analyse multirésolution des images, *etc.*.

Dans cette partie nous avons traité le problème du suivi d'objet en s'intéressant à trois classes d'objet : les objets rigides qui peuvent être appris (chapitre 5), les objets rigides de petite taille qui ne peuvent pas être appris (chapitre 6), et enfin les objets non-rigides (chapitre 7). Ces trois types d'objets peuvent respectivement être suivis à l'aide de méthodes basées sur les chaînes de Markov cachées ou le *template matching*, la recherche des détails dans l'image, et les contours actifs. Après avoir effectué un découpage de la séquence en plans (partie I), l'analyse combinée de l'arrière-plan (partie II) et des objets (partie III) permet de fournir les informations nécessaires à l'interprétation du contenu de la séquence (c.f. figure 1 page 4). A l'aide de cette interprétation, il est ensuite possible de détecter des événements prédéfinis. Les différents outils génériques décrits précédemment peuvent donc être intégrés dans des systèmes spécifiques d'indexation vidéo. Nous présentons dans la partie suivante trois systèmes différents d'indexation vidéo, basés sur une source vidéo contrôlée ou non.

# Quatrième partie

# Applications

Dans les parties précédentes, nous avons identifié les différents problèmes à résoudre pour mettre en place un système d'indexation vidéo. Pour chacun des problèmes présentés, nous avons donné quelques points d'entrée dans la littérature et proposé des méthodes performantes permettant d'obtenir le résultat souhaité, en prenant en compte les différentes contraintes énoncées. Ces outils se caractérisent notamment par leur généricité : ils peuvent être utilisés dans différentes applications et ne dépendent pas d'un contexte donné. De même, l'architecture proposée peut être utilisée sans modification pour réaliser différents systèmes d'indexation spécifiques, chacun répondant à un besoin précis. Nos travaux peuvent donc être comparés à ceux de Amer [Ame02], Carrive [Car00], ou encore Haering [Hae00] qui proposent également des architectures et des outils pour la mise en place de systèmes permettant la détection d'événements dans des séquences vidéo.

L'architecture et les outils proposés précédemment ont été validés expérimentalement. Pour cela, nous avons réalisé plusieurs systèmes d'indexation vidéo que nous avons regroupés en deux catégories. La première catégorie, présentée dans le chapitre 8 concerne les systèmes analysant des séquences vidéo acquises par une source non-contrôlée (typiquement une séquence télévisée). Nous illustrerons ce cas par un système temps réel de détection des buts inscrits dans un match de football. La seconde catégorie, illustrée dans le chapitre 9, correspond au cas d'une source contrôlée, comme par exemple une caméra numérique ou une webcam dont on connait et maîtrise les différents réglages. Ce cas sera illustré par deux applications, faisant toutes deux suite aux demandes de l'entreprise Atos Origin ayant financé cette thèse au travers d'un contrat CIFRE. La première application est liée à l'obtention de statistiques relatives à la fréquentation d'un site. Le but de cette application est d'évaluer l'interêt commercial du site en question afin de mettre en place (ou non) un distributeur automatique de billets. La seconde application concerne la vidéosurveillance d'une salle informatique sécurisée. Le problème à résoudre ici consiste en la détection de deux événements prédéfinis : "le nombre de personnes entrant sur le site est supérieur au nombre autorisé", et "la zone d'intervention est différente de celle prévue".

# Chapitre 8

# Cas d'une source vidéo non-contrôlée : Détection de buts dans des matchs de football

La première application que nous présentons dans cette partie illustre le cas d'une séquence vidéo non-contrôlée, c'est-à-dire acquise à l'aide d'une ou plusieurs sources dont les caractéristiques nous sont inconnues. Afin d'illustrer ce cas de figure, nous avons choisi d'analyser des séquences vidéo issues de retransmissions télévisées. Plus précisément, l'objectif ici consiste en la détection, en temps réel, de courtes périodes du film pendant lesquelles un but est inscrit lors d'un match de football. Le système d'indexation vidéo nécessaire s'inspire donc directement de l'architecture décrite au début de ce mémoire et utilise les différents outils qui ont été présentés.

Dans ce chapitre, nous décrirons tout d'abord plus précisément le problème à résoudre. L'analyse de séquences vidéo de football, ou plus généralement de sport, a suscité récemment l'intérêt de nombre de chercheurs de la communauté. C'est pourquoi nous dresserons un panorama des systèmes décrits dans la littérature. Ce panorama nous permettra notamment de situer notre contribution vis-à-vis des travaux existants. Nous introduirons alors l'architecture du système d'indexation vidéo que nous proposons pour résoudre le problème donné. Les différents outils d'analyse d'images utilisés ici ont déjà fait l'objet d'une description dans les chapitres précédents. Par ailleurs, les séquences vidéo télévisées comportent en outre une bande son, qui a également été analysée. Cette analyse fera l'objet d'une section particulière.

## 8.1 Problématique

Le sport occupe une place de plus en plus importante dans les média, notamment la télévision. La durée et la fréquence des retransmissions télévisées d'événements sportifs sont en constante augmentation. Cependant, le public se considère souvent comme "noyé" sous le nombre d'images diffusées, puisque qu'il n'est intéressé que par une faible portion de ces images. Ainsi, de nombreux téléspectateurs souhaiteraient ne voir que les moments importants (ou moments clé) d'une rencontre sportive. D'où la diffusion quasi-systématique, à la fin de la rencontre concernée, d'un résumé constitué des moments principaux. Dans le cas d'un match de football, ces moments sont principalement ceux des buts inscrits lors de la rencontre. Ce résumé n'est disponible qu'à la fin de la rencontre et il n'est donc pas permis aux intéressés de visualiser les moments importants à l'instant où ceux-ci se déroulent. Pourtant, les technologies de diffusion de contenu vidéo permettraient à l'heure actuelle, ou pour certaines permettront bientôt, de diffuser aux utilisateurs, et sur différents canaux, les extraits les plus pertinents d'une retransmission télévisée. Ainsi les destinataires de ces messages vidéo sont ou seront capables de les recevoir *via* la télévision, Internet, la téléphonie mobile (avec l'UMTS), *etc*. Certaines propositions commerciales sont faites actuellement dans ce sens. Mais la sélection des extraits est effectuée manuellement. Le problème que nous traitons ici consiste donc en la détection, en temps réel, d'événements prédéfinis (comme des buts inscrits lors de matchs de football). Ce problème résolu, il serait alors possible, grâce aux technologies énoncées précédemment, de transmettre les extraits pertinents (ici les portions de séquences vidéo correspondant aux buts) aux personnes interessées. La figure 8.1 illustre le principe exposé ici.

L'application que nous considérons dans ce chapitre consiste donc en la détection, en temps réel, des buts inscrits dans des matchs de football, par analyse des séquences vidéo issues des retransmissions télévisées. Le problème s'inscrit dans l'analyse de séquences vidéo sportives, champ d'application ayant fait l'objet de différents travaux publiés dans la littérature. Nous dressons donc un panorama de ces travaux, afin notamment de mieux situer notre contribution.

## 8.2 Panorama de l'analyse de séquences vidéo sportives

De nombreux scientifiques se sont intéressés à la compréhension et l'interprétation automatique d'événements sportifs. Pour cela, une analyse des images et/ou du son est le plus souvent nécessaire, les données à traiter pouvant être issues de retransmissions télévisées ou d'acquisitions spécifiques. Nous dressons ici un panorama des différents travaux dans la littérature, en détaillant successivement les approches générales (notamment le projet européen ASSAVID),

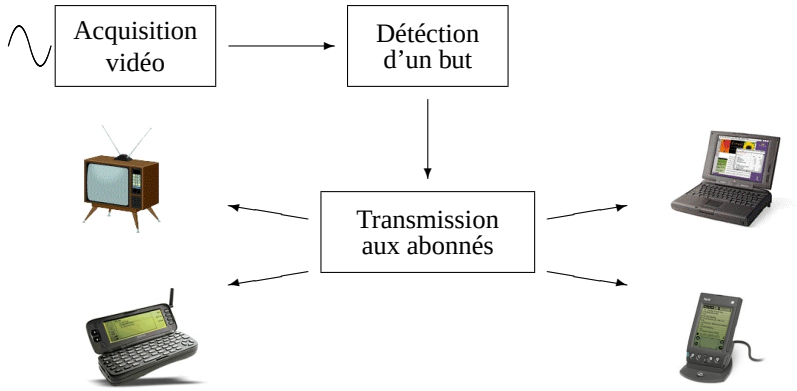


FIG. 8.1 – Principe de la détection et de la transmission d'extraits pertinents à différents utilisateurs.

puis celles dédiées au football, et enfin celles relatives à d'autres sports de balle.

### 8.2.1 Approches générales

L'analyse de séquences vidéo sportives fait l'objet de plus en plus d'études. Ainsi, le projet européen ASSAVID [Ass00] a pour but d'annoter automatiquement des séquences vidéo sportives. Dans ce cadre, de nombreux travaux ont été publiés [Che02b, Kou02, Mes02, Che02a, Ass01, Ass02, Kit01, Che02c, Che01b]. La périodicité des mouvements présents dans la séquence vidéo est analysée pour déterminer le type de contenu des images [Che02b] : sprint, aviron, course de fond, *etc.*. Cette classification des images en différents sports peut également être obtenue par analyse couleur [Kou02]. Il est aussi possible de considérer les informations relatives à la texture, voire de combiner différentes caractéristiques, pour résoudre ce problème [Mes02]. L'annotation des séquences peut provenir d'une reconnaissance du texte incrusté dans les images [Che02a]. Le système global d'annotation automatique a notamment été présenté dans [Ass01, Ass02].

Agbinya et Rees [Agb99, Ree98] ont également proposé un système d'annotation de séquences vidéo sportives. L'utilisation de différentes caractéristiques (telles que la couleur ou le mouvement) au sein d'un même outil permet un suivi d'objets multiples. Celui-ci fournit quant à lui les informations nécessaires pour aider un utilisateur à annoter semi-manuellement les séquences vidéo.

Des travaux ont également été effectués pour détecter les plans "ralentis" dans des retransmissions télévisées. Kobla *et al.* [Kob99, Kob00] détectent ce type de plan en utilisant les vecteurs de mouvement contenus dans les séquences MPEG. Leur but est de déterminer si une séquence vidéo donnée représente une retransmission sportive ou non. Pan *et al.* [Pan01] utilisent quant à eux les ralentis pour localiser les moments intéressants dans une séquence sportive. Leur approche est basée sur des chaînes de Markov cachées en considérant des caractéristiques liées à la différence pixel-à-pixel entre deux trames successives et à la distribution des couleurs dans les trames.

Les techniques précédentes sont adaptées à différents types de sport et n'en concernent pas un en particulier. Nous présentons donc maintenant les techniques publiées dans la littérature et spécifiques à un sport donné, en traitant le cas du football dans un premier temps, puis celui plus général des autres sports de balle.

### 8.2.2 Le domaine du football

Le football est très certainement le sport faisant l'objet du plus grand nombre d'études dans le domaine du traitement de l'information [Pen98]. Des laboratoires tels que celui d'Intel ont réalisé des études [Haz01, Sal01] pour montrer les débouchés commerciaux possibles dans ce domaine.

Parmi les premiers travaux publiés dans la littérature, le système proposé par Gong *et al.* [Gon95a, Gon95b] permet de classifier les différentes périodes de jeu. Pour cela, le terrain est extrait par recherche de la plus grande composante connexe verte puis les lignes blanches sont obtenues par détection de contours et appariement à un modèle. Une technique de *block matching* permet alors de calculer les mouvements des différents blocs de pixels tandis que des informations couleur sont utilisées pour suivre le ballon et les joueurs. De manière indépendante, mais dans la même conférence, Yow *et al.* [Yow95] ont présenté un système similaire. Le mouvement de la caméra est estimé puis compensé, ce qui permet par la suite d'obtenir une représentation panoramique de la scène (appelée mosaïque). Le ballon est suivi par *template matching* tandis que les poteaux sont localisés par détection des lignes verticales. Xu *et al.* [Xu01b, Xu01a] ont proposé plus récemment un système dont le but est de fournir des informations quant à la structure d'un match de football. En déterminant dans l'image la taille de la pelouse et son orientation, il est possible de reconnaître le type de plan (proche ou éloigné) et de déterminer s'il s'agit d'un moment de jeu ou non. Kim et Yang [Kim01a] proposent d'utiliser des réseaux de neurones pour reconnaître les différentes actions se déroulant pendant un match. Une première étape leur permet d'extraire les lignes et les joueurs à l'aide d'un en-

semble d'heuristiques. Les positions des joueurs sont alors replacées dans un modèle du terrain *via* l'utilisation de mosaïques. La classification par le réseau de neurones des différentes actions est basée sur les positions des différents joueurs dans la scène.

Les chercheurs du laboratoire Postech ont également abordé ce sujet. Ainsi, une méthode permettant d'obtenir les trajectoires des différents joueurs est proposée dans [Seo97]. La pelouse est extraite par combinaison des informations de couleur et de connexité. Les joueurs sont suivis par *template matching* et filtre de Kalman et sont différenciés par analyse des distributions verticales des différentes composantes couleur. Le ballon est quant à lui recherché localement autour des joueurs. La création d'un modèle du terrain et d'une mosaïque de la scène permet finalement d'obtenir les positions (et les trajectoires) des différents joueurs dans la scène. Les techniques de mosaïques ont notamment été étudiées dans [Kim00a, Kim01b] tandis que les autres travaux se sont focalisés sur l'obtention de la position 3-D du ballon, soit à l'aide d'une caméra [Kim98a,Kim98b], soit à l'aide de plusieurs caméras [Kim00b]. Enfin, plus récemment, les occlusions ont été prises en compte dans le processus de suivi [Ok02].

Vandenbroucke [Van00] a proposé une méthode de segmentation et de classification couleur qu'il a ensuite combinée avec des *snakes* pour suivre les différents joueurs dans la séquence. Misu *et al.* [Mis02] proposent une technique de suivi en combinant différentes informations liées à la couleur, la texture, et au mouvement des objets. Le processus de fusion des différentes caractéristiques permet notamment de gérer les phénomènes d'occlusion. Zhao et Nevatia [Zha02] utilisent un modèle 3-D pour suivre les joueurs de football et déterminer leur état (arrêt, marche, course, *etc*). A l'aide du modèle 3-D, il est également possible de générer des vues virtuelles des joueurs suivis. Figueroa [Fig98] considère le cas de caméras fixes placées de manière à couvrir toutes les positions possibles des joueurs sur le terrain. Une étape de segmentation lui permet d'extraire les différents joueurs de l'image. Le suivi est ensuite effectué entre chaque image en modélisant chaque joueur dans une image donnée par un nœud dans un graphe. Le problème revient alors à rechercher le plus court chemin entre deux noeuds dans ce graphe.

Taki *et al.* [Tak96, Tak99, Tak00], considérant également un ensemble de caméras statiques disposées tout autour du terrain, extraient tout d'abord les lignes et les joueurs. Le mouvement de chaque joueur est ensuite utilisé pour générer une région, appelée *dominant region*, pour chaque joueur. Cette région peut être considérée comme la zone d'influence d'un joueur, et représente toutes les positions que ce joueur est susceptible d'atteindre d'abord (avant les autres joueurs). La segmentation du terrain en zones d'influence obtenue peut être alors utilisée pour améliorer les stratégies des équipes. Les chercheurs d'Osaka ont développé une méthode permettant de suivre les joueurs par analyse couleur (même en cas d'occlusion) ainsi que le

ballon et de calculer leurs positions respectives dans l'espace 3-D à l'aide d'un modèle du terrain. Ils ont considéré successivement le cas d'une caméra statique [Ohn99, Ohn00] puis dynamique [Yam02].

Reid s'est quant à lui focalisé sur le suivi du ballon et l'estimation de la position 3-D de celui-ci. Différentes approches ont été proposées, selon que l'on dispose de deux séquences vidéo [Rei96] ou bien d'une seule [Rei98] en utilisant dans ce dernier cas l'ombre de l'objet suivi. Tsuda *et al.* [Tsu01] proposent un système de vision active où le but est de disposer d'une caméra capable de suivre le ballon de manière autonome. Les chercheurs de l'université de Bari proposent de détecter le ballon en utilisant soit une transformée de Hough pour les formes circulaires [D'O02], soit un classifieur SVM (*Support Vector Machine*) [Anc02]. La localisation du ballon dans l'espace peut quant à elle être obtenue en utilisant les positions des poteaux [Bra01]. Les mouvements de la caméra (translation, rotation, zoom) sont directement liés à la position 3-D estimée du ballon dans les images.

Ekin et Tekalp proposent de détecter des buts par un système à base de règles [Eki02] basées sur la comparaison des positions et des trajectoires des différents objets. Kurokawa *et al.* [Kur99] utilisent trois concepts différents (action, interaction, et événement) afin de retrouver des moments spécifiques durant un match de football. A partir d'une séquence vidéo acquise à l'aide d'une caméra statique, et d'un ensemble de connaissances *a priori* sur le football, des chercheurs de Saarbrücken proposent un système appelé SOCCER pour générer une description en langage naturel de chaque scène d'un match complétée d'une représentation graphique de l'action [And88, And94, RS88]. Le langage naturel peut également être transcrit oralement par synthèse vocale [And01]. Un autre système d'analyse, nommé ADMIRE, a été proposé dans [Wou98, dP97, Hee97]. Dans ce dernier, les auteurs insistent notamment sur le fait que la sémantique est associée aux caractéristiques afin d'effectuer une recherche basée sur des concepts et non sur des caractéristiques. Petkovic et Jonker proposent dans [Pet00] un modèle permettant de modéliser les événements dans des séquences vidéo et considèrent les quatre couches d'informations suivantes (du plus bas-niveau au plus haut-niveau) : les pixels, les caractéristiques, les objets, et les événements. Ainsi, il est possible de détecter des événements particuliers (comme des buts) en définissant les états et les interactions entre objets appropriés. Teraguchi *et al.* [Ter02] proposent un système pour l'indexation semi-manuelle d'événements dans des séquences vidéo de football. Celuic-ci permet notamment à un utilisateur expérimenté d'annoter une séquence vidéo d'une durée $D$ en un temps égal à $1,5D$.

Il est également possible de générer des séquences virtuelles, ou de synthèse. Nous pouvons citer ici les travaux de Carlson [Car98] ou ceux de Bebie et Bieri [Beb98, Beb00]. Ces derniers utilisent successivement des étapes de calibration des caméras (deux sont considérées), de suivi

des lignes, d'extraction de la zone de jeu, de reconstruction des trajectoires de la balle et des joueurs pour générer des vues virtuelles 3-D de la scène. Inamoto et Saito [Ina02], à partir d'une séquence vidéo acquise au moyen de deux caméras statiques, reconstruisent artificiellement une troisième vue associée à une caméra virtuelle. Plus précisément, ils effectuent une détection des différents objets (joueurs et ballon) dans chaque couple d'images et un appariement afin d'obtenir une correspondance géométrique. Les poteaux et la pelouse sont reconstruits avec une approche similaire, tandis que les gradins sont générés par une technique de mosaïque. Ohta présente dans [Oht02] un système permettant, à partir de plusieurs caméras placées autour d'un stade, de générer une vue 3-D et virtuelle de la scène analysée. Iwase *et al.* [Iwa95, Mat98] génèrent une vue de synthèse de la scène depuis n'importe quel joueur présent sur le terrain de football. Pour cela, les paramètres de la caméra sont tout d'abord extraits en se basant sur les lignes blanches du terrain, puis les positions réelles des joueurs sont obtenues. L'utilisateur peut finalement choisir un quelconque joueur sur le terrain et générer la vue virtuelle depuis ce joueur.

### 8.2.3 Et les autres sports de balle ?

Le football n'est cependant pas le seul sport de balle concerné par des travaux scientifiques de la communauté. Ainsi, les chercheurs de l'IRIT ont proposé un système [Gur96] permettant d'obtenir différentes informations concernant un match de rugby telles que les déplacements des différents joueurs à partir de séquences télévisées. Le système propose également des fonctionnalités de réalité augmentée.

Pers et Kovacic s'intéressent au handball et présentent dans [Per00, Per01] une méthode de suivi des joueurs à l'aide de deux caméras fixes situées au-dessus du terrain. Ils insistent notamment sur l'évaluation de leur algorithme en collaboration avec la communauté scientifique sportive.

Le base-ball fait également l'objet de différentes études [Kaw98, Rui00]. Dans [Rui00], les auteurs n'utilisent que les données sonores (afin d'obtenir un traitement plus rapide) et combinent différents outils (tels qu'un classifieur Bayesien, les *Support Vector Machines*, et les K plus proches voisins) afin de localiser les moments importants. Sung et Chun [Sun02] appliquent une technique de reconnaissance de texte incrusté dans des retransmissions télévisées coréennes de base-ball. Ainsi, lorsqu'une vitesse de balle est affichée en miles par heure dans la séquence originale, elle est convertie en kilomètres par heure et est à son tour incrustée dans les images diffusées.

L'analyse des données audio est aussi principalement utilisée par Chang *et al.* [Cha96] pour

détecter certains types d'événements (ici des *touchdown*) dans des matchs de football américain. Les informations visuelles (détection des lignes et des poteaux) sont alors utilisées pour confirmer les résultats obtenus. L'analyse vidéo peut également être combinée au traitement du langage naturel pour interpréter le contenu de la séquence, comme le montrent les travaux de Lazarescu *et al.* [Laz98, Laz99], toujours dans le domaine du football américain. Miyauchi *et al.* [Miy02b] utilisent quant à eux le canal textuel (présent notamment dans les retransmissions télévisées sportives aux Etats-Unis et au Japon) pour localiser les événements importants, puis le canal audio pour effectuer une vérification, et enfin le canal vidéo pour associer l'événement à une séquence d'images donnée. Mais les travaux majeurs dans ce domaine restent ceux de Intille et Bobick [Int94, Int95, Bob95, Int97, Int99, Int01] qui vont du suivi des objets dans un monde restreint à la reconnaissance des actions dans un match par un système multi-agent.

D'autres chercheurs ont travaillé dans le domaine du basket-ball. Ainsi, Nepal *et al.* ont proposé dans [Nep01] un système capable de détecter les paniers inscrits dans un match de basket-ball. Tan *et al.* [Tan00, Sau97] détectent certains événements importants en calculant des caractéristiques (telles que le mouvement de la caméra) directement depuis la séquence vidéo compressée au format MPEG. Leur système, basé également sur des connaissances *a priori*, permet de plus de séparer les plans rapprochés des plans larges. Xu *et al.* [Xu02] reconnaissent les différentes séquences de jeu d'un match à l'aide de chaînes de Markov cachées (une chaîne pour chaque événement à reconnaître) en utilisant l'information de mouvement présente dans la séquence. Zhou *et al.* [Zho00b] proposent une approche à base de règles. Mann *et al.* [Man02] utilisent une approche par programmation dynamique pour obtenir la trajectoire du ballon.

Le tennis fait fait également l'objet d'un certain nombre d'études. Ainsi, Sudhir *et al.* [Sud98] définissent un modèle du court de tennis et effectuent un suivi des joueurs pour convertir les positions (qui sont des informations de bas-niveau) en données plus sémantiques correspondant aux différents événements détectés. En se basant sur la structure temporelle (connue *a priori*) d'une retransmission télévisée d'un match de tennis, Zhong et Chang [Zho01b, Zho01a] présentent un système capable de détecter les différentes périodes de jeu. Les scènes de service sont repérées à l'aide d'un filtrage adaptatif couleur, suivi d'une segmentation et d'une détection de contours. Les traitements suivants concernent le suivi des joueurs et l'analyse de leurs trajectoires. Les différentes étapes du processus étant effectuées directement sur les données compressées, le système fonctionne en temps réel. Les auteurs proposent d'utiliser le résultat pour effectuer une diffusion dont le débit dépend de l'interêt de la période de jeu [Cha01b]. Petkovic *et al.* [Pet01] font appel aux chaînes de Markov cachées pour détecter les frappes au cours du jeu. Le système proposé par Miyamori dans [Miy02a] permet de comprendre le comportement des joueurs au cours de la rencontre. Une première étape est nécessaire pour extraire

les lignes du court et les joueurs. L'extraction du point d'impact (lorsque la raquette frappe la balle) est ensuite obtenue par combinaison du suivi de la balle [Miy00] et de l'analyse de la piste audio. Il est finalement possible de connaître les différents gestes effectués par les joueurs (coup droit, revers, déplacement, smash, *etc*.). Les derniers travaux recensés dans le domaine du tennis sont ceux de Pingali *et al.* [Pin98, Pin99, Pin00b, Pin00a]. Le logiciel développé, connu sous le nom de *LucentVision*, est actuellement utilisé dans plusieurs tournois majeurs de l'ATP par différentes chaînes de télévision. Il permet, à partir de différentes caméras statiques placées autour du court, de suivre en temps réel les joueurs et la balle. Les résultats obtenus sont alors utilisés pour générer des statistiques sur la rencontre, indexer des séquences de jeu (en vue d'une recherche dans une base de données), ou encore présenter les actions par des vues virtuelles.

Le panorama dressé dans cette section illustre la grande variété des travaux de la littérature. Malgré cette diversité importante des approches, nous notons tout de même certaines similarités. Ainsi, la plupart des méthodes proposées sont des méthodes spécifiques à l'application envisagée et ne peuvent généralement pas être utilisées dans d'autres contextes. Elles font appel à des connaissances *a priori* leur permettant de formuler un certain nombre d'hypothèses. Au contraire, nous avons choisi dans ce mémoire une démarche résolument différente puisque nous proposons de combiner des outils génériques (paramétrables en fonction du contexte) pour répondre à un problème particulier.

En outre, notre contribution se situe également au niveau des contraintes respectées par le système proposé. Nous considérons ici les contraintes énoncées en page 3 : $\mathcal{C}_{\text{rapidité}}$, $\mathcal{C}_{\text{couleur}}$, $\mathcal{C}_{\text{illumination}}$, $\mathcal{C}_{\text{mouvement}}$, voire $\mathcal{C}_{\text{compression}}$. Ce n'est pas le cas des approches de la littérature qui ne considèrent que certaines de ces contraintes.

Nous souhaitons respecter les contraintes prédéfinies, et utiliser les outils génériques introduits précédemment au sein d'une architecture spécifique. Dans la prochaine section, nous détaillerons l'architecture proposée.

## 8.3 Architecture proposée

Dans la section précédente, nous avons dressé un panorama des travaux de la littérature relatifs à l'analyse de séquences vidéo sportives. Cela nous a permis notamment de situer le problème que nous traitons ici par rapport à la littérature. Parmi les contraintes spécifiques à notre application, nous pouvons citer, notamment celles relatives à :

$\mathcal{C}_{\text{source}}$ : nous considérons une **source vidéo non-contrôlée**, ce qui implique que nous ne disposons *a priori* que de peu d'informations sur la source vidéo télévisée utilisée, qui com-

porte, outre un bruit relativement important, de nombreux artefacts insérés en régie : changements de plans brusques ou progressifs, ralentis (voire arrêts sur images), texte incrusté dans l'image (score, temps de jeu écoulé, *etc*), bande son provenant de différentes sources audio, *etc.*

$\mathcal{C}_{\text{mouvement}}$ : les images sont acquises à l'aide d'une ou plusieurs caméras en mouvement, et non statiques comme dans de nombreuses approches de la littérature, et dont les paramètres (de mouvement, de focale) nous sont également inconnus.

$\mathcal{C}_{\text{rapidité}}$ : le temps de calcul doit être le plus faible possible, de façon à garantir un délai minimal entre l'instant où l'événement important se produit et celui où l'extrait vidéo correspondant peut être diffusé aux utilisateurs.

Afin de résoudre le problème présenté précédemment, nous proposons ici d'utiliser l'architecture et les outils décrits dans les chapitres précédents de ce document : le découpage de la séquence vidéo en plans (ou segmentation temporelle), la séparation des objets et du fond, la modélisation de la scène, et le suivi des différents objets. Dans le contexte de cette application, trois autres étapes nous semblent également nécessaires et seront décrites brièvement : l'interprétation du contenu de la séquence vidéo, l'interprétation des données écrites, et l'interprétation des données sonores. La figure 8.2 donne une vue d'ensemble du système d'indexation proposé, une courte description a été donnée dans [Lef01a].

**Découpage de la séquence en plans**

La première étape du système d'indexation vidéo consiste en la détection des différents changements de plans présents dans la séquence. Cette segmentation temporelle peut être réalisée à l'aide des nombreuses solutions proposées dans la littérature. En considérant les différentes contraintes à prendre en compte dans notre application, l'approche par blocs dans l'espace TSL, décrite dans le chapitre 2, semble particulièrement adaptée [Lef00b, Lef01d, Lef01c]. Elle donne notamment des résultats intéressants dans le cas de plans successifs au contenu similaire (par exemple des images représentant en grande partie la zone de jeu, c'est-à-dire la pelouse) ou dans le cas de forts mouvements (qui sont fréquents dans des séquences vidéo issues de retransmissions sportives).

Pour chaque changement de plans détecté, il est possible d'isoler le plan correspondant en vue d'une analyse plus poussée. Ainsi, chaque plan peut être caractérisé en fonction de différents attributs : plan rapproché ou large [Xu01b], ralenti [Kob99, Pan01] (voire arrêts sur images), plan de synthèse, plan publicitaire, *etc*.

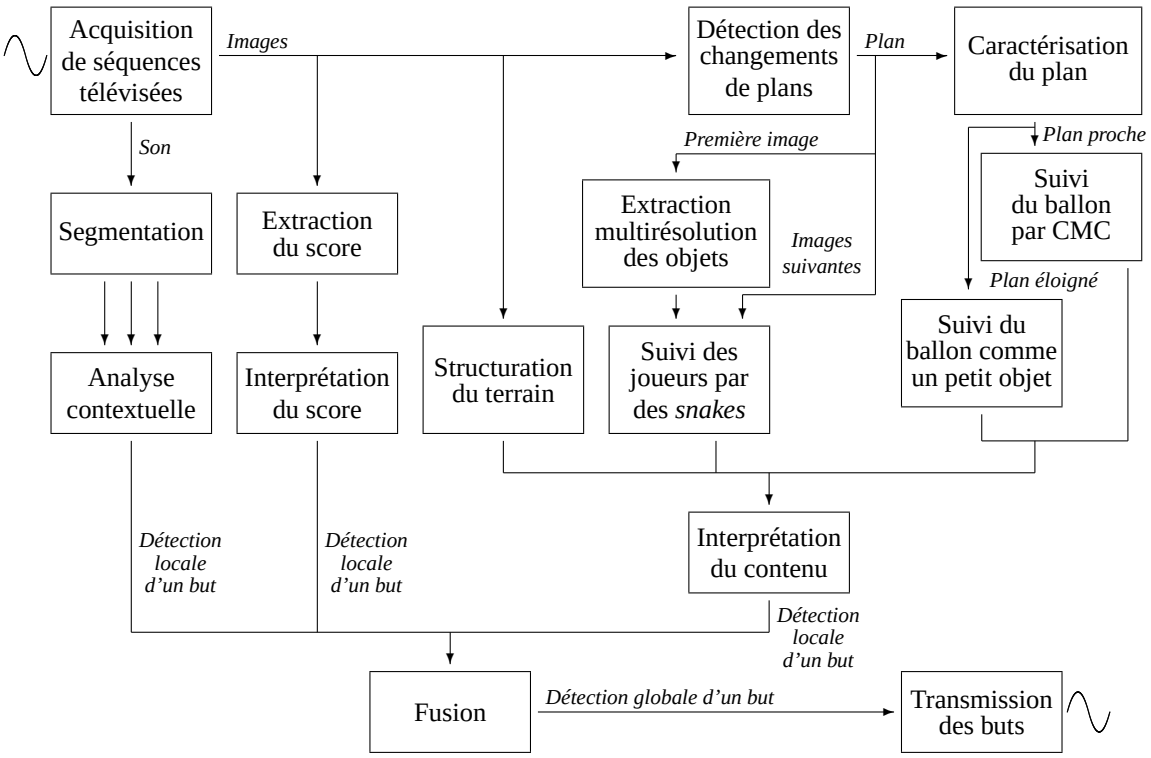


FIG. 8.2 – Architecture permettant la détection des buts dans des matchs de football.

**Séparation des objets et du fond**

A l'apparition de chaque nouveau plan dans la séquence, certaines informations connues par le système deviennent obsolètes. Ainsi, il devient inutile d'utiliser les positions des objets dans l'image précédente pour prédire celles dans l'image courante. Nous analysons donc la première image de chaque plan afin de déterminer les positions initiales des objets dans le plan. Cette détection peut être effectuée en utilisant l'approche multirésolution (section 3.3) [Lef02i, Lef02a], qui est particulièrement adaptée aux séquences vidéo représentant des matchs de football. En effet, le fond (c'est-à-dire la pelouse) étant uniforme, l'utilisation de cette méthode nous permet de garantir un traitement en temps réel. Le résultat obtenu est alors utilisé comme initialisation pour l'étape de suivi d'objet.

**Structuration et modélisation de la scène**

L'étude de l'arrière-plan consiste également en la modélisation de la scène analysée. Comme il a été précisé dans le chapitre 4, nous proposons d'utiliser des primitives extraites des images

(plus particulièrement des lignes, c'est-à-dire des contours rectilignes) afin de modéliser la structure de l'arrière-plan. La scène représente ici un terrain de football, et peut donc être modélisée par les limites du terrain et les poteaux. Les limites du terrain consistent principalement en deux bandes blanches horizontales et deux bandes blanches dont l'orientation dépend de la perspective. De même, les poteaux consistent en deux bandes blanches verticales et une bande blanche dont l'orientation dépend de la perspective. La figure 8.3 illustre cette situation. Nous sommes donc confronté à deux types de bandes : celles dont l'orientation est connue (ici horizontale ou verticale) et celles dont l'orientation varie selon la perspective (et est donc inconnue *a priori*). Afin de limiter les calculs à effectuer, nous avons fait le choix de ne considérer que les bandes dont l'orientation est connue. Lorsque les différentes orientations (ici horizontale et verticale) sont connues, la méthode décrite dans la section 4.3 [Lef02c] peut être utilisée puisqu'elle fournit comme résultat les lignes pertinentes selon un contexte donné. En considérant que chaque bande blanche est représentée par deux lignes parallèles, nous cherchons dans un premier temps, parmi les lignes horizontales, les deux couples de lignes parallèles correspondant aux limites du terrain. Pour cela nous considérons la longueur de chaque ligne ainsi que la distance séparant chaque couple de lignes. Une fois les deux couples de lignes correspondant aux bandes horizontales du terrain détectées, nous simplifions chaque couple par une seule ligne centrale. Nous disposons donc de deux lignes $L_1$ et $L_2$ dont les extrémités horizontales sont notées $(x_{\text{min}}^{L_1}, x_{\text{max}}^{L_1})$ et $(x_{\text{min}}^{L_2}, x_{\text{max}}^{L_2})$. Nous nous focalisons ensuite sur la recherche de la position des poteaux dans l'image. En se basant sur les connaissances *a priori* disponibles sur la représentation du terrain dans l'image, nous analysons d'une part les lignes verticales situées dans l'intervalle d'extrémités $x_{\text{min}}^{L_1}$ et $x_{\text{min}}^{L_2}$ et d'autre part celles situées dans l'intervalle d'extrémités $x_{\text{max}}^{L_1}$ et $x_{\text{max}}^{L_2}$. Dans chaque cas, et afin d'isoler les lignes correspondant aux poteaux, nous cherchons les deux couples de lignes vérifiant les propriétés suivantes :

- distances inter-lignes (ou intra-couple) inférieures à un seuil,
- distances inter-couple proches,
- et enfin extrémités $y_{\text{min}}^{L}$ et $y_{\text{max}}^{L}$ toutes deux comprises dans l'intervalle $[y^{L_1}, y^{L_2}]$.

Les couples sont alors remplacés par des lignes uniques suivant le même procédé que précédemment. Puisque nous disposons maintenant de six lignes (les deux lignes horizontales du terrain, et les deux lignes verticales de chacun des deux poteaux), il nous est possible de reconstruire artificiellement les autres lignes, dont l'orientation varie en fonction de la prise de vue. La figure 8.4 illustre cette reconstruction. Le processus de modélisation décrit ici nous permet d'obtenir la structure de la scène, qui sera utilisée par la suite pour interpréter les positions des différents objets dans la séquence vidéo analysée.

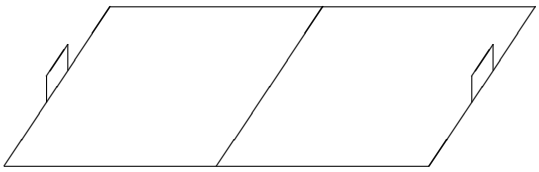

FIG. 8.3 – Représentation des limites d'un terrain de football et des poteaux.

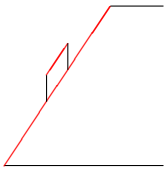

FIG. 8.4 – Reconstruction des lignes obliques dans le modèle d'une scène de football à l'aide des lignes horizontales et verticales.

**Suivi des différents objets présents dans la scène**

Dans une séquence vidéo représentant un match de football, différents objets d'intérêt se déplacent dans la scène et doivent être suivis. Nous nous focalisons ici sur les acteurs du match (joueurs et arbitres) et le ballon. Les premiers sont des objets non-rigides tandis que le ballon est un objet rigide pouvant dans certains cas (lorsqu'il ne s'agit pas d'un plan éloigné) faire l'objet d'un apprentissage. Nous avons donc décidé de suivre ces objets de différente nature avec des méthodes adaptées (partie III). Les joueurs et l'arbitre, du fait de leur caractère non-rigide, sont suivis à l'aide de la méthode basée sur les *snakes*. L'initialisation sur la première image de chaque plan est basée sur le résultat obtenu par la séparation des objets et du fond. Le ballon est quant à lui un objet rigide. Il peut donc être suivi avec l'approche basée sur les chaînes de Markov cachées multidimensionnelles ou simplement par *template matching*. Cependant, nous ne pouvons utiliser cette méthode que si un apprentissage préalable du ballon est possible et uniquement si le ballon est représenté par un nombre de pixels suffisament important dans les images. Or, dans certains plans (notamment les plans larges ou éloignés), la faible taille du ballon l'empêche de pouvoir être correctement suivi avec cette méthode. Dans ce cas, il est préférable d'employer les méthodes spécifiques au suivi de petits objets. Pour chaque plan de la séquence, un choix doit donc être effectué quant à la méthode à employer pour suivre le ballon.

Aussi, lorsqu'un changement de plans est détecté et qu'une segmentation du fond et des objets est effectuée, nous sélectionnons la méthode à utiliser en déterminant le type de plan analysé (proche ou éloigné). Cette information est obtenue par analyse de la taille moyenne des objets, en utilisant des informations *a priori* sur le contenu d'une retransmission sportive.

A l'aide des différentes techniques de suivi d'objet utilisées, nous pouvons connaître dans chaque image la position des joueurs et du ballon.

**Interprétation du contenu de la séquence**

En combinant les résultats des deux étapes précédentes, nous pouvons interpréter le contenu de la séquence et obtenir une première décision quant à la détection d'un événement prédéfini (ici un but inscrit lors d'un match de football). Pour ce faire, nous comparons la position du ballon dans l'image avec la position d'une des deux zones de buts (définie par deux lignes verticales, les poteaux, et deux lignes obliques) si l'une d'entre elles est présente dans l'image. Si le ballon est compris dans une zone de but, nous considérons que l'événement a de grandes chances de s'être produit. Les positions des joueurs peuvent alors être utilisées pour vérifier cette décision en cherchant, par exemple, si un regroupement des joueurs (c'est-à-dire des positions proches) est observé dans les images suivantes. D'autres règles peuvent également être utilisées.

**Interprétation des données écrites**

L'analyse des positions des objets dans la scène nous a fourni une première décision. Nous pouvons également analyser le texte incrusté dans la séquence vidéo pour détecter les événements prédéfinis. En effet, lorsqu'un but est inscrit, le score s'en trouve modifié et des changements dans l'une des zones de texte incrusté sont observés. Le problème de la détection du texte incrusté dans les images ou les séquences vidéo est relativement complexe et a fait l'objet de nombreux travaux publiés dans la littérature. Les domaines considérés peuvent être spécifiques (comme le sport [Sun02, Che02a] ou les journaux télévisés [Sat99]) ou plus généraux [Jai98a, Kim01c, Li00a, Wu99, Zho00a]. Nous avons développé une méthode simple pour résoudre ce problème. En prenant comme hypothèse l'invariance relative des valeurs des pixels appartenant à des zones de texte (par rapport au reste de l'image), la détection des zones de texte est basée sur l'évolution temporelle (mesurée par la variance) des pixels dans l'image. Afin de limiter la sensibilité de la méthode au bruit, nous travaillons sur des blocs plutôt que sur des pixels directement. Enfin, les résultats sont également améliorés par intégration d'une étape de détection de contours (les zones de texte étant des zones où la densité de contours

est relativement élevée). Des exemples de résultats obtenus sont donnés dans la figure 8.5. Les zones de texte détectées sont ensuite analysées au cours du temps pour détecter un changement du contenu du texte incrusté. L'apparition d'un tel changement se traduit par la décision qu'un événement a été détecté. Nous ne considérons évidemment pas les zones caractérisées par des changements très fréquents (comme celle correspondant au temps écoulé depuis le début du match).


FIG. 8.5 – Extraction du texte incrusté par détection de contours et analyse de l'évolution temporelle des couleurs des pixels.

**Interprétation des données sonores**

Nous proposons en outre une troisième soure d'information permettant d'obtenir des indices supplémentaires pour la décision. Celle-ci est basée sur l'analyse des données audio contenues dans les retransmissions télévisées. Dans ce cadre, nous avons développé différentes techniques de classification d'extraits audio [Lef02g, Lef02h] décrites dans la section suivante. Le résultat obtenu consiste en un étiquetage de chaque extrait en une classe parmi 3 : la voix du commentateur, le bruit de la foule, et le sifflet de l'arbitre. Chaque type d'extrait peut ensuite faire l'objet d'une analyse plus contextuelle nous permettant de détecter un événement prédéfini.

**Décision globale par fusion des décisions locales**

Finalement, les décisions obtenues par les différents canaux (images, texte, audio) peuvent être combinées dans le but de générer une décision globale. Cette combinaison peut être effectuée par des règles simples (vote $2/3$ par exemple) mais aussi par des techniques plus sophistiquées telles qu'une fusion hiérarchique [Mee99] ou à base de logique floue.

Les différents outils, pour la plupart génériques, décrits précédemment peuvent donc être combinés en respectant une architecture donnée, générique elle aussi, afin de former un système

spécifique d'indexation vidéo dont le but est la détection, en temps réel, des buts inscrits lors d'un match de football.

## 8.4 Analyse supplémentaire des données sonores

Les séquences vidéo analysées sont issues de retransmissions télévisées. Elles contiennent, en plus des images, une bande son qu'il est possible d'analyser. Nous aborderons dans cette section le problème lié à l'analyse de pistes audio pour la détection d'événements (ici des buts inscrits dans des matchs de football) [Lef02g, Lef02h].

Après avoir expliqué plus en détail le problème à résoudre et donné quelques références de travaux similaires dans la littérature, nous présenterons un ensemble de méthodes permettant de classer de courts extraits audio (d'une durée d'environ une seconde). Parmi celles-ci, nous insisterons notamment sur un classifieur en deux étapes combinant l'algorithme des K-Means avec les chaînes de Markov cachées multidimensionnelles.

### 8.4.1 Problématique

Le but de l'analyse décrite ici est de détecter, à partir des données audio, des événements prédéfinis. La bande son de la séquence vidéo est ici analysée séparément des images afin de détecter des événements particuliers.

La détection d'un but par analyse de la piste audio peut être obtenue de différentes manières. En effet, il est possible d'effectuer une reconnaissance vocale de la voix du commentateur afin de chercher des occurrences de mots importants dans ce contexte, tels que "but", "tir", "score", *etc.*. L'analyse des bruits de la foule des spectateurs peut également fournir une information importante. Enfin, d'un point de vue plus théorique, les connaissances disponibles sur les règles d'un match de football peuvent amener à la recherche d'un but par la recherche de deux coups de sifflet consécutifs de la part de l'arbitre.

Puisque la piste audio contient de nombreux signaux différents (voix du commentateur, bruit de la foule, coup de sifflet, *etc.*), et que chacun de ces types de signaux peut être analysé séparément pour fournir une information quant à la présence ou non de l'événement prédéfini, nous proposons d'analyser les données audio en deux étapes.

La première étape consiste à découper le signal sonore en portions de courte durée et à les classer. La durée de chaque extrait sonore est de l'ordre d'une seconde. Les classes corres-

pondent aux différents types de son : voix du commentateur, bruit de la foule, coup de sifflet de l'arbitre.

Il est alors possible, dans une seconde étape, d'analyser de manière contextuelle chaque extrait sonore. Ainsi, la reconnaissance vocale sera limitée aux extraits de voix du commentateur, le comptage de coups de sifflet ne sera effectué que dans les extraits correspondants, et le volume sonore de la foule sera analysé uniquement dans les extraits adéquats.

Par manque de temps, la seconde étape n'a pu être étudiée au cours de cette thèse. Nous ne traiterons donc par la suite que de la classification d'un extrait sonore. Après avoir décrit quelques travaux similaires de la littérature, nous présenterons différentes méthodes qu'il est possible d'utiliser pour segmenter des pistes audio de retransmissions de matchs de football.

### 8.4.2 Analyse de données audio

Nous devons tout d'abord déterminer si l'analyse des séquences audio sera effectuée globalement ou localement. Dans le premier cas, le but est de classer des séquences audio complètes, comme dans l'approche de Wang *et al.* [Wan00] pour classer des pistes audio télévisées. Comme nous l'avons décrit précédemment, il est également possible de classer de courts segments ou extraits audio (généralement d'une durée d'une seconde) afin de détecter des événements dans des séquences audio. Il est possible de se référer dans ce cas aux travaux de Kermit et Eide [Ker00] ou de Zhang et Kuo [Zha99], ces derniers effectuant un traitement en temps réel.

La segmentation et la classification de données audio ont été étudiées par de nombreux chercheurs. Ce problème peut être vu comme un problème de reconnaissance de formes où deux choix doivent être effectués. Ceux-ci concernent le classifieur à utiliser et les caractéristiques audio sur lesquelles se baser. Li *et al.* [Li01] ont étudié un total de 143 caractéristiques pour déterminer leurs capacités de discrimination. Pfeiffer *et al.* présentent dans [Pfe96] des caractéristiques utilisées dans l'analyse de données audio. Wold *et al.* [Wol99a] analysent et comparent des caractéristiques audio dans un but d'indexation audio basée sur le contenu. Li [Li00b] effectue des expérimentations pour comparer diverses méthodes de classification et différents ensembles de caractéristiques. Bocchieri et Wilpon [Boc93] discutent de l'influence du nombre de caractéristiques et de la nécessaire sélection d'une partie des caractéristiques disponibles. Lorsque des séquences audio compressées sont analysées, il est également possible de calculer certaines caractéristiques spécifiques, comme dans les travaux de Tzanetakis et Cook [Tza00] avec le MPEG audio.

Plusieurs chercheurs ont proposé d'utiliser les CMC afin d'effectuer l'analyse de données audio. Kimber et Wilcox [Kim96] créent une CMC pour chaque locuteur ou classe acoustique. L'apprentissage et la reconnaissance sont basés sur les algorithmes de Baum-Welch et de Viterbi. Battle et Cano [Bat00] proposent, dans le cas d'un apprentissage non-supervisé, l'utilisation de CMC compétitives au lieu des CMC traditionnelles. Finalement Hirsch [Hir01] utilise une architecture de CMC adaptative afin de traiter des signaux audio issus des télécommunications.

### 8.4.3 Description de quelques approches

Nous présentons ici quelques techniques que nous avons expérimentées pour classer chaque extrait audio en deux ou trois classes parmi celles définies précédemment : voix du commentateur, bruit de la foule, sifflet de l'arbitre. Ces approches [Lef02g] sont présentées dans l'ordre croissant de leur complexité. La dernière est une approche plus personnelle et originale dont nous préconisons l'utilisation [Lef02h].

#### Quelques approches simples

Avant de proposer une méthode complexe pour la segmentation audio, nous avons vérifié que les limitations des approches les plus simples ne permettaient pas de résoudre convenablement notre problème. Nous pourrons ainsi utiliser la fréquence ou l'amplitude du signal sonore comme un élément discriminant.

La fréquence du signal peut être utilisée pour détecter les extraits audio correspondant au sifflet de l'arbitre. En effet, celui-ci produit un son composé de deux ou trois fréquences appartenant à l'intervalle $[3700, 4300]$ Hz. Un exemple de spectrogramme représentant un extrait de sifflet est donné dans la partie gauche de la figure 8.6. Il est aisé de distinguer les lignes horizontales représentant les fréquences du son correspondant au sifflet. La classification en deux classes, sifflet et autre, peut alors être effectuée en trois étapes successives. Le spectrogramme est seuillé afin de ne garder uniquement que les valeurs les plus significatives. Ensuite, pour chaque fréquence, l'amplitude associée est calculée. Finalement l'extrait audio est classé en sifflet si le nombre de lignes obtenues est égal ou supérieur à deux. La principale limite de cette approche vient du fait que les fréquences liées au sifflet peuvent être un sous-ensemble des fréquences liées à la voix du commentateur. De ce fait, il est possible de classer des extraits de voix du commentateur en sifflet de l'arbitre. La partie droite de la figure 8.6 contenant le spectrogramme d'un extrait de voix illustre ce problème.

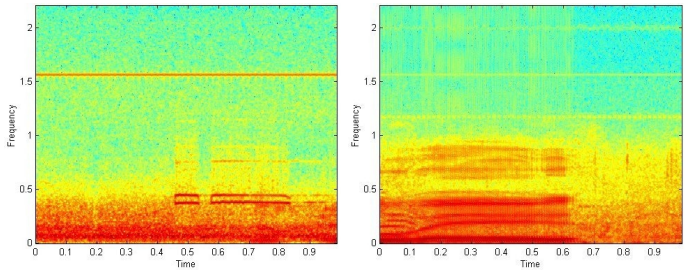


FIG. 8.6 – Spectogrammes de signaux audio correspondant au sifflet de l'arbitre (à gauche) et à la voix du commentateur (à droite).

Il est également possible d'utiliser l'amplitude du signal. Celle-ci peut notamment être utile pour classer un extrait en voix du commentateur ou en bruit de la foule. Un signal audio contenant ces deux types d'extraits classés manuellement est présenté dans la figure 8.7. Nous constatons que l'amplitude moyenne n'est pas égale pour les deux classes. Cette mesure peut donc être utilisée pour effectuer la classification. Si l'amplitude moyenne est supérieure à un seuil, l'extrait audio est classé en voix du commentateur. Sinon il est associé au bruit de la foule. Nous considérons que les propriétés des données audio sont constantes tout au long de la séquence. Il n'est alors pas nécessaire d'utiliser un seuil adaptatif mais simplement un seuil fixe dont la valeur est fixée après apprentissage. Si nous considérons des pistes audio, provenant de différentes retransmissions télévisées, qui se caractérisent par une grande variation de leurs propriétés, l'apprentissage devra être effectué en ligne en considérant un corpus constitué des premières secondes de la piste audio. Les résultats obtenus sont donnés dans le tableau 8.1. A partir de ces résultats, nous pouvons conclure que la qualité de la méthode n'est pas suffisante pour effectuer une classification correcte.

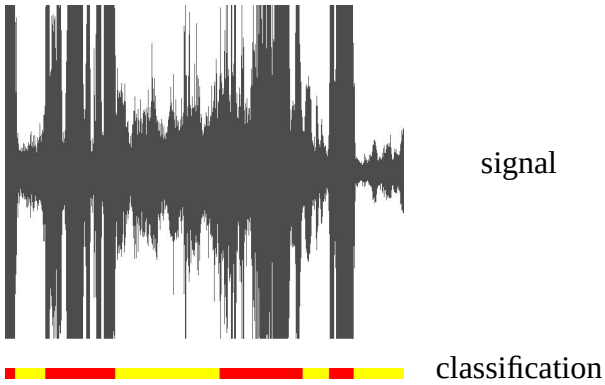


FIG. 8.7 – Signal audio (en haut) contenant des extraits classés manuellement (en bas) comme "commentateur" (en rouge) ou "foule" (en jaune).

Les taux de qualité présentés dans les tableaux de cette section ont été calculés en comparant les résultats obtenus par les différentes méthodes de segmentation avec une segmentation de référence obtenue manuellement par plusieurs auditeurs. Nous ne considérons ici que les extraits audio ayant fait l'objet d'un consensus entre tous les utilisateurs. Précisons également que les pistes audio traitées correspondent à différentes retransmissions télévisées de différents matchs de football.

| Classe | Rappel | Précision |
|---|---|---|
| Foule | 77 % | 50 % |
| Commentateur | 62 % | 84 % |

TAB. 8.1 – Résultats d'une segmentation en deux classes basée sur l'analyse de l'amplitude du signal.

Nous venons de montrer que l'utilisation directe d'éléments simples tels que la fréquence ou l'amplitude ne permet pas d'obtenir des résultats satisfaisants. Nous proposons donc d'utiliser des caractéristiques plus complexes, comme une analyse cepstrale.

**Analyse cepstrale**

Le cepstre est un outil fréquemment utilisé en analyse et reconnaissance de la parole. Il est défini comme une application de trois opérateurs successifs : une transformée de Fourier, un logarithme, et une transformée de Fourier inverse. Cette transformation permet de déterminer la fréquence fondamentale de la parole et de séparer le signal d'excitation du signal de parole pure. De même qu'un spectrogramme représente le spectre d'un signal, il est possible d'utiliser un cepstrogramme qui est une représentation graphique en trois dimensions du signal audio basée sur le calcul du cepstre. La figure 8.8 montre les projections 2-D de deux cepstrogrammes obtenus pour des extraits de foule et de voix du commentateur.

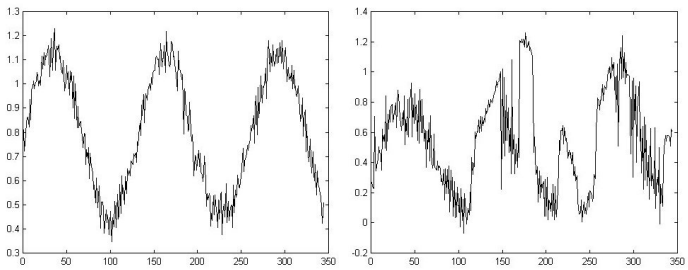

FIG. 8.8 – Projection 2-D du cepstrogramme d'un signal audio correspondant à la foule (à gauche) et à la voix du commentateur (à droite).

A partir de cette figure, nous pouvons noter que la foule est représentée par une courbe sinusoïdale, ce qui n'est pas le cas de la voix du commentateur. Deux raisons peuvent être avancées pour expliquer ce phénomène : le son émis par la foule peut être considéré comme un bruit, ou bien il peut être amplifié par le stade (phénomène d'écho). Il est donc possible de segmenter ces deux types de son (foule et voix du commentateur) en se basant sur le principe énoncé. Pour cela, nous effectuons une régression de la courbe obtenue par une sinusoïde de même fréquence et de même phase. La différence avec la sinusoïde théorique est mesurée puis comparée à un seuil (obtenu par une procédure d'apprentissage supervisé) pour classer l'extrait en foule ou en voix. Les résultats obtenus sont donnés dans le tableau 8.2.

| Classe | Rappel | Précision |
|---|---|---|
| Foule | 72 % | 96 % |
| Commentateur | 98 % | 86 % |

TAB. 8.2 – Résultats d'une segmentation en deux classes basées sur une analyse cepstrale.

Il est clair que la qualité de cette dernière approche est plus élevée que celle des approches simples décrites précédemment. Cependant, les taux de qualité ne sont pas encore satisfaisants et la méthode décrite ici ne permet de gérer que deux classes différentes. Nous proposons alors d'utiliser les chaînes de Markov cachées pour caractériser les extraits de chacune des classes afin de procéder à la classification de nouveaux extraits.

**Utilisation des chaînes de Markov cachées**

Les méthodes décrites précédemment, basées sur l'utilisation d'une unique caractéristique, ne permettent pas d'obtenir des résultats satisfaisants. Afin de combiner ces différentes caractéristiques, il est possible d'utiliser les chaînes de Markov cachées déjà présentées dans la section 5.2. L'architecture ici employée est également ergodique, tandis que l'apprentissage et la reconnaissance sont effectués avec les algorithmes de Baum-Welch et de Forward respectivement. Nous voulons définir trois CMC, une pour chaque classe. Chaque extrait sera alors affecté à la classe associée au meilleur score.

Les données d'observation que nous considérons consistent en un ensemble de caractéristiques dont l'intérêt a été validé dans [Liu98]. Pour chaque segment audio d'une durée d'une seconde, nous calculons les 11 caractéristiques suivantes : non-silence ratio (NSR), volume standard deviation (VSTD), standard deviation of zero crossing rate (ZSTD), volume dynamic range (VDR), standard deviation of pitch period (PSTD), smooth pitch ratio (SPR), non-pitch ratio (NPR), frequency centroid (FC), frequency bandwidth (FB), 4 Hz modulation energy (4ME), et

energy ratio of subband 1-3 (ERSB1-3). Lorsque l'on travaille avec plusieurs caractéristiques en analyse des données audio, il est nécessaire de déterminer quelles caractéristiques fournissent la plus grande contribution pour la performance de la reconnaissance, et de sélectionner ces caractéristiques [Boc93]. Nous avons donc effectué une Analyse en Composantes Principales (ACP) en considérant ces 11 caractéristiques. En conclusion de cette analyse, nous avons observé qu'aucune caractéristique ne devait être rejetée du fait de son manque de contribution.

Afin d'analyser des extraits sonores d'une durée d'une seconde, nous les divisons en trames contenant chacune 1024 échantillons. Deux trames successives seront décalées de 512 échantillons dans le but de conserver la propriété de continuité. Le tableau 8.3 montre les résutlats obtenus avec cette méthode.

| Classe | Rappel | Précision |
|---|---|---|
| Sifflet | 88 % | 88 % |
| Foule | 61 % | 87 % |
| Commentateur | 77 % | 90 % |

TAB. 8.3 – Résultats d'une segmentation en trois classes basée sur les chaînes de Markov cachées.

Afin d'améliorer la qualité de la classification, nous proposons une méthode plus élaborée, basée sur la diminution de la variabilité des caractéristiques des signaux au sein de chacune des classes. Pour cela nous utilisons conjointement un classifieur K-Means et les chaînes de Markov cachées multidimensionnelles.

### 8.4.4 Classification en 2 étapes

Au vu des expérimentations décrites précédemment, nous avons noté que la difficulté la plus importante pour résoudre le problème posé est liée à la variabilité des caractéristiques extraites pour chaque classe. Cette remarque a motivé les choix que nous avons fait quant à l'élaboration d'une nouvelle méthode [Lef02h] plus adaptée. Ainsi, nous considérons ici deux échelles d'observations, l'échelle des segments et l'échelle des trames. Dans un premier temps, les segments complets sont classifiés dans le but d'obtenir des classes aux caractéristiques plus similaires. Ces classes n'ont cependant aucun rapport avec l'information finale que nous souhaitons extraire des données. Elles ont pour but de permettre d'adapter le processus aux caractéristiques partagées au sein de chaque classe. La deuxième étape est alors basée sur les résultats obtenus et sur l'appretissage de chaînes de Markov cachées multidimensionnelles sur une population plus homogène.

**Rappels sur le classifieur K-Means**

Le classifieur K-Means est un outil fréquemment utilisé en reconnaissance des formes [Fuk90]. Ce classifieur considère un nombre de classes K déterminé *a priori*. Le résultat est défini par la position des centres de chaque classe, et est obtenu en minimisant les distortions moyennes (calculées à l'aide d'une distance euclidienne) de tous les points appartenant aux K classes.

Précisons les étapes successives de cet algorithme. Tout d'abord les centres initiaux des K classes sont fixés aléatoirement. Ensuite, chaque point à classer est assigné à la classe la plus proche, ce qui forme une nouvelle partition des données. La classe choisie est celle qui minimise une distance euclidienne calculée entre son centre et le point concerné. Les centres des classes sont alors recalculés en fonction de la nouvelle partition. Ces deux dernières étapes sont répétées jusqu'à convergence, c'est-à-dire lorsqu'aucun point ne change d'affectation.

**Description générale de l'approche**

Comme dans n'importe quelle autre méthode, la méthode de reconnaissance décrite ici est basée sur un ensemble de caractéristiques. Nous pensons qu'il est important d'analyser des informations à différentes échelles d'observation car celles-ci permettent d'obtenir différents types d'informations. Cependant, il est difficile de gérer ces différentes échelles en se basant uniquement sur un modèle de Markov caché. Aussi avons-nous décidé de considérer deux échelles d'observation dans la piste audio. Cette piste audio (notée PA) est divisée en segments (notés SA), eux-mêmes divisés en trames (notées TA). Les caractéristiques peuvent être calculées à l'échelle des segments ou à l'échelle des trames. Nous utilisons tout d'abord un classifieur K-Means à l'échelle des segments nous permettant d'obtenir un ensemble de K classes dites "virtuelles" car elles ne correspondent pas aux C classes prédéfinies pour qualifier chaque segment. Au sein de chaque classe virtuelle, la variabilité du signal par rapport aux caractéristiques utilisées est donc diminuée. Il est alors possible de restreindre la recherche des classifieurs à chaque classe virtuelle afin d'étiqueter les segments audio. Nous utilisons pour cela les caractéristiques calculées à l'échelle des trames et ($C \times K$) chaînes de Markov cachées multidimensionnelles. Le diagramme de l'approche proposée est donné dans la figure 8.9.

Nous allons maintenant préciser les caractéristiques utilisées, ainsi que les procédures d'apprentissage et de reconnaissance.

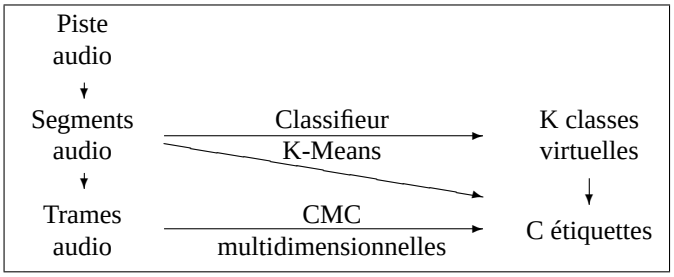


FIG. 8.9 – Diagramme de l'approche en deux étapes.

**Caractéristiques audio**

Nous effectuons la segmentation audio en utilisant un ensemble de 12 caractéristiques. Les onze premières, tirées de [Liu98], ont été introduites page 158. Parmi celles-ci, cinq sont liées à un segment audio (NSR, SPR, NPR, VDR, et 4ME) et vont être utilisées dans la première étape de segmentation basée sur un classifieur K-Means. Les 6 autres caractéristiques (VSTD, ZSTD, PSTD, FC, FB, ERSB) sont calculées à l'échelle des trames et seront donc analysées *via* des chaînes de Markov cachées multidimensionnelles. La dernière caractéristique, également calculée à cette échelle, représente l'information cepstrale et est notée CBF (Cepstrum-Based Feature). Son intérêt a été validé page 157.

**Phase d'apprentissage**

L'apprentissage est effectué de manière supervisée et est basé sur trois étapes successives décrites ci-dessous : le calcul des caractéristiques, la classification à l'aide de l'algorithme du K-Means, et la création des CMC.

Afin d'analyser un segment audio SA, les caractéristiques sont calculées, et nous définissons N trames se chevauchant. Nous les notons $FA_1$ à $FA_N$.

Le classifieur K-Means peut alors être appliqué en utilisant seulement les caractéristiques calculées à l'échelle des segments afin d'obtenir une première classification en K groupes ou classes virtuelles (figure 8.10 (a)). Le paramètre K doit être fixé *a priori* et a une influence directe sur le nombre de CMC multidimensionnelles à construire par la suite. L'utilisation de ce classifieur K-Means dans le processus de création des CMC multidimensionnelles permet d'augmenter la qualité de l'étape suivante d'étiquetage.

Une fois les segments audio classés dans l'une des K classes virtuelles, ils servent à l'apprentissage des CMC multidimensionnelles appropriées. Le processus d'apprentissage est ef-

fectué en utilisant uniquement les données comprises dans l'une des K classes virtuelles, et pour chacune de ces classes virtuelles, C CMC multidimensionnelles sont élaborées. La figure 8.10 (b) résume ce principe, où p segments audio $SA^{j}_{V_i}$ sont utilisés pour créer les CMC notées $CMC^{i}_{1}$ à $CMC^{i}_{c}$ qui sont liées à la classe virtuelle $SA_{V_i}$. L'algorithme d'apprentissage utilisé est l'algorithme de Baum-Welch dans sa version multidimensionnelle. Il utilise les caractéristiques calculées à l'échelle des trames permettant de décrire les segments par une observation multidimensionnelle dont la longueur est égale au nombre $N$ de trames. La méthode proposée nécessite donc $K \times C$ CMC multidimensionnelles afin d'effectuer une segmentation en C classes.

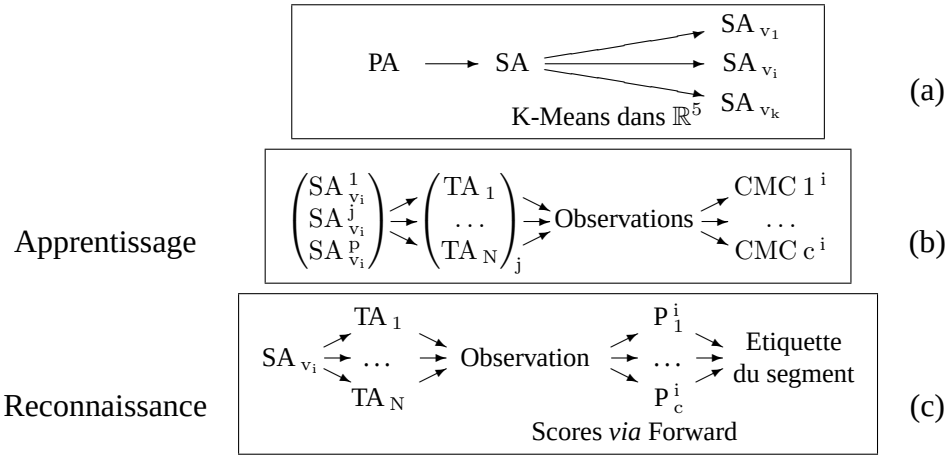


FIG. 8.10 – Description des étapes successives : (a) classification dans des classes virtuelles suivie par (b) l'apprentissage ou (c) la reconnaissance.

**Phase de reconnaissance**

Le but de cette seconde phase est de classer chaque segment audio dans l'une des C classes. Pour cela, nous suivons une procédure similaire à celle présentée pour l'apprentissage. Ainsi, les caractéristiques relatives à chaque segment audio sont tout d'abord calculées. Chaque segment est ensuite classé en utilisant le classifieur K-Means dans l'une des K classes virtuelles (comme le montre la figure 8.10 (a)). Soit $SA_{V_i}$ cette classe. Nous utilisons alors l'algorithme Forward avec les C chaînes de Markov cachées multidimensionnelles (notées $CMC^{i}_{1}$ à $CMC^{i}_{c}$). Le segment audio est finalement associé à la classe pour laquelle l'algorithme de Forward donne le score le plus élevé (noté $P^{i}_{max}$), comme le montre la figure 8.10 (c).

**Résultats**

La méthode décrite dans cette section est une approche générale qui permet de segmenter n'importe quel type de piste audio. Pour le problème qui nous concerne ici, nous cherchons à classer chaque segment (ou extrait) audio dans l'une des trois classes ($\mathrm{C} = 3$) suivantes : le sifflet de l'arbitre, le bruit de la foule, et la voix du commentateur. Une fois que cette classification a été obtenue, nous pouvons, comme il a été précisé précédemment, analyser de manière plus contextuelle chaque segment audio.

Nous avons choisi de manière empirique d'utiliser 3 classes virtuelles ($\mathrm{K} = 3$) pour le classifieur K-Means. La méthode proposée nécessite donc 9 CMC multidimensionnelles.

Afin d'évaluer la qualité de l'approche proposée ici, nous avons effectué des tests sur un ensemble de 616 segments audio (dont 21 représentent le sifflet de l'arbitre, 148 le bruit de la foule, et 447 la voix du commentateur) issus de différentes séquences vidéo. Les taux de reconnaissance sont donnés dans le tableau 8.4. Par comparaison avec l'approche basée sur les CMC traditionnelles, nous constatons que le taux de rappel est de 10 à 15 % plus élevé pour un taux de précision similaire.

| Classe | K-Means + CMC-MD/I | | CMC (*rappels*) | |
|---|---|---|---|---|
| | Rappel | Précision | Rappel | Précision |
| Sifflet | 95 % | 86 % | 88 % | 88 % |
| Foule | 75 % | 86 % | 88 % | 88 % |
| Commentateur | 95 % | 90 % | 77 % | 90 % |

TAB. 8.4 – Résultats d'une segmentation en trois classes basée sur l'approche en deux étapes.

### 8.4.5 Conclusion

Nous avons abordé ici le problème supplémentaire de l'analyse des données audio, dans le but de détecter des buts inscrits dans des matchs de football. Ces données contiennent de nombreuses informations provenant de différentes sources. Nous avons donc envisagé de résoudre le problème en deux étapes : la première consiste en la classification de chaque extrait sonore en différentes classes (par exemple : voix du commentateur, bruit de la foule, sifflet de l'arbitre) tandis que la seconde concerne l'analyse contextuelle de chaque extrait sonore en fonction de sa classe. Par manque de temps, cette deuxième étape n'a pu faire l'objet d'une étude au cours de cette thèse.

Les méthodes les plus simples ont montré qu'elles ne pouvaient garantir une qualité suffi-

sante. Nous avons proposé de faire précéder par un classifieur K-Means l'utilisation de chaînes de Markov cachées multidimensionnelles. Les résultats obtenus nous ont permis de valider expérimentalement cette méthode originale.

## 8.5 Conclusion

Dans ce chapitre, nous avons présenté une application illustrant le cas de l'analyse d'une séquence vidéo non-contrôlée où l'on recherche à détecter, en temps réel, des buts inscrits dans des matchs de football, grâce à l'analyse des séquences vidéo issues des retransmissions télévisées. Le problème qui nous concerne se différencie des autres travaux par les contraintes supplémentaires qu'il intègre : source vidéo non-contrôlée ($\mathcal{C}_{\text{source}}$), caméra en mouvement ($\mathcal{C}_{\text{mouvement}}$), traitement en temps réel ($\mathcal{C}_{\text{rapidité}}$). Ce problème peut être résolu en utilisant le système spécifique d'indexation vidéo, décrit dans ce chapitre, et basé sur l'architecture et les outils génériques présentés dans les différentes parties de ce document. Ce système intègre notamment une étape d'analyse des données audio contenues dans la bande son. Les méthodes d'analyse sonore présentées ici peuvent en outre être utilisées dans d'autres contextes.

Il est également possible de détécter des événements dans des séquences vidéo acquises à l'aide d'une source vidéo dite contrôlée. Ce deuxième cas de figure fait l'objet du chapitre suivant, où deux applications différentes sont présentées. Ces applications font suite aux demandes de l'entreprise Atos Origin, et en particulier à deux de ses équipes.

# Chapitre 9

# Cas d’une source vidéo contrôlée

Dans ce chapitre, nous présentons deux systèmes d’indexation vidéo nous permettant d’illustrer le cas d’une séquence vidéo contrôlée. Ces deux applications ont été étudiées et réalisées à la demande de l’entreprise Atos Origin. La première application concerne l’obtention de statistiques relatives à la fréquentation d’un site. La seconde application répond au besoin de l’équipe Recherche et Développement de l’entreprise, tout particulièrement désireuse d’effectuer une vidéosurveillance autonome d’une salle informatique sécurisée.

Avant de présenter ces deux applications selon une démarche similaire, nous donnerons tout d’abord une définition d’une source vidéo contrôlée. Nous verrons ainsi que ce cas de figure est plus facile à gérer que le précédent, et que, de ce fait, certains algorithmes présentés dans les parties précédentes ne seront pas utilisés ici. L’application liée aux statistiques de fréquentation d’un site sera ensuite présentée. Après avoir introduit le contexte relatif à cette application, nous décrirons l’architecture proposée en insistant sur les différents problèmes à résoudre : l’extraction des objets et du fond, la modélisation de la scène, le suivi des objets, et enfin l’interprétation du contenu de la scène. Nous commenterons finalement les résultats obtenus, qui ont permis de valider une première version "simple" du logiciel (tandis qu’une seconde version "évoluée" est en cours de réalisation au moment de la rédaction de ce document). La seconde application, concernant la vidéosurveillance d’une salle informatique sécurisée, fera l’objet de la section suivante et sera présentée selon le même principe. Une fois la problématique décrite, nous introduirons l’architecture utilisée et détaillerons les différents outils nécessaires pour résoudre les problèmes exposés précédemment. Cette application n’étant, au moment de la rédaction de ce mémoire, qu’à l’état de projet, et n’ayant pas fait l’objet d’une réalisation complète, nous ne pourrons pas donner de résultats illustrant la validité de la solution proposée.

## 9.1 Les simplifications

Au cours du chapitre précédent, nous avons présenté le problème de l'indexation de séquences vidéo acquises à l'aide d'une source vidéo non-contrôlée, de type séquence télévisée. Nous considérons ici le cas d'une source vidéo contrôlée.

Nous considérons qu'une source vidéo est contrôlée lorsqu'il est possible de disposer de toutes les informations relatives au paramétrage de la source d'acquisition. Ainsi, le mouvement (translation, rotation) de la caméra, la focale et le zoom du capteur sont considérés comme disponibles. Un exemple typique (que nous utiliserons d'ailleurs ici) d'une source vidéo contrôlée est l'acquisition à l'aide d'une caméra numérique fixée sur pied ou plus précisément d'une webcam. Ce type de matériel sera utilisé dans les deux applications présentées dans ce chapitre.

Les séquences vidéo obtenues à l'aide d'une webcam présentent certaines caractéristiques qui permettent de simplifier le processus d'indexation, l'architecture et les outils utilisés.

Ainsi, les séquences vidéo ne sont composées que d'un seul plan, puisqu'aucun effet n'a été ajouté en régie audiovisuelle. Il n'est donc pas nécessaire d'effectuer une détection des changements de plans, comme dans les chapitres 1 et 2.

De plus, selon le besoin de l'application, la caméra peut être statique ou non. Dans les deux applications présentées plus loin dans ce chapitre, nous considérons le cas d'une ou plusieurs caméras statiques. Il est alors plus facile de séparer les objets et l'arrière-plan de la scène, puisque ce dernier est fixe dans les différentes images de la séquence vidéo. Aux outils présentés dans le chapitre 3 peuvent donc être substituées des techniques plus simples, comme la différence avec une image de référence ou entre images successives.

Le suivi des objets (chapitres 5 à 7) est également facilité du fait de l'analyse d'images où l'arrière-plan est fixe et où seul les mouvements des objets (et non plus également celui de la caméra) interviennent dans la modification de l'état des pixels entre deux images successives.

Puisque le cas d'une source contrôlable est plus simple à traiter que le cas d'une source non contrôlable, l'utilisation des techniques avancées présentées dans les trois parties précédentes n'est pas toujours nécessaire. Ainsi il n'est pas illogique de trouver, dans les solutions présentées plus loin, des alternatives plus simples que les techniques présentées précédemment. Ces alternatives vont être décrites dans les deux sections suivantes.

## 9.2 Statistiques de fréquentation d'un site

Dans cette section, nous présentons l'application réalisée à la demande de l'équipe Automates de l'entreprise d'Atos Origin. Cette application concerne l'obtention de statistiques relatives à la fréquentation d'un site, dans le but d'évaluer l'intérêt commercial de la mise en place d'un Distributeur Automatique de Billets (DAB) sur un lieu donné.

Nous présenterons tout d'abord la problématique de cette application. Nous décrirons ensuite l'architecture proposée dans ce contexte, ainsi que les différents outils utilisés pour résoudre les problèmes de l'extraction des objets, de la modélisation de la scène, du suivi des objets, et enfin de l'interprétation du contenu de la séquence.

### 9.2.1 Problématique

Actuellement, une des activités de l'équipe Automates de l'entreprise Atos Origin concerne la gestion commerciale et technique des DAB pour différents clients (banques, assureurs, *etc.*). Parmi les offres proposées aux clients, l'entreprise peut fournir une étude permettant d'estimer l'intêret commercial (en termes de nombre de retraits bancaires) d'un site géographique donné. Cette étude est pour l'instant réalisée manuellement : certaines personnes ont notamment pour rôle de compter, sur un laps de temps donné, le nombre de personnes passant devant l'emplacement présumé du DAB, puis d'effectuer une extrapolation, plus ou moins réaliste selon leur expérience. L'extrapolation effectuée entraîne cependant une marge d'erreur importante. Afin de réduire celle-ci, nous avons donc envisagé de remplacer l'estimation basée sur un comptage manuel par une mesure plus fiable obtenue par un logiciel de comptage autonome.

Le scénario de mise en place de ce logiciel est le suivant. Un client demande à l'entreprise Atos Origin d'évaluer le potentiel commercial d'un site (comme par exemple une devanture commerciale). Pour répondre à ce besoin, une solution informatique est mise en place chez le client. Cette solution consiste en un micro-ordinateur de type PC, une webcam, ainsi qu'un logiciel autonome permettant l'obtention de statistiques de fréquentation du site en question, par analyse des images acquises à l'aide de la webcam. La caméra, statique, est placée à l'intérieur d'une vitrine et perpendiculairement à la vitre de manière à obtenir une représentation de l'exterieur de la boutique.

Le but de l'application est alors d'estimer le nombre de passages devant la caméra, c'est-à-dire devant la boutique, ou encore devant le potentiel DAB. Si le DAB était effectivement installé, le nombre de passages aurait une influence directe sur le nombre de retraits effectués. Après discussion avec les différents intervenants, la définition d'un passage a été donnée de

la manière suivante : une personne qui entre dans le champ de vision de la caméra (donc qui apparaît dans les images de la séquence vidéo), qui se déplace dans une zone dite centrale et qui finalement quitte le champ de vision de la caméra.

Une fois le contexte de l'application présenté, il nous est maintenant possible de justifier l'architecture et les outils que nous avons proposés pour résoudre ce problème. Nous cherchons ici à mesurer le nombre de passages devant une caméra statique. Afin de résoudre ce problème, nous proposons un système spécifique d'analyse vidéo basé sur l'utilisation des outils génériques présentés dans les parties précédentes.

La séquence vidéo n'étant composée que d'un seul plan, nous ne ferons pas appel ici aux techniques de détection de changement de plans présentées dans la partie I. Nous n'aborderons donc que les problèmes décrits dans les parties II et III : séparation du fond et des objets, structuration du fond, et suivi des objets.

Pour résoudre le problème posé, nous avons mentionné précédemment deux versions du logiciel. En effet, à la demande d'Atos Origin, une première version, utilisée comme prototype, devait être fournie dans un laps de temps relativement court. Nous avons donc élaboré dans un premier temps un système relativement simple, basé sur les algorithmes connus dans la littérature. Comme nous le verrons par la suite, ce système ne répond pas entièrement aux attentes de l'entreprise. Un second système est donc en cours d'élaboration. Celui-ci intègre des outils plus performants, dérivés notamment des travaux présentés dans les chapitres précédents. Comparativement à la première version "simple" (ou système "prototype"), ce système dit "évolué" permet un traitement plus fiable et plus robuste des séquences vidéo. Par la suite, nous présenterons donc les deux solutions correspondant aux deux versions du logiciel.

### 9.2.2 Système "prototype"

Le système d'indexation proposé pour résoudre le problème posé repose sur l'architecture décrite dans la figure 1. Un seul plan est présent ici. De plus, les canaux associés aux données audio et textuelles ne seront pas étudiés. Le problème à résoudre peut alors se décomposer en 4 sous-problèmes qui sont :

- l'extraction des objets,
- la modélisation de la scène,
- le suivi des objets,
- et l'interprétation du contenu.

La première étape concerne la séparation du fond et des objets. La caméra étant statique et l'arrière-plan relativement hétérogène, nous avons eu recours à une méthode plus simple que

celle introduite dans le chapitre 3.

Le traitement suivant consiste en la détermination des zones d'intérêt dans l'image. Il est possible, puisque la caméra est statique, de préciser ces zones directement sur les images de la séquence analysée.

Il est alors possible d'effectuer un suivi des objets. Disposant d'images binaires obtenues après séparation du fond et des objets, nous avons choisi d'effectuer un étiquetage des objets et de suivre ceux-ci en appariant leurs caractéristiques d'une image à l'autre.

Finalement, la dernière étape concerne l'interprétation du contenu de la séquence. Cette interprétation doit permettre de détecter et de notifier la présence d'un événement (ici un passage). Elle est directement liée à la définition d'un passage.

Nous pouvons illustrer la première version du logiciel d'obtention de statistiques de fréquentation d'un site par la figure 9.1.

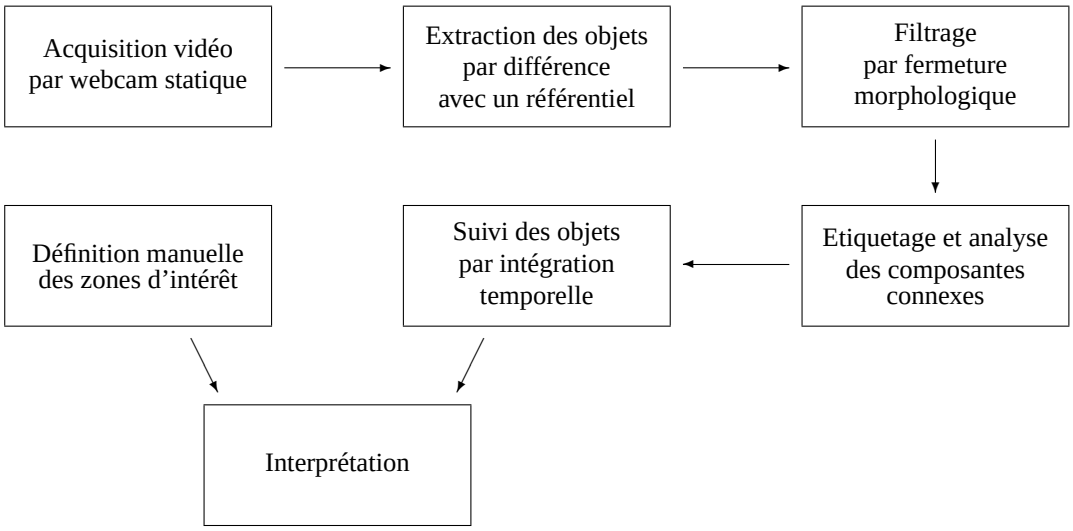


FIG. 9.1 – Architecture permettant l'obtention de statistiques de fréquentation d'un site (première version).

Nous donnons maintenant une description plus précise des différents outils utilisés.

**Extraction des objets**

L'extraction des objets est ici la premiere étape de la chaîne de traitement de la séquence vidéo. Elle a pour but de fournir, à partir des images originales en couleur, des images binaires représentant les objets (c.f. chapitre 3).

Dans le cas le plus simple, il est possible de considérer qu'à un instant donné aucun objet n'est présent dans la scène. L'image obtenue à cet instant est logiquement utilisée comme image de référence. Les images suivantes de la séquence sont alors comparées à l'image de référence pour déterminer les pixels correspondant à des objets. Pour cela, la méthode illustrée par l'équation (3.1) page 44 est utilisée.

Puisque le résultat est relativement sensible à la valeur de seuil $S_{\text{diff}}$ utilisée, et que la sensibilité du capteur (webcam) peut être importante, il est nécessaire de filtrer le résultat obtenu. Nous utilisons ici une succession d'opérations de morphologie mathématique pour éliminer les zones correspondant à du bruit. Plus précisément, nous utilisons une fermeture par un élément structurant de rayon 2. Les résultats obtenus sont satisfaisants.

**Modélisation de la scène**

Le second traitement effectué concerne la modélisation de la scène. Celle-ci permet la définition des zones d'intérêt dans l'arrière-plan de la scène. Ainsi, lorsqu'un objet donné apparaîtra dans une zone donnée de l'arrière-plan, un certain événement sera détecté.

Nous considérons ici deux zones d'intérêt différentes. La première zone, dite d'entrée et de sortie, représente la portion de l'arrière-plan réellement représentée dans l'image. En effet, la caméra étant placée à l'intérieur de la boutique du client, il est possible que certains objets (comme par exemple une porte) cachent une partie du champ de vision de la caméra. Seuls les mouvements des objets présents dans cette zone seront interprétés par la suite. De plus, en connaissant les limites réelles de la scène, il est possible de déterminer plus fidèlement les entrées et les sorties des objets : à gauche, à droite, en face, *etc.*. La figure 9.2 illustre l'intérêt de la définition d'une telle zone.

FIG. 9.2 – Les 2 zones d'intérêt : zone d'entrée et de sortie (en bleu) et zone centrale (en rouge).

La seconde zone, dite centrale, est utilisée pour calculer le nombre de passages. En effet, à chaque fois qu'un objet présent dans la zone d'entrée et de sortie, entre dans la zone centrale, il

est étiqueté comme valide. Lorsque cet objet quittera la scène (ou la zone d'entrée et de sortie), un passage supplémentaire sera comptabilisé. Afin de vérifier qu'un objet est présent dans la zone centrale, nous considérons ses coordonnées horizontale et verticale dans l'image. Son centre de gravité horizontal doit appartenir à la zone centrale, tandis que la partie culminante (au niveau vertical) de l'objet doit dépasser la zone centrale. Ce test supplémentaire relatif à la position verticale de l'objet nous permet d'éliminer les candidats potentiels que sont les animaux ou les enfants (même si ces derniers seront certainement un jour porteurs d'une carte de retrait !).

Pour définir ces deux zones, la solution la plus simple consiste à noter manuellement les positions des zones sur l'image.

#### Suivi des objets

Le résultat de l'extraction des objets, c'est-à-dire de la séparation des objets et du fond, est utilisé dans les différentes images pour effectuer le suivi des objets. Ce suivi fournit la position des différents objets dans les images successives de la séquence vidéo.

Puisque nous disposons d'images binaires représentant les objets de la scène, il est possible d'effectuer un étiquetage en composantes connexes afin d'obtenir les différents objets présents dans la scène. Les séquences vidéo acquises à l'aide de la caméra de type webcam étant relativement bruitées, nous avons chosi d'éliminer les régions dont l'aire serait inférieure à un certain seuil $S_{\text{aire}}$.

Nous disposons alors, pour chaque image de la séquence, d'un ensemble d'objets binaires caractérisés par leur position (représentée par leur centre de gravité) et leur aire dans l'image. Nous effectuons alors pour chaque couple d'images successives un appariement bilatéral entre les différents objets détectés. Chaque objet d'une image est apparié avec l'objet le plus proche (en terme de position et de taille) dans l'image précédente et dans l'image suivante. Si aucun objet n'est considéré comme proche, l'appariement de l'objet concerné échoue. Les liens entre objets appartenant à des images successives sont alors simplifiés en ne considérant que les appariements bilatéraux. Lorsque deux objets sont en concurrence, l'objet le plus ancien (c'est-à-dire présent depuis le plus grand nombre de trames) est favorisé.

#### Interprétation

Lorsque les objets ont été détectés et suivis dans les différentes images de la séquence, et que les zones d'intêret ont pu être localisées sur ces mêmes images, il est finalement possible

d'interpréter le contenu de la scène et de détecter les événements prédéfinis (ici les passages).

Pour cela des règles spécifiques à l'application sont définies. Ici, étant donné la définition d'un passage, l'interprétation consiste en quatre étapes que sont la détection des objets entrants, sortants, présents dans la zone centrale, et finalement l'augmentation du nombre de passages le cas échéant. Ces étapes utilisent les règles décrites ci-dessous.

Nous considérons qu'un objet entre dans la scène s'il n'a été apparié avec aucun objet de l'image précédente. De même, un objet est noté comme sortant lorsqu'il n'a été apparié avec aucun objet de l'image suivante. Dans les deux cas, la position de l'objet est comparée avec les limites de la zone d'entrée et de sortie, pour connaître l'origine ou la destination de l'objet : à gauche, à droite, ou en face de la boutique. Pour déterminer les objets présents dans la zone centrale, nous utilisons la règle énoncée précédemment qui stipule qu'un objet doit respecter deux conditions pour être validé : son centre de gravité doit être situé dans la zone centrale, et certains pixels de l'objet doivent être au-dessus de la limite haute de la zone centrale.

Finalement, le nombre de passages est incrémenté à chaque fois qu'un objet est entré dans la zone d'entrée et de sortie, puis a été validé dans la zone centrale, et est finalement sorti de la zone d'entrée et de sortie.

Le système "prototype" décrit ici, réalisé en un laps de temps relativement court, est basé sur des algorithmes connus dans la littérature. Il a été validé par un ensemble d'expérimentations afin de déterminer ses limites.

#### Résultats

De manière à faciliter la maintenance, et à la demande d'Atos Origin, le logiciel a été réalisé avec le langage Java. La bibliothèque Java Media Framework (JMF), permettant notamment le traitement et l'affichage de données multimédia, a donc été utilisée. Chacune des différentes étapes de traitement a été implémentée sous la forme d'une classe `Effect`.

La figure 9.3 présente les résultats obtenus à l'issue des différentes étapes, et ceci pour plusieurs images d'une même séquence.

Après une première batterie de tests dans un environnement fermé (scène d'intérieur), le taux d'erreur (estimé par comparaison avec un comptage manuel) a été évalué à moins de 8 %.

Cependant, les limites connues des algorithmes utilisés dans le système laissent penser que le taux d'erreur sera bien supérieur dans le cas d'une scène réelle d'extérieur. De plus, le système nécessite dans son état actuel un paramétrage manuel assez important. Il ne répond pas complètement aux attentes des utilisateurs. Nous proposons donc un système "évolué" tirant

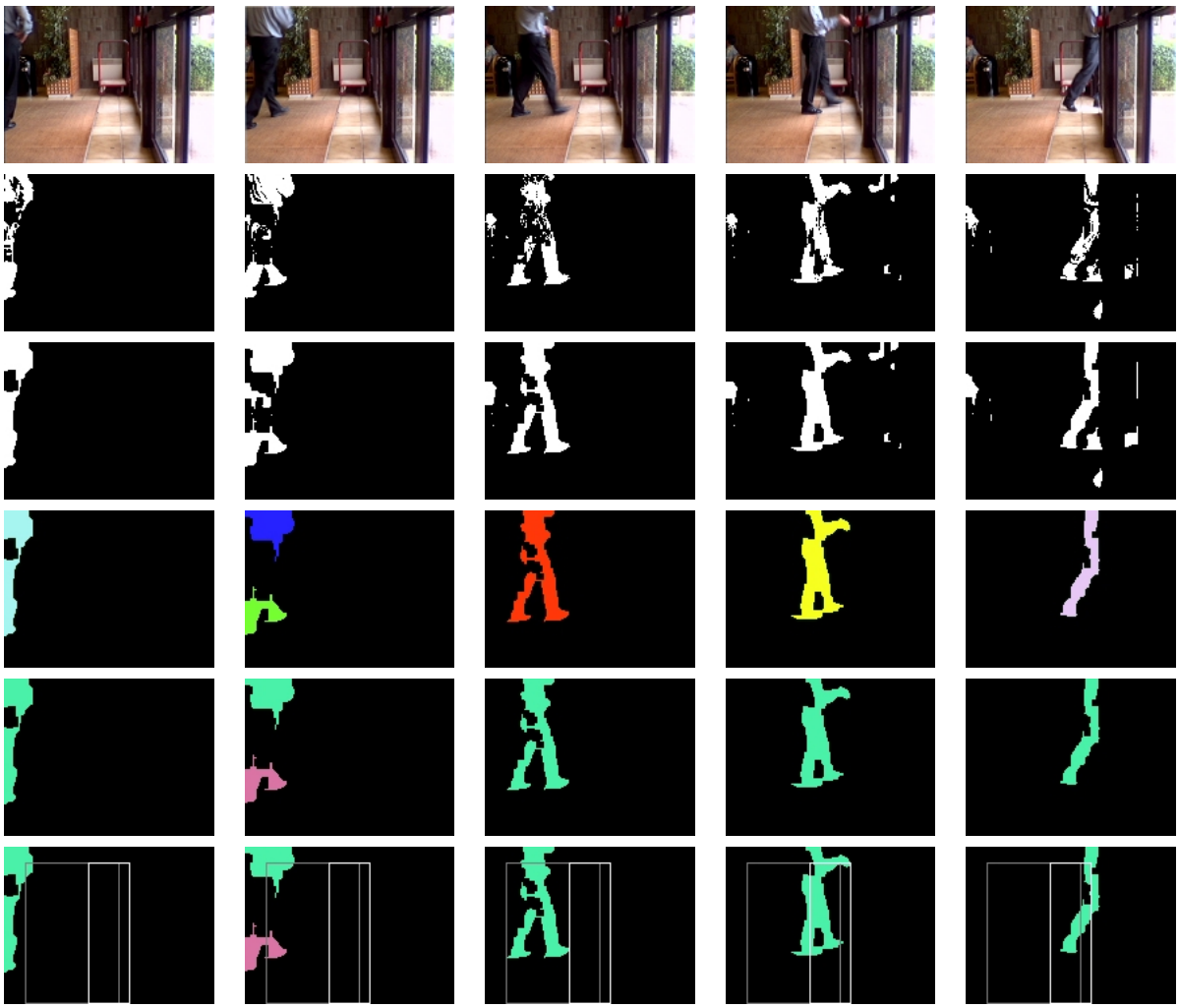

FIG. 9.3 – Images obtenues (extrait d'une séquence, de gauche à droite) après les différentes étapes de traitement (de haut en bas) : image originale, différence avec l'image de référence, filtrage morphologique, étiquetage et analyse des composantes connexes, suivi des objets, et interprétation.

parti des outils présentés dans les chapitres précédents de ce mémoire.

### 9.2.3 Système "évolué"

Après avoir vérifié rapidement la faisabilité de l'application *via* la réalisation d'un système "prototype", nous avons dû élaborer un système plus complexe, dit "évolué", qui permettait de combler les lacunes du premier système. Pour cela, nous avons utilisé les outils proposés dans cette thèse et présentés dans les chapitres précédents. Nous décrivons ici ce système en suivant la même présentation que précédemment. Ainsi, nous abordons les différents problèmes

successivement.

La séquence vidéo acquise par la caméra représente une scène d'extérieur. Dans cette scène, de forts changements d'éclairement peuvent donc apparaître au cours de la journée. La séparation du fond et des objets, pour être de meilleure qualité et plus robuste à ce type d'artefact, doit donc considérer un modèle adaptatif de l'arrière-plan.

La structure du fond peut, comme il a été énoncé précédemment, nous permettre de déterminer les zones d'intérêt dans l'image. Les limites de celles-ci seront représentées par les lignes ou segments de droites détectés dans les images.

Les objets à suivre dans la séquence représentent des corps humains. Ce sont donc des objets non-rigides qui peuvent être suivis avec la méthode basée sur les *snakes* (chapitre 7).

L'interprétation du contenu de la séquence est, quant à elle, identique à celle utilisée lors de la première version.

Nous pouvons modéliser le système présenté ici par la figure 9.4.

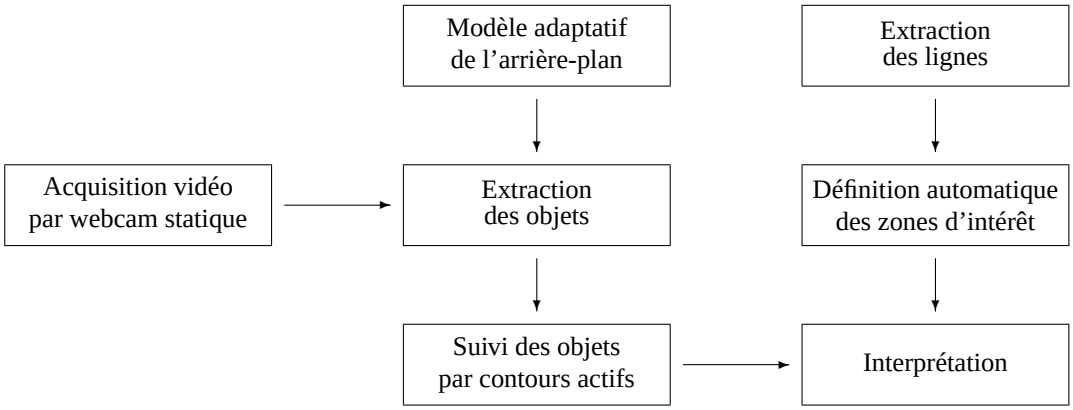


FIG. 9.4 – Architecture permettant l'obtention de statistiques de fréquentation d'un site (seconde version).

Nous donnons maintenant plus de détails sur la manière dont nous proposons de résoudre les différents problèmes.

**Extraction des objets**

La méthode présentée dans le système "prototype" est limitée par sa sensibilité aux conditions d'éclairement. Puisque la séquence vidéo représente une scène d'extérieur, il est nécessaire d'utiliser un modèle adaptatif de l'arrière-plan. Différentes techniques peuvent alors être utilisées. On peut citer notamment le moyennage itératif de chaque pixel ou une modélisation plus

robuste de chaque pixel du fond par un mélange de gaussiennes [Sta99]. Ainsi, les variations de l'arrière-plan sont progressivement intégrées au modèle.

Il est également possible d'utiliser la méthode introduite dans le chapitre 3, relativement robuste aux changements d'éclairement. Ainsi, dans le cas de l'approche multirésolution, un changement d'éclairement modifiera les caractéristiques du fond à toutes les résolutions, et le résultat obtenu sera donc correct et indépendant du changement d'éclairement.

**Modélisation de la scène**

La solution présentée dans la première version, qui consiste à noter manuellement les positions des lignes sur l'image présente deux inconvénients. Si la caméra est déplacée au sein de la boutique, il est nécessaire de redéfinir manuellement les positions des zones. De plus, à chaque installation dans une boutique, l'intervention d'un utilisateur est obligatoire.

Pour pallier ces limites, il est également possible de définir ces zones à l'aide de la structure du fond. En effet, les bords de la zone d'entrée et de sortie sont représentés dans l'image par des lignes, pouvant être détectées par les méthodes décrites dans le chapitre 4. Si l'on dispose d'un modèle générique de vitrine, la localisation de cette zone dans l'image peut être effectuée automatiquement. La zone centrale, quant à elle, peut également être positionnée automatiquement en fonction de la zone d'entrée et de sortie. Dans ce cas, des informations *a priori* sont obtenues en consultant les utilisateurs. Ceux-ci pourront donner par exemple la définition suivante : «la zone centrale est située dans le tiers central de la zone d'éntrée et de sortie». Par ce moyen, une installation purement automatique peut être envisagée.

**Suivi des objets**

Les objets suivis étant non-rigides, il est également possible d'utiliser la méthode basée sur les *snakes* (chapitre 7). Cette méthode doit cependant être modifiée pour prendre en compte les images disponibles dans ce contexte. En effet, aucune garantie quant à l'homogénéité de l'arrière-plan n'est donnée, ce qui empêche l'utilisation de l'énergie $E_{\text{coul}}$ liée à l'arrière-plan et basée sur sa couleur. Cependant, nous disposons d'une image binaire permettant de dissocier les objets du fond. L'énergie $E_{\text{coul}}$ peut donc être tout de même utilisée en considérant non pas l'image originale mais, pour cette énergie, l'image binaire obtenue après séparation du fond et des objets.

**Résultats**

Cette seconde version du logiciel est en cours de réalisation afin d'obtenir un système permettant une estimation de qualité en toute circonstance. Nous ne pouvons donc pas évaluer pour l'instant la qualité du système.

Une autre demande a été formulée par Atos Origin. Elle concerne la vidéosurveillance d'une salle informatique sécurisée. Puisque cette application n'est encore qu'au stade de projet, elle ne sera décrite que très brièvement dans la section suivante.

## 9.3 Vidéosurveillance d'une salle informatique sécurisée

Au sein de l'entreprise Atos Origin, la mise en place d'un second système d'indexation vidéo a été envisagée. Ce besoin, formulé par l'équipe Recherche et Développement, concerne une application de vidéosurveillance d'une salle informatique sécurisée. Cette vidéosurveillance doit être effectuée par un logiciel autonome, contrairement à la surveillance par un opérateur, mise en place généralement.

Dans cette section, nous décrirons donc brièvement cette application. Après avoir expliqué le principe général d'un système de vidéosurveillance, nous détaillerons les contraintes et les besoins spécifiques à prendre en compte ici. L'architecture proposée sera ensuite commentée et les choix envisagés des différents algorithmes seront justifiés.

### 9.3.1 Problématique

La surveillance de sites basée sur l'analyse de séquence vidéo fait l'objet de nombreuses études menées actuellement [Gav99]. Le lecteur intéressé pourra notamment se référer à plusieurs thèses [Cas98, Die00, Zha01, Zil00] ou ouvrages [For00, Rem01, Reg99] récents. L'intérêt des scientifiques pour ce domaine peut notamment être expliqué par des événements récents [Meh02]. Dans ce type d'applications, la contrainte de temps réel $\mathcal{C}_{\text{rapidité}}$ est très souvent critique et il est donc naturel de trouver des numéros spéciaux dans des revues spécialisées [For01] et de nombreux articles traitant ce problème [Har00, Kan02b]. De plus, les spécificités propres à ce type d'applications nécessitent parfois l'utilisation de techniques dédiées : extraction des objets [McK00], suivi des objets [Bak99, For99], interprétation de leurs mouvements et actions [Dat02, Rot00], *etc.*.

Le besoin d'Atos Origin s'inscrit dans ce type d'applications. Le système d'indexation vidéo

doit en effet permettre, en temps réel, de détecter certains événements puis de les notifier à un poste de contrôle. Ces événements, définis au préalable, représentent des occurrences de situations anormales, qui nécessitent d'alerter les personnes compétentes. Le site à surveiller est une salle informatique sécurisée. Cette salle est découpée en différentes zones grillagées, et dans chaque zone est installé un serveur contenant des données critiques. La salle est dite sécurisée car l'entrée nécessite que les identités de deux personnes habilitées soient vérifiées simultanément par des technologies biométriques.

Ces mesures de sécurité peuvent paraître suffisantes, mais en réalité elles ne permettent pas de garantir une sécurité absolue. Ainsi, au moins deux cas de figure peuvent générer une situation anormale qu'il est nécessaire de détecter. Le système d'indexation aura donc pour but de détecter, en temps réel, les événements associés à ces cas de figure.

Le premier événement à détecter concerne le nombre de personnes entrant dans la salle sécurisée. Le système biométrique nécessite la présence de 2 personnes habilitées mais n'empêche pas l'entrée d'autres personnes non autorisées. Il est donc nécessaire de compter le nombre de personnes entrant dans la salle, et de déclencher une alarme au cas où ce nombre serait supérieur à celui autorisé.

L'entrée de personnes habilitées dans la salle fait suite à une demande d'intervention précise, formulée *a priori*, et qui ne concerne qu'un et un seul serveur. Les intervenants ne sont autorisés qu'à se déplacer dans la zone correspondant au serveur concerné. Il est donc nécessaire de vérifier que les personnes entrant dans la salle n'effectuent pas (ou ne tentent pas d'effectuer) des manipulations sur les autres serveurs, dans les autres zones. Le second événement à détecter est donc lié à la présence d'une ou plusieurs personnes dans une zone non autorisée.

La liste des événements à détecter est actuellement limitée à ces deux événements. Cependant, il est possible que de nouveaux besoins nécessitent la détection d'autres événements. Le système d'indexation vidéo doit donc être évolutif, et son architecture doit permettre l'ajout de nouveaux composants.

### 9.3.2 Architecture proposée

L'application de vidéosurveillance d'une salle informatique sécurisée nécessite l'utilisation des outils génériques présentés précédemment et intégrés de manière spécifique pour permettre la détection des deux événements prédéfinis : "le nombre de personnes entrant sur le site est supérieur au nombre autorisé", et "la zone d'intervention est différente de celle prévue".

Nous présentons ici les justifications théoriques concernant les choix d'architecture et les

différents algorithmes proposés.

Les séquences vidéo analysées ne contiennent là aussi qu'un seul plan, ce qui rend inutile les techniques présentées dans la partie I. Les problèmes abordés dans les parties II et III doivent cependant être résolus afin de permettre la détection des événements.

### Extraction des objets

Ainsi, il est tout d'abord nécessaire de séparer les objets et le fond de la scène. Contrairement à l'application précédente, nous considérons ici une scène d'intérieur avec des éclairages artificiels. Les conditions d'éclairement sont donc plus constantes tout au long de la séquence, et il n'est pas nécessaire d'utiliser un modèle adaptatif. Cependant, la ou les caméras dédiées à l'acquisition peuvent être en mouvement afin de couvrir l'ensemble de la salle. Dans ce cas, il est nécessaire d'effectuer une compensation de mouvement afin de calculer la différence avec une image de référence. De façon alternative, il est également possible d'utiliser l'approche présentée dans le chapitre 3 si l'arrière-plan est relativement homogène. Puisque la scène représente une salle informatique, cette condition peut facilement être respectée, l'arrière-plan étant constitué par un mur de la pièce (celui opposé à la caméra).

### Modélisation de la scène

L'analyse de l'arrière-plan de la scène peut également permettre d'obtenir un modèle de la structure de celui-ci. Le modèle obtenu est alors utilisé pour définir les différentes zones d'intervention. Puisqu'il s'agit d'un environnement artificiel, des primitives telles que les lignes peuvent être utilisées pour obtenir la structure du fond. Ces lignes sont détectées avec la méthode décrite dans le chapitre 4. On pourra de plus se limiter aux lignes horizontales et verticales si l'on cherche à analyser des plans parallèles à celui de la caméra.

### Suivi des objets

Une fois l'arrière-plan analysé, les objets peuvent être étudiés. L'étape d'extraction des objets nous fournit les positions des différents objets dans les images de la séquence. Le suivi peut alors être effectué en utilisant des outils adaptés, tels les *snakes* présentés dans le chapitre 7. En effet, les personnes intervenant dans la salle sont considérées comme des objets non-rigides. Les occlusions partielles pouvant survenir sont traitées par le module de scission décrit dans la section 7.4.

**Interprétation**

Finalement, une fois la structure du fond connue et les positions des objets obtenues dans les différentes images de la séquence, la dernière étape consiste en l'interprétation du contenu de la séquence.

Le premier événement nécessite de compter, dans chaque image, le nombre d'objets suivis. Afin de limiter la sensibilité au bruit, on pourra considérer uniquement les objets présents depuis un nombre d'images donné.

Afin de vérifier la zone d'intervention des personnes présentes dans la salle, une comparaison de la position de celles-ci avec la position dans l'image des différentes zones est effectuée. Si un objet (c'est-à-dire une personne) est situé dans une zone non autorisée, l'événement est détecté.

Les différents éléments décrits précédemment peuvent finalement être regroupés afin de construire le système d'indexation permettant la détection des événements prédéfinis. La figure 9.5 présente l'architecture générale d'un tel système.

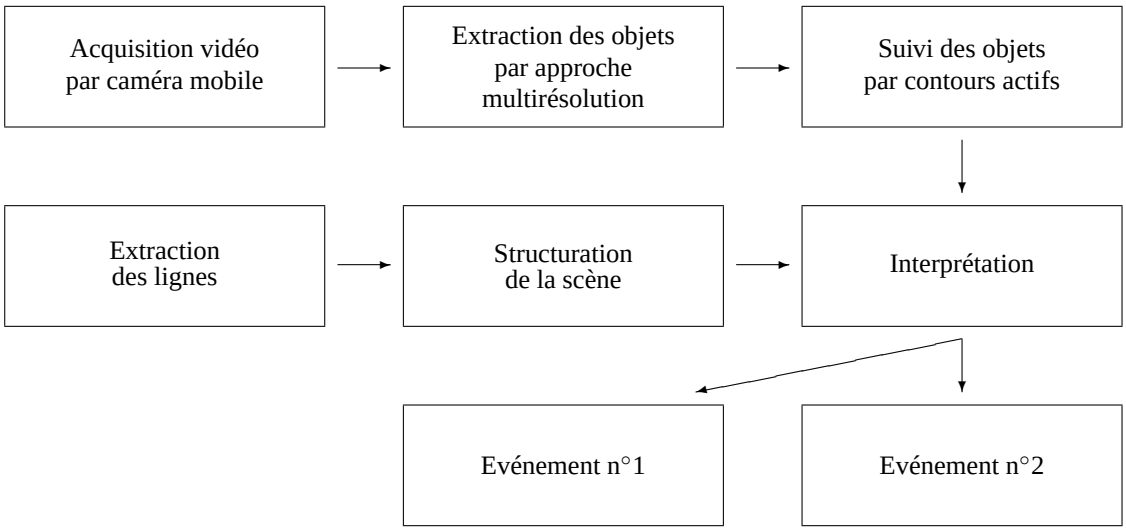


FIG. 9.5 – Architecture permettant la vidéosurveillance d'une salle informatique sécurisée.

## 9.4 Conclusion

Dans ce chapitre, nous avons présenté deux applications permettant d'illustrer le cas d'une source vidéo contrôlée. Ce cas, plus facile à traiter que le cas envisagé dans le chapitre pré-

cédent, permet parfois l'utilisation de techniques plus simples que celles présentées dans les parties précédentes. Pourtant, celles-ci pourraient également être utilisées, afin d'améliorer notamment l'efficacité des systèmes d'indexation vidéo.

# Conclusion et perspectives

## Conclusion

Dans cette thèse, nous avons abordé le problème de l'**indexation multimédia**. Plus précisément, nous cherchions à détecter des événements prédéfinis dans des séquences vidéo, en imposant au processus un faible temps de calcul, tout en proposant des solutions les plus génériques possibles.

Notre étude s'est donc concentrée sur la **définition d'une architecture et l'élaboration d'outils génériques pour la mise en œuvre de systèmes spécifiques d'indexation vidéo**. Cependant ce travail reste trop ambitieux pour être mené à bien dans sa totalité, dans le cadre d'une thèse.

Ainsi, nous avons dû tout d'abord effectuer un **découpage en plans** de la séquence vidéo. Après une étude de la littérature (chapitre 1), les contraintes qui nous étaient imposées nous ont amené à choisir l'espace couleur TSL pour élaborer une nouvelle méthode de détection des changements de plans (chapitre 2), dont le caractère adaptatif permet de gérer différents types de séquences vidéo.

Disposant de plans, une approche multirésolution nous a permis, en évitant de rechercher explicitement les informations de mouvement, de réaliser une **séparation entre fond et objets** (chapitre 3), rendant ainsi possible l'initialisation sous-jacente des étapes de suivi. D'autre part, faisant l'hypothèse que la **structure du fond** peut être obtenue en considérant les lignes présentes dans les images, nous avons proposé une approche rapide et semi-locale pour résoudre le problème de la détection des lignes (chapitre 4).

Après avoir étudié l'arrière-plan de la scène, nous nous sommes focalisé sur les objets en cherchant à les suivre dans les séquences vidéo. Pour conserver la propriété de généricité, nous avons choisi d'utiliser **différentes méthodes selon le type d'objet suivi**. Ainsi, dans le cas des objets rigides pouvant faire l'objet d'un apprentissage, nous avons utilisé un des outils

développés au laboratoire, les chaînes de Markov cachées multdimensionnelles à processus indépendant (chapitre 5). Nous avons observé que ces techniques ne donnaient pas de meilleurs résultats qu'un algorithme plus simple de *template matching*. En ce qui concerne le suivi des objets rigides sans apprentissage, et plus précisément des objets de petite taille (chapitre 6), et afin de minimiser le temps de calcul nécessaire au suivi, nous avons remarqué là encore l'intérêt des approches locales comparées à des approches globales. Dans le cas d'objets non-rigides, notre contribution a consisté en l'élaboration d'une méthode rapide de suivi par contours actifs (chapitre 7) admettant une phase de scission utile dans le cas d'objets proches et pouvant momentanément s'occulter.

Nous avons finalement validé notre contribution, c'est-à-dire une architecture et des outils pour la détection d'événements dans une séquence vidéo, sur des **applications réelles**. La première a pour objectif de détecter des buts inscrits dans un match de football (chapitre 8). Pour résoudre ce problème complexe, nous avons en outre proposé d'analyser le son à l'aide de méthodes adéquates. D'autres applications, faisant suite aux demandes de l'entreprise Atos Origin, nous ont également amené à élaborer deux systèmes d'indexation vidéo (chapitre 9) permettant l'obtention de statistiques relatives à la fréquentation d'un site et la vidéosurveillance d'une salle informatique sécurisée. Nous avons ainsi considéré le cas plus simple d'une caméra statique.

# Perspectives

En se basant sur l'architecture et les outils proposés dans cette thèse, il est dorénavant possible de mettre en œuvre un système rapide d'indexation vidéo par détection d'événements. Pourtant, nous souhaitons maintenant intégrer les différents outils au sein de l'architecture proposée, afin de disposer d'un système complet de détection d'événements : c'est d'ailleurs la perspective la plus importante de notre travail.

Parmi les autres orientations que nous désirons suivre, la **gestion des scènes caractérisées par un arrière-plan non uniforme** nous semble prioritaire. Elle permettrait de traiter correctement tous les types de situation, et assurerait ainsi un niveau supérieur de généricité.

Tout au long de cette thèse, nous avons cherché des solutions de faible complexité algorithmique aux différents problèmes posés. Par conséquent, nous disposons d'outils qui peuvent chacun traiter des images à la même fréquence que l'acquisition (25 images par seconde, soit 40 millisecondes par image). Cependant, le système global, consistant en une intégration de l'ensemble des outils, est caractérisé par un temps de calcul plus important. Pour le diminuer,

nous envisageons une **implémentation des algorithmes sur une architecture parallèle** (station multiprocesseurs ou *cluster* de stations). Nous pourrions également formuler tous les algorithmes utilisés selon un cadre multirésolution [Jol90].

# Publications

## Articles dans des revues internationales

Sébastien LEFÈVRE, Jérôme HOLLER et Nicole VINCENT, A Study of Real-Time Segmentation of Uncompressed Video Sequences for Content-Based Search and Retrieval, *Real Time Imaging*, accepté pour publication

## Communications dans des congrès internationaux avec comité de lecture

Sébastien LEFÈVRE, Jérôme HOLLER et Nicole VINCENT, Real Time Temporal Segmentation of Compressed and Uncompressed Dynamic Colour Image Sequences, *International Workshop on Real Time Image Sequence Analysis*, Oulu, Finlande, Août 2000, pages 56–62

Anne-Sophie CAPELLE, Olivier ALATA, Christine FERNANDEZ, Sébastien LEFÈVRE et Jean-Claude FERRIÉ, Unsupervised segmentation for automatic detection of brain tumors in MRI, *IEEE International Conference on Image Processing*, Vancouver, Canada, Septembre 2000

Sébastien LEFÈVRE, Cyril FLUCK, Benjamin MAILLARD et Nicole VINCENT, A Fast Snake-based Method To Track Football Players, *IAPR International Workshop on Machine Vision Applications*, Tokyo, Japon, Novembre 2000, pages 501–504

Sébastien LEFÈVRE, Loïc MERCIER, Vincent TIBERGHIEN et Nicole VINCENT, Multiresolution color image segmentation applied to background extraction in outdoor images, *IS&T European Conference on Color in Graphics, Image and Vision*, Poitiers, France, Avril 2002, pages 363–367

Sébastien LEFÈVRE, Benjamin MAILLARD et Nicole VINCENT, 3 Class Segmentation for Analysis of Football Audio Sequences, *IEEE International Conference on Digital Signal Processing*, Santorin, Grèce, Juillet 2002, pages 975–978

Sébastien LEFÈVRE, Charles DIXON, Cédric JEUSSE et Nicole VINCENT, A Local Approach for Fast Line Detection, *IEEE International Conference on Digital Signal Processing*, Santorin, Grèce, Juillet 2002, pages 1109-1112

Sébastien LEFÈVRE, Benjamin MAILLARD et Nicole VINCENT, A Two Level Classifier Process for Audio Segmentation, *IAPR International Conference on Pattern Recognition*, Québec, Canada, Août 2002, pages 891–894

Sébastien LEFÈVRE, Jean-Pierre GÉRARD, Aurélie PIRON et Nicole VINCENT, An Extended Snake Model for Real-Time Multiple Object Tracking, *International Workshop on Advanced Concepts for Intelligent Vision Systems*, Gand, Belgique, Septembre 2002, pages 268–275

## Communications dans des congrès nationaux avec comité de lecture

Sébastien LEFÈVRE, Jérôme HOLLER et Nicole VINCENT, Segmentation temporelle de séquences d'images en couleur compressées et non-compressées en temps réel, *Congrès Francophone ORASIS de Vision par Ordinateur*, Cahors, France, Juin 2001, pages 329–338

Sébastien LEFÈVRE, Cyril FLUCK, Benjamin MAILLARD et Nicole VINCENT, Un modèle de contour actif pour le suivi rapide d'objets en mouvement. Application au suivi de joueurs de football, *Colloque GRETSI sur le Traitement du Signal et des Images*, Toulouse, France, Septembre 2001

## Communications dans des groupes nationaux, séminaires, forums

Sébastien LEFÈVRE, Elaboration et validation d'un outil automatique de segmentation et de reconstruction volumique sur imagerie IRM de tumeurs cérébrales, *Séminaire du Laboratoire d'Informatique de l'Université de Tours*, Tours, France, Décembre 1999

Sébastien LEFÈVRE, Détection temps-réel de changements de plans dans une séquence d'images couleur, *Journées Jeunes Chercheurs du GDR-ISIS*, Paris, France, Mai 2000, pages 105–106

Sébastien LEFÈVRE, Indexation de séquences vidéo en temps-réel et transmission d'extraits pertinents relatifs à une problématique donnée, *Journées de l'Ecole Doctorale SST de l'Université de Tours*, Tours, France, Mai 2001

Sébastien LEFÈVRE, Détection temps-réel de changements de plans basée sur l'espace HSV, *Réunion du GDR-ISIS – GT10 sur l'orientation des recherches en indexation multimédia*, Paris, France, Mai 2001

Sébastien LEFÈVRE, Jérôme HOLLER et Nicole VINCENT, Indexation de séquences vidéo par détection des changements de plan, *Rencontres VISIOMIP Laboratoires – Entreprises en Vision par Ordinateur*, Cahors, France, Juin 2001, pages 24–25

Sébastien LEFÈVRE, Etude de l'arrière-plan d'une scène, *Séminaire du Laboratoire d'Informatique de l'Université de Tours*, Tours, France, Janvier 2002

Sébastien LEFÈVRE, Segmentation multi-résolution d'images couleur appliquée à l'extraction de l'arrière-plan dans des scènes d'extérieur, *Journées de l'Ecole Doctorale SST de l'Université de Tours*, Tours, France, Mai 2002

## Rapports internes

Sébastien LEFÈVRE, Emmanuel BOUTON, Thierry BROUARD et Nicole VINCENT, Multidimensional Hidden Markov Models for Object Tracking, Rapport interne n°255 du Laboratoire d'Informatique de l'Université de Tours, Mai 2002

Sébastien LEFÈVRE, Jérôme HOLLER et Nicole VINCENT, A Study of Real-Time Segmentation of Uncompressed Video Sequences, Rapport interne n°256 du Laboratoire d'Informatique de l'Université de Tours, Mai 2002

Sébastien LEFÈVRE, Jérôme HOLLER et Nicole VINCENT, A Review on Compressed Video Parsing, Rapport interne n°257 du Laboratoire d'Informatique de l'Université de Tours, Mai 2002

# Bibliographie


[Aas99] K. Aas, L. Eikvil, et R. B. Huseby. Applications of hidden markov chains in image analysis. *Pattern Recognition*, 32(4) :703–713, 1999.

[Ach98] S. Acharya et B. Smith. Compressed domain transcoding of mpeg. Dans *IEEE International Conference on Multimedia Computing and Systems*, pages 295–306, Austin, USA, Juillet 1998.

[Agb99] J.I. Agbinya et D. Rees. Multi-object tracking in video. *Journal of Real-Time Imaging*, 5(5) :295–304, 1999.

[Aha96] G. Ahanger et T.D.C. Little. A survey of technologies for parsing and indexing digital video. *Journal of Visual Communication and Image Representation*, 7(1) :28–43, Mars 1996.

[Aig96] P. Aigrain, H.J. Zhang, et D. Petkovic. Content-based representation and retrieval of visual media : A state-of-the-art review. *International Journal of Multimedia Tools and Applications*, 3(3) :179–202, Novembre 1996.

[Ame02] A. Amer, A. Mitiche, et E. Dubois. Context-independent real-time event recognition : Application to key-image extraction. Dans *IAPR International Conference on Pattern Recognition*, volume 2, Québec, Août 2002.

[Ami90] A. Amini, T. Weymouth, et R. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9) :855–867, Septembre 1990.

[Anc02] N. Ancona, G. Cicirelli, E. Stella, et A. Distante. Object detection in images : Run-time complexity and parameter selection of support vector machines. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[And88] E. André, G. Herzog, et T. Rist. On the simultaneous interpretation of real world image sequences and their natural language descriptions : the system soccer. Dans *European Conference on Artificial Intelligence*, pages 449–454, Munich, Août 1988.

[And94] E. André, G. Herzog, et T. Rist. Multimedia presentation of interpreted visual data. Dans *Workshop on Integration of Natural Language and Vision Processing*, pages 74–82, Seattle, Août 1994.

[And01] E. André et T. Rist. Presenting through performing : on the use of multiple life-like characters in knowledge-based presentation systems. *Knowledge-Based Systems*, 14(1/2) :3–13, Mars 2001.

[Ard00] M. Ardebilian, X. Tu, et L. Chen. Robust 3d clue-based video segmentation for video indexing. *Journal of Visual Communication and Image Representation*, 11(1) :58–79, Mars 2000.

[Ass00] Assavid : Automatic segmentation and semantic annotation of sports videos. Projet Européen du 5e PCRD, IST-1999-13082, Janvier 2000.

[Ass01] J. Assfalg, M. Bertini, C. Colombo, et A. Del Bimbo. Semantic characterization of visual content for sports videos annotation. Dans *International Workshop on Multimedia Databases and Image Communication*, pages 179–191, Amalfi, Italie, Septembre 2001.

[Ass02] J. Assfalg, M. Bertini, C. Colombo, et A. Del Bimbo. Semantic annotation of sports videos. *IEEE Multimedia*, 9(2) :52–60, Avril/Juin 2002.

[Aub99] G. Aubert et L. Blanc-Féraud. Some remarks on the equivalence between 2d and 3d classical snakes and geodesic active contours. *International Journal of Computer Vision*, 34(1) :19–28, Septembre 1999.

[Bak99] A. Bakowski et G.A. Jones. Video surveillance tracking using colour region adjacency graphs. Dans *IEE Conference on Image Processing and its Applications*, volume 465, pages 794–798, Manchester, UK, Juillet 1999.

[Bar02] M. Barlaud et C. Labit, éditeurs. *Compression et codage des images et vidéos*. Traité IC2-Série Traitement du Signal et de l'Image. Hermès Sciences Publications, 2002.

[Bat00] E. Battle et P. Cano. Automatic segmentation for music classification using competitive hidden markov models. Dans *International Symposium on Music Information Retrieval*, Plymouth, MA, Octobre 2000.

[Bau67] L. E. Baum et J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. American Society*, 73 :360–363, 1967.

[Beb98] T. Bebie et H. Bieri. Soccerman : reconstructing soccer games from video sequences. Dans *IEEE International Conference on Image Processing*, volume 1, pages 898–902, Chicago, 1998.

[Beb00] T. Bebie et H. Bieri. A video-based 3d-reconstruction of soccer games. *Eurographics Computer Graphics Forum*, 19(3) :391–400, 2000.

[Beu02] R. Beun. Image based camera motion stabilization using static features in the distant scene. Dans *International Workshop on Advanced Concepts for Intelligent Vision Systems*, pages 276–283, Gand, Belgique, Septembre 2002.

[Bim99] A. Del Bimbo, V. Castelli, S.F. Chang, et C.S. Li, éditeurs. *Computer Vision and Image Understanding, Special Issue on Content-Based Access of Image and Video Libraries*, 75(1-2), Juillet/Août 1999.

[Bla99] A. Blake et M. Isard. *Active Contours : The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag, 1999.

[Bob95] A.F. Bobick. Video annotation : Computers watching video. Dans *Asian Conference on Computer Vision*, volume 1, pages 19–23, Singapour, Décembre 1995.

[Boc93] E.L. Bocchieri et J.G. Wilpon. Discriminative feature selection for speech recognition. *Computer Speech & Language*, 7(3) :229–246, Juillet 1993.

[Bon95] J.F. Bonnet et R. Samy. Développement de contours actifs robustes et adaptatifs et application à la poursuite d'objets dans une séquence d'images infrarouges. Dans *Colloque GRETSI sur le Traitement du Signal et des Images*, pages 657–660, Juan-les-Pins, Septembre 1995.

[Bon97] J.F. Bonnet et R. Samy. Poursuite et reconnaissance d'objets dans une séquence infrarouge par contours actifs robustes incrémentaux et réseaux neuronaux. Dans *Colloque GRETSI sur le Traitement du Signal et des Images*, pages 1133–1136, Grenoble, Septembre 1997.

[Bon01] J. Bonnardot. *Contribution physico-mathématique à l'étude de l'interface perceptive forme-couleur en vision humaine. Approche par dilemmes interdisciplinaires*. Thèse d'état, Institut National des Sciences Appliquées de Lyon et Université Lyon 1, Juin 2001.

[Bor96] J.S. Boreczky et L.A. Rowe. Comparison of video shot boundary detection techniques. *Journal of Electronic Imaging*, 5(2) :122–128, Avril 1996.

[Bou99] P. Bouthemy, M. Gelgon, et F. Ganansia. A unified approach to shot change detection and camera motion characterization. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(7) :1030–1044, Octobre 1999.

[Bré97] F. Brémond. *Environnement de Résolution de Problèmes pour l'Interprétation de Séquences d'Images*. Thèse de doctorat, INRIA Sophia-Antipolis, Octobre 1997.

[Bra01] A. Branca, G. Cicirelli, N. Ancona, et A. Distante. Goal distance estimation in soccer game. Dans *IAPR International Conference on Image Analysis and Processing*, pages 565–571, Palerme, Septembre 2001.

[Bro97] T. Brouard, M. Slimane, J.P. Asselin de Beauville, et G. Venturini. Hybrid genetic learning of hidden markov models. Dans *AIDRI Conference on LEARNING – From Natural Principles to Artificial Methods*, pages 127–130, Geneva, Switzerland, Juin 1997.

[Bro98] T. Brouard, M. Slimane, J.P. Asselin de Beauville, et G. Venturini. Apprentissage d'une chaîne de markov cachée : Problèmes numériques liés à l'application à l'image. *Revue de Statistique Appliquée*, 46(2) :83–108, 1998.

[Bro99] T. Brouard. *Algorithmes Hybrides d'Apprentissage de Chaînes de Markov Cachées : Conception et Applications à la Reconnaissance de Formes*. Thèse de doctorat, University de Tours, France, Janvier 1999.

[Bru99] R. Brunelli, O. Mich, et C.M. Modena. A survey on the automatic indexing of video data. *Journal of Visual Communication and Image Representation*, 10(2) :78–112, Juin 1999.

[Bui01] H. H. Bui, S. Venkatesh, et G. West. Tracking and surveillance in wide-area spatial environments using the abstract hidden markov model. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1) :177–195, Février 2001.

[Cae01] T. Caelli, A. McCabe, et G. Briscoe. Shape tracking and production using hidden markov models. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1) :197–221, Février 2001.

[Car94] T. Carron et P. Lambert. Color edge detector using jointly hue, saturation and intensity. Dans *IEEE International Conference on Image Processing*, pages 977–981, Austin, USA, Novembre 1994.

[Car95] T. Carron. *Segmentations d'images couleur dans la base Teinte-Luminance-Saturation : approche numérique et symbolique*. Thèse de doctorat, Université de Savoie, Décembre 1995.

[Car98] B. Carlson. Interactive sports graphics from real video. *Advanced Imaging*, pages 28–30, Juillet 1998.

[Car00] J. Carrive. *Classification de Séquences Audiovisuelles*. Thèse de doctorat, Université Pierre et Marie Curie – Paris 6, Décembre 2000.

[Cas98] R. Castagno. *Video Segmentation Based on Multiple Features for Interactive and Automatic Multimedia Applications*. Thèse de doctorat, Ecole Polytechnique Fédérale de Lausanne, 1998.

[Ced95] C. Cedras et M. Shah. Motion-based recognition : A survey. *Image and Vision Computing*, 13(2) :129–155, 1995.

[Cha95] S.F. Chang et D.G. Messerschmitt. Manipulation and compositing of mc-dct compressed video. *IEEE Journal Selected Areas in Communications*, 13(1) :1–11, Janvier 1995.

[Cha96] Y.L. Chang, W. Zeng, I. Kamel, et R. Alonso. Integrated image and speech analysis for content-based video indexing. Dans *IEEE International Conference on Multimedia Computing and Systems*, pages 306–313, Hiroshima, Japon, Juin 1996.

[Cha01a] H.D. Chang, X.H. Jiang, Y. Sun, et J. Wang. Color image segmentation : Advances and prospects. *Pattern Recognition*, 34(12) :2259–2281, Décembre 2001.

[Cha01b] S.F. Chang, D. Zhong, et R. Kumar. Real-time content-based adaptive streaming of sports videos. Dans *IEEE International Workshop on Content-Based Access to Image and Video Libraries*, Kauai, USA, Décembre 2001.

[Che93] C.T. Chen. Video compression : Standards and applications. *Journal of Visual Communication and Image Representation*, 4(2) :103–111, Juin 1993.

[Che00] C. Chesneaud. *Techniques Statistiques de Segmentation par Contour Actif et Mise en Oeuvre Rapide*. Thèse de doctorat, Université de droit, d'économie, et des sciences d'Aix-Marseille, Février 2000.

[Che01a] Y. Chen, T.S. Huang, et Y. Rui. Optimal radial contour tracking by dynamic programming. Dans *IEEE International Conference on Image Processing*, Thessaloniki, Greece, Octobre 2001.

[Che01b] F. Cheng, W.J. Christmas, et J. Kittler. Extracting a feature of human running behaviour in sports video sequences. Dans *International Workshop on Information Retrieval*, pages 161–170, Oulu, Finlande, Septembre 2001.

[Che02a] D. Chen, J.M. Odobez, et H. Bourlard. Text segmentation and recognition in complex background based on markov random field. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Che02b] F. Cheng, W. Christmas, et J. Kittler. Recognising human running behaviour in sports video sequences. Dans *IAPR International Conference on Pattern Recognition*, volume 2, Québec, Août 2002.

[Che02c] F. Cheng, W.J. Christmas, et J. Kittler. A non-rigid foreground object detection algorithm for the application of sports video annotation and retrieval. Rapport Interne VSSP-TR-1/2002, University of Surrey, 2002.

[Cho01] W.P. Choi, K.M. Lam, et W.C. Siu. An adaptative active contour model for highly irregular boundaries. *Pattern Recognition*, 34(2) :323–331, Février 2001.

[Cie01] M. Ciesielska. Testing uniformity for circular data. Mémoire de maîtrise, West Virginia University, Mai 2001.

[Coh91] L.D. Cohen. On active contour models and balloons. *Computer Vision, Graphics and Image Processing : Image Understanding*, 53(2) :211–218, Mars 1991.

[Coh98] I. Cohen et G. Medioni. Detection and tracking of objects in airborne video imagery. Dans *Workshop on Interpretation of Visual Motion*, Santa Barbara, USA, 1998.

[Com95] M.C. Comer et E.J. Delp. Multiresolution image segmentation. Dans *International Conference on Acoustics, Speech, and Signal Processing*, pages 2415–2418, Detroit, MI, USA, Mai 1995.

[Dai95] A. Dailianas, R.B. Allen, et P. England. Comparison of automatic video segmentation algorithms. Dans *SPIE Conference on Integration Issues in Large Commercial Media Delivery Systems*, volume 2615, pages 2–16, Philadelphia, PA, Octobre 1995.

[Dat02] A. Datta, M. Shah, et N. Da Vitoria Lobo. Person-on-person violence detection in video data. Dans *IAPR International Conference on Pattern Recognition*, volume 1, Québec, Août 2002.

[Dav98] D. Davies, P. Palmer, et M. Mirmehdi. Detection and tracking of very small low contrast objects. Dans *British Machine Vision Conference*, pages 599–608, Southampton, UK, Septembre 1998.

[Dav99] C. Davatzikos et J.L. Prince. Convexity analysis of active contour problems. *Image and Vision Computing*, 17(1) :27–36, Janvier 1999.

[Dav00] N.E. Davison, H. Eviatar, et R.L. Somorjai. Snakes simplified. *Pattern Recognition*, 33(10) :1651–1664, Octobre 2000.

[Del95] P. Delagnes, J. Benois, et D. Barba. Active contours approach to object tracking in image sequences with complex background. *Pattern Recognition Letters*, 16(2) :171–178, Février 1995.

[Del00] E. Dellandrea, P. Makris, M. Boiron, et N. Vincent. Active contours for bolus tracking in x-ray images sequences. Dans *IAPR International Workshop on Machine Vision Applications*, pages 303–306, Tokyo, Novembre 2000.

[Den95] J. Denzler et H. Niemann. Evaluating the performance of active contours models for real-time object tracking. Dans *Asian Conference on Computer Vision*, volume 2, pages 341–345, Singapore, 1995.

[Den99] J. Denzler et H. Niemann. Active rays : Polar-transformed active contours for real-time contour tracking. *Journal of Real-Time Imaging*, 5(3) :203–213, Juin 1999.

[Deu97] C. Deutsch. Suivi et localisation d'une cible complexe connue par vision monoculaire. Rapport prédoctoral RT-LVSN-97-04, Laboratoire de Vision et Systèmes Numériques, Université Laval, Juin 1997.

[Die00] C.P. Diehl. *Toward Efficient Collaborative Classification for Distributed Video Surveillance*. Thèse de doctorat, Carnegie Mellon University, Décembre 2000.

[Dje02a] C. Djeraba, éditeur. *IEEE Multimedia, Special Issue on Content-Based Multimedia Indexing and Retrieval*, 9(2), Avril/Juin 2002.

[Dje02b] C. Djeraba, éditeur. *International Journal of Multimedia Tools and Applications, Special Issue on Content-Based Multimedia Indexing and Retrieval*, 14(2), Juin 2002.

[D'O02] T. D'Orazio, N. Ancona, G. Cicirelli, et M. Nitti. A ball dectection algorithm for real soccer image sequences. Dans *IAPR International Conference on Pattern Recognition*, volume 1, Québec, Août 2002.

[dP97] H.J.G. de Poot, F. Moelaert El-Hadidy, et D.D. Velthausz. Multi media manifestation part 1 : Modelling a soccer vds using admire. Rapport Interne IRS/97020, Telematica Instituut, 1997.

[Dre99] M.S. Drew, J. Wei, et Z.N. Li. Illumination-invariant image retrieval and video segmentation. *Pattern Recognition*, 32(8) :1369–1388, Août 1999.

[Eic00] S. Eickeler et G. Rigoll. A novel error measure for the evaluation of video indexing systems. Dans *International Conference on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, Juin 2000.

[Eki02] A. Ekin et A.M. Tekalp. A framework for tracking and analysis of soccer video. Dans *SPIE Conference on Visual Communications and Image Processing*, volume 4671, San Jose, USA, Janvier 2002.

[Elg00] A. Elgammal, D. Harwood, et L. Davis. Non-parametric model for background substraction. Dans *European Conference on Computer Vision*, volume 2, pages 751–767, Dublin, Ireland, Juin/Juillet 2000.

[Ell99] B. Elliger. Analysis of motion compensated filters concerning motion correctness and accuracy. *Signal Processing : Image Communication*, 14(9) :697–720, Juillet 1999.

[EY99] A. El-Yacoubi, M. Gilloux, R. Sabourin, et C. Y. Suen. An hmm-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8) :752–760, Août 1999.

[Fan01] J. Fan, W.G. Aref, M.S. Hacid, et A.K. Elmagarmid. An improved automatic isotropic color edge detection technique. *Pattern Recognition Letters*, 22(13) :1419–1429, Novembre 2001.

[Fen96] J. Feng, K.T. Lo, et H. Mehrpour. Scene change detection algorithm for mpeg video sequence. Dans *IEEE International Conference on Image Processing*, volume 2, pages 821–824, Lausanne, Switzerland, Septembre 1996.

[Fig98] P. Figueroa, N.J. Leite, R.M. Barros, et R. Brenzikofer. Tracking marker for human motion analysis. Dans *EURASIP European Conference on Signal Processing EUSIPCO*, Rhodes, Grèce, 1998.

[For98] A. Ford et A. Roberts. Colour space conversions. Document Web *http ://www.wmin.ac.uk/ITRG/docs/coloureq.html*, Août 1998.

[For99] G.L. Foresti. Object recognition and tracking for remote video surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(7) :1045–1062, Octobre 1999.

[For00] G.L. Foresti, P. Mahonen, et C.S. Regazzoni. *Multimedia Video-Based Surveillance Systems : from User Requirements to Research Solutions*. Kluwer Academic Publishers, Septembre 2000.

[For01] G.L. Foresti et C.S. Regazzoni, éditeurs. *Journal of Real-Time Imaging, Special Issue on Video Processing and Communications in Real-Time Video-Based Surveillance*, 7(5), 2001.

[Fre77] W. Frei et C. Chen. Fast boundary detection : A generalization and a new algorithm. *IEEE Transactions on Computer*, 26(10) :988–998, Octobre 1977.

[Fuk90] K. Fukunaga. *Statistical Pattern Recognition*. Academic Press, 1990.

[Gar96] U. Gargi et R. Kasturi. An evaluation of color histogram-based methods in video indexing. Dans *International Workshop on Image Databases and Multimedia Search*, pages 75–82, Amsterdam, The Netherlands, Août 1996.

[Gar00] U. Gargi, R. Kasturi, et S.H. Strayer. Performance characterization of video-shot-change detection methods. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1) :1–13, Février 2000.

[Gav99] D. Gavrilla. The visual analysis of human movement : Survey. *Computer Vision and Image Understanding*, 73(1) :82–98, Juin 1999.

[Ge02] X. Ge et J. Tian. An automatic active contour model for multiple objects. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Gio97] N. Giordana et W. Pieczynski. Estimation of generalized multisensor hidden markov chains and unsupervised image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5) :465–475, Mai 1997.

[Gon92a] S. Gong. Visual observation as reactive learning. Dans *SPIE International Conference on Adaptative and Learning Systems*, volume 1706, pages 175–186, Orlando, FL, Avril 1992.

[Gon92b] R.C. Gonzalez et R.E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[Gon95a] Y. Gong, C. Hock-Chuan, et L.T. Sin. An automatic video parser for tv soccer games. Dans *Asian Conference on Computer Vision*, volume 2, pages 509–513, Singapour, Décembre 1995.

[Gon95b] Y. Gong, L.T. Sin, C.H. Chuan, H. Zhang, et M. Sakauchi. Automatic parsing of tv soccer programs. Dans *IEEE International Conference on Multimedia Computing and Systems*, pages 167–174, Washington DC, USA, Mai 1995.

[Gor99] G. Gordon, T. Darell, M. Harville, et J. Woodfill. Background estimation and removal based on range and color. Dans *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 459–464, Fort Collins, CO, USA, Juin 1999.

[Gra96] A.E. Grace et D. Pycock. Multiresolution active contour models in textured stereo images. Dans *British Machine Vision Conference*, pages 575–584, Edinburgh, UK, 1996.

[Gre91] J. J. Grefenstette. Lamarckian learning in multi-agent environments. Dans *International Conference on Genetic Algorithms*, pages 303–310, San Mateo, USA, Juillet 1991.

[Gur96] P. Gurdjos, P. Dalle, et S. Castan. Tracking 3d coplanar points in the invariant perspective coordinates plane. Dans *IAPR International Conference on Pattern Recognition*, Vienne, Autriche, Août 1996.

[Hae00] N. Haering, N. da Vitoria Lobo, R. Qian, et I Sezan. A framework for designing event detectors. Dans *Asian Conference on Computer Vision*, Taipei, Taiwan, Janvier 2000.

[Ham95] A. Hampapur, R.C. Jain, et T. Weymouth. Production model based digital video segmentation. *International Journal of Multimedia Tools and Applications*, 1(1) :9–46, Mars 1995.

[Ham01] R. Hamdan. *Détection, Suivi et Reconnaissance des Formes et du Mouvement par Modèles Probabilistes d'Apparence*. Thèse de doctorat, Université Louis Pasteur – Strasbourg 1, Janvier 2001.

[Har00] I. Haritaoglu, D. Harwood, et L. Davis. W4 : Realtime surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8) :809–830, Août 2000.

[Has98] B.G. Haskell et al. Image and video coding — emerging standards and beyond. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7) :814–837, Novembre 1998.

[Haz01] R. Hazra. Delivering immersive soccer content to sports fans. Developer update magazine, Intel, Juin 2001.

[Hee97] J.N.H. Heemskerk, R.G. Heller, W. Jonker, E.J. Sommer, et A. Woudstra. Multi media manifestation part 2 : Implementation of a soccer vds. Rapport Interne R&D-RA-97-1036, KPN Research, 1997.

[Hir01] H.G. Hirsch. Hmm adapatation for applications in telecommunication. *Speech Communication*, 34 :127–139, 2001.

[Hol75] J. H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, MI, 1975.

[Hou62] P. V. C. Hough. Methods and means for recognizing complex patterns. U.S. Patent n°3069654, Décembre 1962.

[Idr97] F. Idris et S. Panchanathan. Review of image and video indexing techniques. *Journal of Visual Communication and Image Representation*, 8(2) :146–166, Juin 1997.

[Ina02] N. Inamoto et H. Saito. Intermediate view generation of soccer scene from multiple videos. Dans *IAPR International Conference on Pattern Recognition*, volume 2, Québec, Août 2002.

[Int94] S.S. Intille et A.F. Bobick. Tracking using a local closed-world assumption : Tracking in the football domain. Rapport Interne 296, M.I.T. Media Lab Perceptual Computing Group, Août 1994.

[Int95] S.S. Intille et A.F. Bobick. Closed-world tracking. Dans *IEEE International Conference on Computer Vision*, pages 672–678, Los Alamitos, Juin 1995.

[Int97] S.S. Intille. Tracking using a local closed-world assumption : Tracking in the football domain. Dans *SPIE Conference on Storage and Retrieval for Image and Video Databases*, pages 216–227, 1997.

[Int99] S. Intille et A. Bobick. A framework for recognizing multi-agent action from visual evidence. Dans *American Conference on Artificial Intelligence (AAAI)*, pages 518–525, Orlando, USA, Juillet 1999.

[Int01] S.S. Intille et A.F. Bobick. Recognizing planned, multiperson action. *Computer Vision and Image Understanding*, 81(3) :414–445, Mars 2001.

[Iva00] Y. Ivanov, A. Bobick, et J. Liu. Fast lighting independent background subtraction. *International Journal of Computer Vision*, 37(2) :199–207, Juin 2000.

[Iwa95] M. Iwase, T. Tanaka, et N. Ohnishi. Image synthesis from a soccer player. *Transactions of IEICE*, PRU95-72, 1995.

[Jai98a] A. Jain et B. Yu. Automatic text location in images and video frames. *Pattern Recognition*, 31(12) :2055–2076, 1998.

[Jai98b] A.K. Jain, Y. Zhong, et M.P. Dubuisson-Joly. Deformable template models : A review. *Signal Processing*, 71(12) :109–129, Décembre 1998.

[Jeu02] C. Jeusse. Détection de lignes dans des images couleur en temps réel. Mémoire de dea, E3i, Université de Tours, Septembre 2002.

[Jia98] H. Jiang, A.S. Helal, A.K. Elmagarmid, et A. Joshi. Scene change detection techniques for video database systems. *Multimedia Systems*, 6(3) :186–195, Mai 1998.

[Jia01] X. Jiang et H. Bunke, éditeurs. *Pattern Recognition Letters, Special Issue on Image / Video Indexing and Retrieval*, 22(5), Avril 2001.

[Jol90] J.M. Jolion. Analyse d'images : le modèle pyramidal. *Traitement du Signal*, 7 :5–17, 1990.

[Jol93] J.M. Jolion et A. Rosenfeld. *A Pyramid Framework for Early Vision*. Kluwer Academic Publishers, Décembre 1993.

[Kal96] H. Kalviainen, P. Hirvonen, et E. Oja. Houghtool – a software package for the use of the hough transform. *Pattern Recognition Letters*, 17(8) :889–897, 1996.

[Kan02a] T. Kaneko et O. Hori. Template update criterion for template matching of image sequences. Dans *IAPR International Conference on Pattern Recognition*, volume 2, Québec, Août 2002.

[Kan02b] H. Kang, D. Kim, et S.Y. Bang. Real-time multiple people tracking using competitive condensation. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Kas88] M. Kass, A . Witkins, et D. Terzopoulos. Snakes : Active contour models. *International Journal of Computer Vision*, 1(4) :321–331, Janvier 1988.

[Kas99] A. A. Kassim, T. Tan, et K. H. Tan. A comparative study of efficient generalised hough transform techniques. *Image and Vision Computing*, 17(10) :737–748, Août 1999.

[Kaw98] T. Kawashima, K. Tateyama, T. Iijima, et Y. Aoki. Indexing of baseball telecast for content-based video retrieval. Dans *IEEE International Conference on Image Processing*, pages 871–875, Chicago, IL, Octobre 1998.

[Ker00] M. Kermit et A.J. Eide. Audio signal identification via pattern capture and template matching. *Pattern Recognition Letters*, 21 :269–275, 2000.

[Kim96] D. Kimber et L. Wilcox. Acoustic segmentation for audio browsers. Dans *Interface Conference*, Sydney, Australia, Juillet 1996.

[Kim98a] T. Kim, Y. Seo, et K.S. Hong. Physics–based 3d position analysis of a soccer ball from monocular image sequences. Dans *IEEE International Conference on Computer Vision*, pages 721–726, Bombay, Inde, Janvier 1998.

[Kim98b] T. Kim, Y. Seo, et K.S. Hong. Soccer game analysis : 3d positioning of a soccer ball from single view. Dans *Frontiers on Computer Vision*, pages 148–153, Tokyo, Février 1998.

[Kim00a] H.W. Kim et H.S. Hong. Soccer video mosaicing using self-calibration and line tracking. Dans *IAPR International Conference on Pattern Recognition*, pages 592–595, Barcelone, Septembre 2000.

[Kim00b] H.W. Kim, Y. Seo, et K.S. Hong. Calculation of 3d trajectories of moving objects from unsynchronized stereo videos. *Electronic Letters*, 36(8) :714–716, Avril 2000.

[Kim01a] H.S. Kim et Y.K. Yang. An automatic classification technique for indexing of soccer highlights using neural networks. *Machine Graphics and Vision*, 10(4) :467–487, 2001.

[Kim01b] H.W. Kim et K.S. Hong. Robust image mosaicing of soccer videos using self-calibration and line tracking. *Pattern Analysis and Applications*, 4(1) :9–19, 2001.

[Kim01c] K.I. Kim, K. Jung, S.H. Park, et H.J. Kim. Support vector machine-based text detection in digital video. *Pattern Recognition*, 34(2) :527–529, 2001.

[Kit01] J. Kittler, K. Messer, W.J. Christmas, B. Levienaise-Obadia, et D. Koubaroulis. Generation of semantic cues for sports video annotation. Dans *IEEE International Conference on Image Processing*, pages 26–29, Thessaloniki, Greece, Octobre 2001.

[Kob97] V. Kobla, D.S. Doermann, K.I. Lin, et C. Falutsos. Compressed domain video indexing techniques using dct and motion vector information in mpeg video. Dans *SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 3022, pages 200–211, San Jose, CA, Février 1997.

[Kob98] V. Kobla et D. Doermann. Indexing and retrieval of mpeg compressed video. *Journal of Electronic Imaging*, 7(2) :294–307, Avril 1998.

[Kob99] V. Kobla, D. DeMenthon, et D. Doermann. Special effect edit detection using videotrails : a comparison with existing techniques. Dans *SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 3656, pages 302–313, San Jose, CA, Janvier 1999.

[Kob00] V. Kobla, D. DeMenthon, et D. Doermann. Identification of sports videos using replay, text, and camera motion features. Dans *SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 3972, pages 332–343, Janvier 2000.

[Kon00] J. Konrad. Motion detection and estimation. Dans A. Bovik, éditeur, *Handbook of Image and Video Processing*, pages 207–225. Academic Press, 2000.

[Kop99] S.K. Kopparapu, P. Mudalige, et P. Corke. A multiresolution based image segmentation. Dans *IEE Conference on Image Processing and its Applications*, pages 567–571, Manchester, UK, Juillet 1999.

[Kop01] I. Koprinska et S. Carrato. Temporal video segmentation : A survey. *Signal Processing : Image Communication*, 16(5) :477–500, Janvier 2001.

[Kou02] D. Koubaroulis, J. Matas, et J. Kittler. Colour-based object recognition for video annotation. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Kur99] M. Kurokawa, T. Echigo, A. Tomita, J. Maeda, H. Miyamori, et S.I. Iisaku. Representation and retrieval of video scene by using object actions and their spatio-temporal relationships. Dans *IEEE International Conference on Image Processing*, volume 2, pages 86–90, Kobe, Japan, Octobre 1999.

[Lac00] L. Lacassagne. *Détection de Mouvement et Suivi d'Objets en Temps Réel*. Thèse de doctorat, Université Pierre et Marie Curie – Paris 6, Juin 2000.

[Lam98] C.L. Lam et S.Y. Yuen. An unbiased active contour algorithm for object tracking. *Pattern Recognition Letters*, 19(5–6) :491–498, Avril 1998.

[Lam02] M. Lamarre et J.J. Clark. Background subtraction using competing models in the block-dct domain. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Laz98] M. Lazarescu, S. Venkatesh, et G. West. Combining nl processing and video data to query american football. Dans *IAPR International Conference on Pattern Recognition*, pages 1238–1240, Brisbane, Australia, Août 1998.

[Laz99] M. Lazarescu, S. Venkatesh, G. West, et T. Caelli. On the automated interpretation and indexing of american football. Dans *IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 802–806, Florence, Italy, Juin 1999.

[Lef00a] S. Lefèvre, C. Fluck, B. Maillard, et N. Vincent. A fast snake-based method to track football players. Dans *IAPR International Workshop on Machine Vision Applications*, pages 501–504, Tokyo, Japon, Novembre 2000.

[Lef00b] S. Lefèvre, J. Holler, et N. Vincent. Real time temporal segmentation of compressed and uncompressed dynamic colour image sequences. Dans *International Workshop on Real Time Image Sequence Analysis*, pages 56–62, Oulu, Finlande, Août 2000.

[Lef01a] S. Lefèvre. Indexation de séquences vidéo en temps-réel et transmission d'extraits pertinents relatifs à une problématique donnée. Dans *Journées de l'Ecole Doctorale SST de l'Université de Tours*, Tours, France, Mai 2001.

[Lef01b] S. Lefèvre, C. Fluck, B. Maillard, et N. Vincent. Un modèle de contour actif pour le suivi rapide d'objets en mouvement. Application au suivi de joueurs de football. Dans *Colloque GRETSI sur le Traitement du Signal et des Images*, Toulouse, France, Septembre 2001.

[Lef01c] S. Lefèvre, J. Holler, et N. Vincent. Indexation de séquences vidéo par détection des changements de plan. Dans *Rencontres Laboratoires - Entreprises en Vision par Ordinateur VISIOMIP'2001*, pages 24–25, Cahors, France, Juin 2001.

[Lef01d] S. Lefèvre, J. Holler, et N. Vincent. Segmentation temporelle de séquences d'images en couleur compressées et non-compressées en temps réel. Dans *Congrès Francophone de Vision par Ordinateur ORASIS'2001*, pages 329–338, Cahors, France, Juin 2001.

[Lef02a] S. Lefèvre. Segmentation multi-résolution d'images couleur appliquée à l'extraction de l'arrière-plan dans des images d'extérieur. Dans *Journées de l'Ecole Doctorale SST de l'Université de Tours*, Tours, France, Mai 2002.

[Lef02b] S. Lefèvre, E. Bouton, T. Brouard, et N. Vincent. Multidimensional hidden markov models for object tracking. Rapport Interne 255, Laboratoire d'Informatique, Université de Tours, Mai 2002.

[Lef02c] S. Lefèvre, C. Dixon, C. Jeusse, et N. Vincent. A local approach for fast line detection. Dans *IEEE International Conference on Digital Signal Processing*, volume 2, pages 1109–1112, Santorin, Grèce, Août 2002.

[Lef02d] S. Lefèvre, J.P. Gérard, A. Piron, et N. Vincent. An extended snake model for real-time multiple object tracking. Dans *International Workshop on Advanced Concepts for Intelligent Vision Systems*, pages 268–275, Gand, Belgique, Septembre 2002.

[Lef02e] S. Lefèvre, J. Holler, et N. Vincent. A review on compressed video parsing. Rapport Interne 257, Laboratoire d'Informatique, Université de Tours, Mai 2002.

[Lef02f] S. Lefèvre, J. Holler, et N. Vincent. A study of real-time segmentation of uncompressed video sequences for content-based search and retrieval. Real Time Imaging, accepté pour publication, Septembre 2002.

[Lef02g] S. Lefèvre, B. Maillard, et N. Vincent. 3 class segmentation for analysis of football audio sequences. Dans *IEEE International Conference on Digital Signal Processing*, volume 2, pages 975–978, Santorin, Grèce, Août 2002.

[Lef02h] S. Lefèvre, B. Maillard, et N. Vincent. A two level classifier process for audio segmentation. Dans *IAPR International Conference on Pattern Recognition*, volume 3, pages 891–894, Québec, Canada, Août 2002.

[Lef02i] S. Lefèvre, L. Mercier, V. Tiberghien, et N. Vincent. Multiresolution color image segmentation applied to background extraction in outdoor images. Dans *IS&T European Conference on Color in Graphics, Image and Vision*, pages 363–367, Poitiers, France, Avril 2002.

[LeG91] D. LeGall. Mpeg : a video compression standard for multimedia applications. *Communications of the ACM*, 34(4) :46–58, 1991.

[Ler96] B. Leroy, I.L. Herlin, et L.D. Cohen. Multi-resolution algorithms for active contour models. Dans *International Conference on Analysis and Optimization of Systems*, pages 58–65, Rocquencourt, France, 1996.

[Lev83] S. E. Levinson, L. R. Rabiner, et M. M. Sondhi. An introduction to the application of theory of probabilistic functions of a markov process to automatic speech recognition. *Bell Systems Technical Journal*, 62(4) :1035–1074, Avril 1983.

[Ley93] F. Leymarie et M.D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6) :617–634, Juin 1993.

[Li00a] H. Li, D. Doerman, et O. Kia. Automatic text detection and tracking in digital video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1) :147–156, 2000.

[Li00b] S.Z. Li. Content-based classification and retrieval of audio using the nearest feature line method. *IEEE Transactions on Speech and Audio Processing*, 8(5) :619–625, Septembre 2000.

[Li01] D. Li, I.K. Sethi, N. Dimitrova, et T. McGee. Classification of general audio data for content-based retrieval. *Pattern Recognition Letters*, 22 :533–544, 2001.

[Lie99] R. Lienhart. Comparison of automatic shot boundary detection algorithms. Dans *SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 3656, pages 290–301, San Jose, CA, Janvier 1999.

[Lie01] R. Lienhart. Reliable transition detection in videos : A survey and practitioner's guide. *International Journal of Image and Graphics*, 1(3) :469–486, Juillet 2001.

[Lip98] A. Lipton, H. Fujiyoshi, et R. Patil. Moving target classification and tracking from real-time video. Dans *IEEE Workshop on Applications of Computer Vision*, pages 8–14, Princeton, USA, Octobre 1998.

[Lit95] T. Little et S. Stevens, éditeurs. *ACM Transactions on Information Systems, Special Issue on Video Information Retrieval*, 13(4), Octobre 1995.

[Liu94] J. Liu et Y.H. Yang. Multiresolution color image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7) :685–700, Juillet 1994.

[Liu98] Z. Liu, Y. Wang, et T. Chen. Audio feature extraction and analysis for scene segmentation and classification. *Journal of VLSI Signal Processing for Signal, Image, and Video Technology*, 20(1/2) :61–79, Octobre 1998.

[Mal89] S. Mallat. Multifrequency channel decompositions of images and wavelet models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37 :2091–2110, 1989.

[Man99] M.K. Mandal, F. Idris, et S. Panchanathan. A critical evaluation of image and video indexing techniques in the compressed domain. *Image and Vision Computing*, 17(7) :513–529, Mai 1999.

[Man00] B.S. Manjunath, T.S. Huang, A.M. Tekalp, et H.J. Zhang, éditeurs. *IEEE Transactions on Image Processing, Special Issue on Image and Video Processing for Digital Libraries*, 9(1), Janvier 2000.

[Man02] R. Mann, A.D. Jepson, et T. El-Maraghi. Trajectory segmentation using dynamic programming. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Mar98] J.S. Marques. A link between image-based and feature-based active contours. *Signal Processing*, 67(3) :271–278, Juin 1998.

[Mar99] F.C.M. Martins, B.R. Nickerson, V. Bostrom, et R. Hazra. Implementation of a real-time foreground / background segmentation system on the intel architecture. Dans *Workshop on Frame Rate Applications, Methods and Experiences with Regularly Available Technology and Equipment*, Kerkyra, Greece, Septembre 1999.

[Mar00] K.V. Mardia et P.E. Jupp. *Directional Statistics*. Wiley & Sons Ltd., 2000.

[Mat98] K. Matsui, M. Iwase, M. Agata, T. Tanaka, et N. Ohnishi. Soccer image sequence computed by a virtual camera. Dans *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 860–865, Santa Barbara, USA, Juin 1998.

[McK00] K. McKoen, R. Navarro-Prieto, B. Duc, F. Ziliani, E. Durucan, et T. Ebrahimi. Evaluation of segmentation methods for surveillance applications. Dans *EURASIP European Conference on Signal Processing EUSIPCO*, pages 1413–1416, Tampere, Finland, Septembre 2000.

[Mee99] W. Mees et M. Acheroy. Trois niveaux de fusion dans un système d'analyse de scène. Dans *Congrès Francophone de Vision Orasis*, pages 225–234, Aussois, France, Avril 1999.

[Meh02] R. Mehrotra et T. Titsworh. More than face value : Airports and multimedia security. *IEEE Multimedia*, 9(2) :11–13, Avril/Juin 2002.

[Men96] J. Meng et S.F. Chang. Cveps : A compressed video editing and parsing system. Dans *ACM International Conference on Multimedia*, pages 43–53, Boston, MA, Novembre 1996.

[Mes02] K. Messer, W. Christmas, et J. Kittler. Automatic sports classification. Dans *IAPR International Conference on Pattern Recognition*, volume 2, Québec, Août 2002.

[Mey92] Y. Meyer. *Wavelets and Operators*. Cambridge University Press, 1992.

[Mis02] T. Misu, M. Naemura, W. Zheng, Y. Izumi, et K. Fukui. Robust tracking of soccer players based on data fusion. Dans *IAPR International Conference on Pattern Recognition*, volume 1, Québec, Août 2002.

[Mit96] A. Mitiche et P. Bouthemy. Computation and analysis of image motion. A synopsis of current problems and methods. *International Journal of Computer Vision*, 19(1) :29–55, 1996.

[Miy00] H. Miyamori et al. Video annotation for content-based retrieval using human behaviour analysis and domain knowledge. Dans *International Conference on Automatic Face and Gesture Recognition*, pages 320–325, Grenoble, France, Mars 2000.

[Miy02a] H. Miyamori. Improving accuracy in behaviour identification for content-based retrieval by using audio and video information. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Miy02b] S. Miyauchi, A. Hirano, N. Babaguchi, et T. Kitahashi. Collaborative multimedia analysis for detecting semantical events from broadcasted sports video. Dans *IAPR International Conference on Pattern Recognition*, volume 2, Québec, Août 2002.

[Nag88] H.H. Nagel. From image sequences towards conceptual descriptions. *Image and Vision Computing*, 6(2) :59–74, 1988.

[Nag91] A. Nagasaka et Y. Tanaka. Automatic video indexing and full-video search for object appearances. Dans *IFIP Working Conference on Visual Database Systems*, pages 113–127, Budapest, Hungary, Octobre 1991.

[Nep01] S. Nepal, U. Srinivasan, et G. Reynolds. Automatic detection of goal segments in basketball videos. Dans *ACM International Conference on Multimedia*, pages 261–269, Ottawa, Canda, Septembre 2001.

[Nga98] K.N. Ngan, S. Panchanathan, T. Sikora, et M.T. Sun, éditeurs. *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Segmentation, Description, and Retrieval of Video Content*, 8(5), Septembre 1998.

[Odo95] J.M. Odobez et P. Bouthemy. Robust multiresolution estimation of parametric motion models. *Journal of Visual Communication and Image Representation*, 6(4) :348–365, Décembre 1995.

[Ohn99] Y. Ohno, J. Miura, et Y. Shirai. Tracking players and a ball in soccer games. Dans *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 147–152, Taiwan, Août 1999.

[Ohn00] Y. Ohno, J. Miura, et Y. Shirai. Tracking players and estimation of the 3d position of a ball in soccer games. Dans *IAPR International Conference on Pattern Recognition*, volume 1, pages 145–148, Barcelone, 2000.

[Oht02] Y. Ohta. Pattern recognition and understanding for visual information media. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Ok02] H.W. Ok, Y. Seo, et K.S. Hong. Multiple soccer players tracking by condensation with occlusion alarm probability. Dans *International Workshop on Statistical Methods for Vision Processing*, Copenhague, Danemark, Juin 2002.

[Pan96] S. Panchanathan et B. Liu, éditeurs. *Journal of Visual Communication and Image Representation, Special Issue on Indexing, Storage, Browsing, and Retrieval of Images and Video*, 7(4), Décembre 1996.

[Pan01] H. Pan, P. Van Beek, et M.I. Sezan. Detection of slow-motion replay segments in sports video for highlights generation. Dans *International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, USA, Mai 2001.

[Par00] N.K. Paragios. *Geodesic Active Regions and Level Set methods : Contributions and Applications in Artificial Vision*. Thèse de doctorat, Université de Nice Sophia-Antipolis, Janvier 2000.

[Par01] M. Pardas et E. Sayrol. Motion estimation based tracking of active contours. *Pattern Recognition Letters*, 22(13) :1447–1456, Novembre 2001.

[Pen96] A. Pentland et R. Picard, éditeurs. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Special Issue on Digital Libraries : Representation and Retrieval*, 18(8), Août 1996.

[Pen98] H.P. Penel et P. Traversian. Le football assisté par ordinateur. *Science et Vie*, 969 :118–122, Juin 1998.

[Per00] J. Pers et S. Kovacic. System for tracking players in sports games by computer vision. *Journal for Electrical Engineering and Computer Science*, 67(5) :281–288, 2000.

[Per01] J. Pers et S. Kovacic. Tracking people in sport : Making use of partially controlled environment. Dans *International Conference on Computer Analysis of Image and Patterns*, pages 374–382, Varsovie, Pologne, Septembre 2001.

[Pet99] N. Peterfreund. Robust tracking of position and velocity with kalman snakes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(6) :564–569, Juin 1999.

[Pet00] M. Petkovic et W. Jonker. A framework for video modelling. Dans *IASTED International Conference on Applied Informatics*, Innsbruck, Autriche, Février 2000.

[Pet01] M. Petkovic, Z. Zivkovic, et W. Jonker. Recognizing strokes in tennis videos using hidden markov models. Dans *IASTED International Conference on Visualization, Imaging and Image Processing*, Marbella, Spain, Septembre 2001.

[Pfe96] S. Pfeiffer, S. Fischer, et W. Effelsberg. Automatic audio content analysis. Dans *ACM International Conference on Multimedia*, pages 21–30, Boston, MA, Novembre 1996.

[Pin98] G.S. Pingali, Y. Jean, et I. Carlbom. Real time tracking for enhanced tennis broadcasts. Dans *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 260–265, Santa Barbara, USA, Juin 1998.

[Pin99] G.S. Pingali, Y. Jean, et I. Carlbom. Lucentvision : A system for enhanced sports viewing. Dans *International Conference on Visual Information Systems*, pages 689–696, Amsterdam, The Netherlands, Juin 1999.

[Pin00a] G.S. Pingali, Y. Jean, et A. Opalach. Ball tracking and virtual replays for innovative tennis broadcasts. Dans *IAPR International Conference on Pattern Recognition*, volume 4, pages 152–156, Barcelone, Espagne, Septembre 2000.

[Pin00b] G.S. Pingali, Y. Jean, A. Opalach, et I. Carlbom. Lucentvision : Converting real world events into multimedia experiences. Dans *IEEE International Conference on Multimedia and Expo*, volume 3, pages 1433–1436, New York, USA, Août 2000.

[Poy99] C. Poynton. Frequently asked questions about color. Document Web *http ://www.inforamp.net/~poynton/Poynton-colour.html*, Décembre 1999.

[Qué99] G. Quénot et P. Mulhem. Two systems for temporal video segmentation. Dans *European Workshop on Content Based Multimedia Indexing*, pages 187–194, Toulouse, France, Octobre 1999.

[Rab89] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2) :257–286, 1989.

[Ram97] M.G. Ramos, S.S. Hemami, et M.A. Tamburro. Psychovisually-based multiresolution image segmentation. Dans *IEEE International Conference on Image Processing*, volume 3, pages 66–69, Santa Barbara, CA, USA, Octobre 1997.

[Ray01] N. Ray, B. Chanda, et J. Das. A fast and flexible multiresolution snake with a definite termination criterion. *Pattern Recognition*, 34(7) :1483–1490, Juillet 2001.

[Ree98] D. Rees, J.I. Agbinya, N. Stone, F. Chen, S. Seneviratne, M. de Burgh, et A. Burch. Click-it : Interactive television highlighter for sports action replay. Dans *IAPR International Conference on Pattern Recognition*, volume 2, pages 1484–1487, Brisbane, Australia, Août 1998.

[Reg99] C. Regazzoni, G. Fabri, et G. Vernazza, éditeurs. *Advanced Video Surveillance Systems*. Kluwer Academic Publishers, 1999.

[Rei96] I. Reid et A. Zisserman. Goal-directed video metrology. Dans *European Conference on Computer Vision*, volume 2, pages 647–658, Cambridge, UK, Avril 1996.

[Rei98] I. Reid et A. North. 3d trajectories from a single viewpoint using shadows. Dans *British Machine Vision Conference*, pages 863–872, Southamption, UK, 1998.

[Rem01] P. Remagnino, G.A. Jones, N. Paragios, et C.S. Regazzoni, éditeurs. *Video-Based Surveillance Systems : Computer Vision and Distributed Processing*. Kluwer Academic Publishers, 2001.

[Ren02] J.M. Rendon Mancha. *Régions Actives Morphologiques : Application à la Vision par Ordinateur*. Thèse de doctorat, Université de Paris 5, Juin 2002.

[Rig00] G. Rigoll, S. Eickeler, et S. Muller. Person tracking in real-world scenarios using statistical methods. Dans *International Conference on Automatic Face and Gesture Recognition*, pages 342–347, Grenoble, France, Mars 2000.

[Ron94] R. Ronfard. Region-based strategies for active contour models. *International Journal of Computer Vision*, 13(2) :229–251, Octobre 1994.

[Ros84] A. Rosenfeld. *Multiresolution Image Processing and Analysis*. Springer-Verlag, 1984.

[Ros98] P.L. Rosin. Thresholding for change detection. Dans *IEEE International Conference on Computer Vision*, pages 274–279, Bombay, Inde, Janvier 1998.

[Rot00] N. Rota et M. Thonnat. Activity recognition from video sequence using declarative models. Dans *European Conference on Artificial Intelligence*, Berlin, Germany, Août 2000.

[RS88] G. Retz-Schmidt. A replai of soccer : Recognizing intentions in the domain of soccer games. Dans *European Conference on Artificial Intelligence*, pages 455–457, Munich, Août 1988.

[Rui99] R. Ruiloba, P. Joly, S. Marchand-Maillet, et G. Quénot. Towards a standard protocol for the evaluation of video-to-shots segmentation algorithms. Dans *European Workshop on Content Based Multimedia Indexing*, pages 41–48, Toulouse, France, Octobre 1999.

[Rui00] Y. Rui, A. Gupta, et A. Acero. Automatically extracting highlights for tv baseball programs. Dans *ACM International Conference on Multimedia*, pages 105–115, Los Angeles, USA, 2000.

[Sah88] P.K. Sahoo, S. Soltani, A.K.C. Wong, et Y.C. Chen. A survey of thresholding techniques. *Computer Vision, Graphics and Image Processing*, 41 :233–260, 1988.

[Sal01] K. Salzberg, G. Blair, R. Hazra, et R. Qian. Intel's immersive sports vision. Rapport interne, Intel, Mars 2001.

[San99] J. Sanderson, M. Teal, et T. Ellis. Characterisation of a complex maritime scene using fourier space analysis to identify small craft. Dans *IEE Conference on Image Processing and its Applications*, volume 465, pages 803–807, Manchester, UK, Juillet 1999.

[Sat99] T. Sato, T. Kanade, E. Hughes, M. Smith, et S. Satoh. Video ocr : Indexing digital news libraries by recognition of superimposed captions. *Multimedia Systems, Special Issue on Video Libraries*, 7(5) :385–395, 1999.

[Sau97] D.D. Saur, Y.P. Tan, S.R. Kulkarni, et P.J. Ramadge. Automated analysis and annotation of basketball video. Dans *SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 3022, pages 176–187, San Jose, USA, 1997.

[Sea96] W.B. Seales, C.J. Yuan, W. Hu, et M.D. Cutts. Content analysis of compressed video. Rapport Interne 265-96, University of Kentucky Computer Science Dept., Août 1996.

[See97] G. Seetharaman et B. Zavidovique. Image processing in a tree of peano coded images. Dans *IEEE International Conference on Computer Architectures for Machine Perception*, pages 229–234, Como, Italy, Octobre 1997.

[Seo97] Y. Seo, S. Choi, H. Kim, et K.S. Hong. Where are the ball and players ? Soccer game analysis with color-based tracking and image mosaick. Dans *IAPR International Conference on Image Analysis and Processing*, pages 196–203, Florence, 1997.

[Set96] J.A. Sethian. *Level Set Methods*. Cambridge University Press, 1996.

[Set99] J.A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[She98] B. Shen et K. Sethi. Block-based manipulations of transformed-compressed images and videos. *Multimedia Systems*, 6(2) :113–124, Mars 1998.

[Shp99] R. Shpilman et V. Brailovsky. Fast and robust techniques for detecting straight line segments using local models. *Pattern Recognition Letters*, 20 :865–877, 1999.

[Sik97] T. Sikora. Mpeg digital video-coding standards. *IEEE Signal Processing Magazine*, 14(5) :82–100, Septembre 1997.

[Smi93] B.C. Smith et L.A. Rowe. Algorithms for manipulating compressed images. *IEEE Computer Graphics and Applications*, 13(5) :34–42, Septembre 1993.

[Smi95] B.C. Smith. A survey of compressed domain processing techniques. Dans *Symposium on Reconnecting Science and Humanities in Digital Libraries*, Lexington, USA, Octobre 1995.

[Sob90] I. Sobel. An isotropic $3 \times 3$ image gradient operator. Dans H. Freeman, éditeur, *Machine vision for three-dimensional scenes*, pages 376–379. Academic Press, 1990.

[Son98] J. Song et B.L. Yeo. Spatially reduced image extraction from mpeg-2 video : Fast algorithms and applications. Dans *SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 3312, pages 93–107, San Jose, CA, Janvier 1998.

[Son00] J. Song et B.L. Yeo. A fast algorithm for dct-domain inverse motion compensation based on shared information in a macroblock. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(5) :767–775, Août 2000.

[Sta99] C. Stauffer et W.E.L. Grimson. Adaptive background mixture models for real-time tracking. Dans *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 246–252, Fort Collins, CO, USA, Juin 1999.

[Ste83] R. J. Stevens, A. F. Lehar, et F. H. Perston. Manipulation and presentation of multi-dimensional image data using the peano scan. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(5) :520–526, Septembre 1983.

[Sud98] G. Sudhir, J.C.M. Lee, et A.K. Jain. Automatic classification of tennis video for high-level content-based retrieval. Dans *IEEE International Workshop on Content-based Access to Image and Video Databases*, pages 81–90, Bombay, Inde, Janvier 1998.

[Sun02] S. Sung et W. Chun. Knowledge-based numeric open caption recognition for live sportscast. Dans *IAPR International Conference on Pattern Recognition*, volume 2, Québec, Août 2002.

[Swa93] D. Swanberg, C.F. Shu, et R. Jain. Knowledge guided parsing and retrieval in video databases. Dans *SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 1908, pages 13–24, San Jose, CA, Février 1993.

[Tak96] T. Taki, J.I. Hasegawa, et T. Fukumura. Development of motion analysis system for quantitative evaluation of teamwork in soccer games. Dans *IEEE International Conference on Image Processing*, volume 3, pages 815–818, Lausanne, 1996.

[Tak99] T. Taki et J.I. Hasegawa. Dominant region : A basic feature for group motion analysis and its application to teamwork evaluation in soccer games. Dans *IS&T / SPIE International Conference on Electronic Imaging : Science and Technology*, pages 48–57, San José, Janvier 1999.

[Tak00] T. Taki et J.I. Hasegawa. Visualization of dominant region in team games and its application to teamwork analysis. Dans *IEEE International Conference on Computer Graphics (CGI)*, pages 227–238, Genève, Suisse, Juin 2000.

[Tan00] Y.P. Tan, D.D. Saur, S.R. Kulkarni, et P.J. Ramadge. Rapid estimation of camera motion from compressed video with application to video annotation. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1) :133–146, Février 2000.

[Tao99] H. Tao, H. Sawhney, et R. Kumar. A sampling algorithm for tracking multiple objects. Dans *International Workshop on Vision Algorithms : Theory and Practice*, volume 1883, pages 53–68, Corfu, Greece, Septembre 1999.

[Ter92] D. Terzopoulos et R. Szeliski. Tracking with kalman snakes. Dans A. Blake et A. Yuille, éditeurs, *Active Vision*, pages 3–20. MIT Press, 1992.

[Ter02] M. Teraguchi, K. Masumitsu, T. Echigo, S.I. Sekiguchi, et M. Etoh. Rapid generation of event-based indexes for personalized video digests. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Tra91] D. Travis. *Effective Color Displays. Theory and Practice*. Academic Press, 1991.

[Tsu01] T. Tsuda, D. Kato, A. Ishikawa, et S. Inoue. Automatic tracking sensor camera system. Dans *IS&T / SPIE International Conference on Electronic Imaging : Science and Technology*, pages 144–153, San Jose, 2001.

[Tza00] G. Tzanetakis et P. Cook. Sound analysis using mpeg compressed audio. Dans *International Conference on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, Juin 2000.

[Tzi94] G. Tziritas et C. Labit. *Motion analysis for image sequence coding*. Advances in Image Communication. Elsevier, 1994.

[Van97] N. Vandenbroucke, L. Macaire, et C. Vieren. Suivi automatique de personnes en mouvement par analyse d'images couleur successives. Application au suivi de joueurs de football. Dans *Colloque GRETSI sur le Traitement du Signal et des Images*, volume 2, pages 917–920, Grenoble, 1997.

[Van98] N. Vandenbroucke, L. Macaire, et J.G. Postaire. Color pixels classification in an hybrid color space. Dans *IEEE International Conference on Image Processing*, volume 1, pages 176–180, Chicago, 1998.

[Van00] N. Vandenbroucke. *Segmentation d'images couleur par classification de pixels dans des espaces d'attributs colorimétriques adaptés. Application à l'analyse d'images de football.* Thèse de doctorat, Université des Sciences et Technologies de Lille 1, Décembre 2000.

[Vie95] C. Vieren, F. Cabestaing, et J.G. Postaire. Catching moving objects with snakes for motion tracking. *Pattern Recognition Letters*, 16(7) :679–685, Juillet 1995.

[Vin02] N. Vincent et J.J. Rousselle. Determination of optimal coefficients in active contour method for contour extraction. Dans *International Colloquium on Numerical Analysis and Computer Science with Applications*, Plovdiv, Bulgarie, Août 2002.

[Vit67] A. J. Viterbi. Error bounds for convolutional codes and asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13 :260–269, 1967.

[VM01] S. Venegas-Martinez. *Analyse et segmentation de séquences d'images en vue d'une reconnaissance de formes efficace*. Thèse de doctorat, Université de Paris 5, 2001.

[Wag74] R. A. Wagner et M. J. Fischer. The string to string correction problem. *Journal of ACM*, 22(2) :168–173, 1974.

[Wal91] G.K. Wallace. The jpeg still picture compression standard. *Communications of the ACM*, 34 :31–44, 1991.

[Wan00] Y. Wang, Z. Liu, et J.C. Huang. Multimedia content analysis using both audio and visual clues. *IEEE Signal Processing Magazine*, 17(6) :12–36, Novembre 2000.

[Wan01] J.Z. Wang, J. Li, R.M. Gray, et G. Wiederhold. Unsupervised multiresolution segmentation for images with low depth of field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1) :85–90, Janvier 2001.

[Wan02] Y. Wang, J. Ostermann, et Y.Q. Zhang. *Video Processing and Communications*. Prentice Hall, 2002.

[Wat01] J. Watkinson. *The MPEG Handbook. MPEG-1, MPEG-2, MPEG-4*. Focal Press, 2001.

[Wil92] D.J. Williams et M. Shah. A fast algorithm for active contours and curvature estimation. *Computer Vision, Graphics and Image Processing : Image Understanding*, 55(1) :14–26, Janvier 1992.

[Wil99] A. D. Wilson et A. F. Bobick. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9) :884–900, Septembre 1999.

[Wol99a] E. Wold, T. Blum, D. Keislar, et J. Wheaton. Classification, search, and retrieval of audio. Dans *CRC Handbook of Multimedia Computing*. CRC Press, 1999.

[Wol99b] W. Wolf, éditeur. *Multimedia Systems, Special Issue on Video Libraries*, 7(5), Septembre 1999.

[Won98] Y.Y. Wong, P.C. Yuen, et C.S. Tong. Contour length termination criterion for snake model. *Pattern Recognition*, 31(5) :597–606, Mai 1998.

[Wou98] A. Woudstra, D.D. Velthausz, H.J.G. de Poot, F. Moelaert El-Hadidy, W. Jonker, M.A.W. Houtsma, R.G. Heller, et J.N.H Heemskerk. Modeling and retrieving audio-visual information. A soccer video retrieval system. Dans *International Workshop on Multimedia Information Systems*, pages 161–173, Istanbul, Turquie, Septembre 1998.

[Wu99] V. Wu, R. Manmatha, et E. Riseman. Textfinder : an automatic system to detect and recognize text in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11) :1224–1229, 1999.

[Xu01a] P. Xu, , S.F. Chang, A. Divakaran, A. Vetro, et H. Sun. Algorithms and system for high-level structure analysis and event detection in soccer video. Rapport Interne 111, Columbia University ADVENT, Juin 2001.

[Xu01b] P. Xu, L. Xie, S.F. Chang, A. Divakaran, A. Vetro, et H. Sun. Algorithms and systems for segmentation and structure analysis in soccer video. Dans *IEEE International Conference on Multimedia and Expo*, pages 928–931, Tokyo, Japon, Août 2001.

[Xu02] G. Xu, Y.F. Ma, H.J. Zhang, et S. Yang. Motion based event recognition using hmm. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Yam02] A. Yamada, Y. Shirai, et J. Miura. Tracking players and a ball in video image sequence and estimating camera parameters for 3d interpretation of soccer games. Dans *IAPR International Conference on Pattern Recognition*, volume 1, Québec, Août 2002.

[Yan94] J. Yang, Y. Xu, et C. S. Chen. Hidden markov model approach to skill learning and its application to telerobotics. *IEEE Transactions on Robotics and Automation*, 10(5) :621–631, 1994.

[Yeo95] B.L. Yeo et B. Liu. Rapid scene analysis on compressed video. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(6) :533–544, Décembre 1995.

[You99] S. Young, M. Forshaw, et M. Hodgetts. Image comparison methods for perimeter surveillance. Dans *IEE Conference on Image Processing and its Applications*, volume 465, pages 799–802, Manchester, UK, Juillet 1999.

[Yow95] D. Yow, B.L. Yeo, M. Yeung, et B. Liu. Analysis and presentation of soccer highlights from digital video. Dans *Asian Conference on Computer Vision*, volume 2, pages 499–503, Singapour, Décembre 1995.

[Yu97] H. Yu, G. Bozdagi, et S. Harrington. Feature-based hierarchical video segmentation. Dans *IEEE International Conference on Image Processing*, volume 2, pages 498–501, Santa Barbara, CA, Octobre 1997.

[Yus00] Y. Yusoff, W. Christmas, et J. Kittler. Video shot cut detection using adaptative thresholding. Dans *British Machine Vision Conference*, pages 362–371, Bristol, UK, Septembre 2000.

[Zha95a] H.J. Zhang, C.Y. Low, et S.W. Smoliar. Video parsing and browsing using compressed data. *International Journal of Multimedia Tools and Applications*, 1(1) :89–111, Mars 1995.

[Zha95b] Y.Q. Zhang. Very low bitrate video coding standards. Dans *SPIE Conference on Visual Communications and Image Processing*, volume 2501, pages 1016–1023, Taipei, Taiwan, Mai 1995.

[Zha99] T. Zhang et C.C.J. Kuo. Heuristic approach for generic audio data segmentation and annotation. Dans *ACM International Conference on Multimedia*, volume 1, pages 67–76, Orlando, USA, Octobre 1999.

[Zha01] L. Zhao. *Dressed Human Modeling, Detection, and Parts Localization*. Thèse de doctorat, Carnegie Mellon University, Juillet 2001.

[Zha02] T. Zhao et R. Nevatia. 3d tracking of human locomotion : A tracking as recognition approach. Dans *IAPR International Conference on Pattern Recognition*, Québec, Août 2002.

[Zhe02] W. Zheng, Y. Shishikui, M. Naemura, Y. Kanatsugu, et S. Itoh. Analysis of space-dependent characteristics of motion-compensated frame differences based on a statistical motion distribution model. *IEEE Transactions on Image Processing*, 11(4) :377–386, Avril 2002.

[Zho00a] Y. Zhong, H.J. Zhang, et A.K. Jain. Automatic caption localization in compressed video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4) :385–392, 2000.

[Zho00b] W. Zhou, A. Vellaikal, et C.C.J. Kuo. Rule-based video classification system for basketball video indexing. Dans *ACM International Conference on Multimedia*, Los Angeles, USA, 2000.

[Zho01a] D. Zhong et S.F. Chang. Structure analysis of sports video using domain models. Dans *IEEE International Conference on Multimedia and Expo*, pages 920–923, Tokyo, Japan, Août 2001.

[Zho01b] D. Zhong et S.F. Chang. Structure parsing and event detection for sports video. Rapport Interne 91, Columbia University ADVENT, Décembre 2001.

[Zil00] F. Ziliani. *Spatio-Temporal Image Segmentation : A New Rule-Based Approach*. Thèse de doctorat, Ecole Polytechnique Fédérale de Lausanne, 2000.

**Détection d'événements dans une séquence vidéo**

---

Le problème abordé ici concerne l'indexation de données multimédia par la recherche d'extraits pertinents. Nos travaux se focalisent sur l'analyse de séquences vidéo afin d'y détecter des événements prédéfinis. La recherche de ces événements étant contextuelle, nous proposons une architecture et des outils génériques et rapides pour la mise en œuvre de systèmes d'indexation spécifiques. Nous insistons notamment sur les problèmes suivants : la segmentation temporelle des données, la séparation du fond et des objets, la structuration du fond, le suivi des objets (rigides ou non, avec ou sans apprentissage) et l'analyse des données audio. Afin de résoudre ces différents problèmes, les outils génériques que nous proposons sont basés sur des analyses semi-locales, des approches multirésolution, des modèles de Markov cachées et la méthode des contours actifs. L'architecture et les outils introduits ici ont été validés au travers de plusieurs applications.

**Mots clefs** : Détection d'événements, Temps réel, Couleur, Changements de plan, Segmentation d'image, Multirésolution, Modélisation et structuration, Détection de lignes, Suivi d'objet, Contours actifs, Chaînes de Markov cachées multidimensionnelles

---

**Event detection in a video sequence**

---

Here we are concerned by multimedia data indexing by use of pertinent sample searching. Our work focuses on video sequence analysis in order to detect some predefined events. This search being contextual, we propose an architecture and some rapid tools, both generic, for building specific indexing systems. We deal in particular with the following problems: data temporal segmentation, background and object separation, background structure extraction, object tracking (rigid or non-rigid objects, with or without learning) and audio data analysis. In order to solve these different problems, the generic tools we are proposing are based on semi-local analyses, multiresolution approaches, hidden Markov models, and the active contour method. The architecture and the tools introduced here have been validated through several applications.

**Keywords**: Event detection, Real time, Color, Shot change, Image segmentation, Multiresolution, Model and structure computation, Line detection, Object tracking, Active contours, Multidimensional hidden Markov models

---

**Discipline :** Informatique

---

**Laboratoire d'Informatique (EA 2101), Université de Tours**
Equipe Reconnaissance des Formes et Analyse d'Images
64 avenue Jean Portalis
37200 Tours
http://www.li.univ-tours.fr