Méthode de segmentation par graphe pour le suivi de régions spatio-temporelles

E. Galmar B. Huet

Département Communications Multimédia
Institut Eurécom
2229 Route des Crêtes
06904 Sophia-Antipolis, France

{galmar,huet}@eurecom.fr

Résumé

Les représentations par graphe sont aujourd'hui encore peu utilisées pour la segmentation vidéo du fait de leur complexité. Elles constituent pourtant une alternative au problème de consistence temporelle des régions que rencontrent les méthodes alternant segmentation spatiale et mise en correspondance temporelle. En analysant les relations d'affinité entre points, il devient en effet possible d'extraire les volumes spatio-temporels et la structure de la vidéo.

Dans cet article, nous étendons une méthode [1] efficace de segmentation d'images par graphe au domaine spatio-temporel du flux vidéo. Nous substituons la segmentation directe du volume vidéo par un procédé itératif et causal. Le nouvel algorithme permet une réduction significative de la charge, tout en conservant la majeure partie des propriétés de la méthode originelle.

Mots clefs

Segmentation spatio-temporelle de vidéos, suivi de régions, graphes d'adjacence.

1 Introduction

Des outils d'extraction du contenu s'avèrent indispensables au développement des technologies multimédia. Ce domaine a suscité de nombreuses recherches, visant en particulier l'extraction d'indices visuels et la caractérisation du contenu des vidéos.

L'approche la plus populaire consiste à utiliser l'information de mouvement présent dans une séquence. On obtient alors les différents objets à partir de la classification de ces mouvements. Cette approche a été particulièrement développée dans le cadre de la norme MPEG-4, pour des applications orientées objet [2]. Il est possible, par exemple, de séparer la description de l'objet (couleur, texture), son modèle de mouvement et le fond sur lequel il se déplace. Différentes applications sont désormais possibles (mode *sprite*, vidéo-conference, ...). Les méthodes récentes sont multi-objets et permettent de considérer plusieurs mouvements. Un exemple représentatif est donné par Xu et al. Dans [3], ils partent d'une estimation multi-échelle du mouvement, suivie d'une méthode de classification robuste, afin d'initialiser les régions. Celles-ci sont ensuite suivies en recherchant le déplacement minimisant la distance de Hausdorff entre les contours du modèle et ceux de l'image.

Cependant, l'initialisation du modèle et son suivi posent encore certaines difficultés, comme la gestion des objets dynamiques (disparition, occlusion partielle, nouveaux objets). L'approche dite spatio-temporelle permet d'éviter ces problèmes. Plutôt que d'effectuer une segmentation image par image, on recherche directement des volumes à l'intérieur de la séquence : la segmentation sur chaque image devient alors implicite grâce aux connections temporelles. Différentes techniques coexistent : morphologie, clustering, et *graph-cuts*.

Les techniques morphologiques reposent principalement sur deux outils, le filtrage morphologique et la ligne de partage des eaux (LPE, *watershed*). Le filtrage morphologique vise à simplifier la segmentation. Il est utilisé à la fois dans le domaine spatial (suppression des détails) et le domaine temporel (consistance du mouvement) [4]. La ligne de partage des eaux est une technique de croissance de régions abondamment utilisée pour la segmentation d'image. Celle-ci consiste en l'immersion progressive de la surface de l'image (son gradient) à partir de minima locaux (sources), tout en empêchant le mélange des eaux de deux sources par la construction d'une digue. A la fin du procédé, les bassins formés donnent les régions de l'image. La méthode a été aussi adaptée à la segmentation spatio-temporelle, en considérant les objects comme un flux continu. Dans [5], la surface topologique est ainsi définie à partir de gradients 3D. Cependant, cette technique nécessite d'établir au préalable des marqueurs à l'intérieur de chaque objet [4, 5].

On peut aussi considérer le volume vidéo comme un ensemble de points associés à plusieurs groupes. Les espaces de représentation utilisés restent simples, bâtis à partir d'attributs comme la couleur, la position et le mouvement. Le modèle est choisi en fonction d'une hypothèse sous jacente. De Menthon et al. [6, 7] considèrent qu'à

court terme, les objets sont des motifs de couleur se déplaçant linéairement, une sorte de tube vidéo défini par sa couleur, sa position et son orientation. Greenspan et al. [8] regroupent les points d'un objet en fonction de leur couleur moyenne et de leur position. La séquence est alors représentée par un mélange de gaussiennes.

Plutôt que de considérer la distance entre un point et son modèle, un autre procédé consiste à aggréger les points (voxels) les plus «proches» entre eux. Cette idée est mise en oeuvre dans les méthodes de segmentation par *graph-cut* [9, 10]. Cette approche descendante établit pour cela une matrice d'adjacence décrivant la similarité (poids) entre chacun des pixels (noeuds). On partionne ensuite le graphe à l'aide d'un certain critère, comme celui des *Ncuts* qui maximise la similarité intra-régions, tout en minimisant la similarité inter-régions. Ce critère nécessite la décomposition en vecteurs propres de la matrice d'affinité, chaque vecteur propre étant de dimension égale au nombre de noeuds du graphe. Plusieurs approximations permettent d'alléger la charge de calcul : échantillonage des noeuds et voisinage spatio-temporel limité sur la séquence [9], ou encore le calcul des vecteurs propres à partir d'un sous ensemble de noeuds [11].

A l'opposé des *graph-cuts* il est aussi possible de reconstruire les régions par une propagation locale, en aggrégeant point par point. Deux avantages importants résident dans la faible complexité et la continuité spatio-temporelle des régions formées. En effet, chaque point est ajouté à une région existante en calculant un nombre limité de similarités, et si l'on ne compare que les points connexes, chaque région finalement constituée peut être représentée par un arbre de points, disjoint des autres régions, retraçant sa formation.

L'article est organisé de la manière suivante. Dans la partie 2, nous décrivons d'abord l'algorithme de partitionnement de graphes original et son application directe à la segmentation d'une image ou d'une vidéo (section 2.1). Nous motivons ensuite une nouvelle approche causale (2D+t), et introduisons des contraintes spatio-temporelles pour le suivi des régions (section 2.2). Dans la partie 3, nous comparons les segmentations obtenues par ces deux approches (section 3.1), ainsi que leur complexité (section 3.2).

2 Notre approche

Nous proposons ici de réduire la complexité et d'améliorer la cohérence temporelle des méthodes de segmentation spatio-temporelles utilisant les graphes d'adjacence en limitant les relations entre noeuds par une grille fixée. L'aggrégation et la reconstruction des volumes sont donc réalisées par propagation entre pixels connexes.

2.1 Segmentation de graphe

Nous récapitulons ici l'algorithme rapide de segmentation de graphe décrit dans [1], appliqué à une image. Il permet une segmentation en $O(n \log n)$ pour un graphe à n arêtes, tout en conservant les propriétés globales et locales grâce à un critère de bonne segmentation.

Soit $G = \{V, E\}$ un graphe non orienté de l'image ; les éléments de V sont les pixels. Chaque arête e est pondérée par un poids $w(e)$ mesurant la similarité entre deux sommets adjacents. Le but de l'algorithme est de fournir une partition $S = \{C_1, C_2, \ldots, C_k\}$ de G, où chaque composante $C_{i=1\cdots k}$ est un arbre recouvrant de poids minimum (ou *minimum spanning tree*) et définit ainsi un ensemble de pixels connexes sur l'image. Chaque composante est donc représentée par sa variation interne $Int(C_i)$, c.a.d. par le poids $w(e)$ le plus grand de C_i. Deux composantes se comparent par leur variation externe $Ext(C_i, C_j)$, correspondant au poids mininum des arêtes reliant C_i et Cj.

La partition voulue est obtenue sur la base de l'algorithme de Kruskal. Au départ $S = \{C_1, C_2, \ldots, C_{|V|}\}$, aucun sommet n'est connecté. On ajoute ensuite les arêtes par ordre croissant de leur poids, en évitant les cycles. Un critère d'arrêt $D(C_i, C_j)$ permet de limiter la croissance des régions :

$$D(C_i, C_j) = \begin{cases} 1 \text{ si } Ext(C_i), C_j) > \min(Int(C_i)+\tau_i, Int(C_j)+\tau_j) \\ 0 \text{ sinon} \end{cases} \quad (1)$$

Le seuil τ sert à pénaliser le regroupement des grandes régions, il diminue avec la taille. Suivant [1], nous définissons $\tau = K/|C|$, où K est une constante. La propriété principale de l'algorithme est de respecter un critère de bonne segmentation à partir de D. S est dite *trop fine* si D est faux entre deux régions adjacentes, càd qu'elles doivent être regroupées. S est dite *trop grossière* s'il est possible de partitionner une des régions C_i en une segmentation qui ne soit pas trop fine. Finalement la segmentation est dite *bonne* si elle n'est ni trop fine ni trop grossière.

L'algorithme présente l'avantage d'être générique et ne dépend pas de la structure de graphe utilisée. Nous l'appliquons directement à la segmentation d'image en extrayant un graphe à partir d'un maillage carré et au flux vidéo en utilisant un maillage cubique. Par la suite, nous nous référerons à ces deux types de segmentation respectivement par *algorithme 2D* et *algorithme 3D*.

2.2 Algorithmes 2D+t

La construction du graphe et la segmentation nécessitent, lors de la phase de croissance, de disposer de l'ensemble de la séquence (labels des régions, arêtes). Lorsque la taille de la séquence devient importante, soit la charge mémoire occupée par l'algorithme devient critique, soit sa complexité s'accroît singulièrement du fait du temps d'accès aux éléments (listes). Passer d'une structure 3D à 2D+t, càd en construisant la segmentation $S_{0,\cdots,t}$ (notée S_t) à partir de $S_{0,\cdots,t-1}$, a alors deux avantages :

– Réduire la charge nécessaire (seules les données à l'instant $t-1$ et t sont nécessaires) en gardant le même ordre de complexité.

– Rendre l'algorithme causal, càd obtenir une segmentation à chaque instant t.

Les principaux problèmes à résoudre sont de garder la consistence temporelle et des propriétés de segmentation proches de l'algorithme 3D. En effet, on perd la propriété de *bonne* segmentation si l'on segmente le graphe par parties, par exemple si l'on effectue une segmentation de l'image I_{t-1} sans tenir compte de I_t.

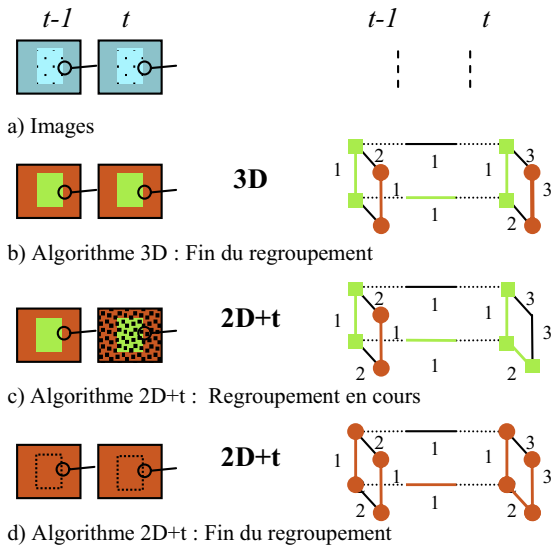


Figure 1 – *Problèmes de la segmentation 2D+t*

Sur l'exemple de la figure 1, on montre la segmentation de régions spatio-temporelles, la plus importante en foncé, la plus petite en clair (fig 1a). On représente à droite une partie de la grille à la jonction des deux régions incluant 4 arêtes spatiales à chaque extrémité, 2 arêtes temporelles au centre, ainsi que le poids respectif de chaque arête. Pour l'algorithme 3D, le regroupement des arêtes s'effectue par ordre croissant, les arêtes reliant les deux régions ne sont traitées qu'à la fin du processus. La taille de chaque région est alors conséquente, càd que leur valeur interne est faible et le critère d'arrêt D est verifié. On obtient alors deux régions distinctes (fig 1b).

Par contre un problème se pose lors de la segmentation 2D+t. Imaginons que l'on construise S_{t-1} à partir des arêtes de I_{t-1}, et ensuite S_t à partir des arêtes de I_t et de celles reliant I_{t-1} à I_t. On réalise alors simultanément la croissance des composantes de S_t et le regroupement temporel entre les images I_{t-1} et I_t. Lorsque l'on teste un regroupement avec une des régions de I_{t-1}, la valeur interne (variabilité) de celle-ci sera plus élevée que lors de la construction de S_{t-1}. En conséquence, une arête peut séparer deux composantes de S_{t-1}, et une arête de même poids les regrouper dans S_t (fig 1c). Autrement dit, la segmentation S_t devient trop *grossière* et les deux régions ont fusionné (fig 1d). La raison sous-jacente à ce problème est le caractère *glouton* (ou *greedy*) de l'algorithme ; une fois deux composantes regroupées, il n'est plus possible de les modifier. D'un autre côté, c'est ce qui donne à la méthode sa faible complexité.

Un moyen de résoudre le problème (fig 1) consiste alors à bâtir la segmentation S_t à partir de S_{t-1}, en la laissant volontairement trop fine. On se base pour cela sur la détection des contours $\mathscr{C}_t$ de I_t. La finesse de S_t est ensuite adaptée en supposant que S_{t-1} est une bonne segmentation. Plutôt que de modifier le critère d'arrêt, nous appliquons une contrainte L locale sur les arêtes (a, b) correspondant aux contours $\mathscr{C}_t$ de I_t.

$$L(a,b) \ : \quad \forall e=(a,b),\ w(e)=\infty \text{ si } a \text{ ou } b \in \mathscr{C}_t \qquad (2)$$

Les composantes sont ainsi contraintes à se propager sur des régions à faible gradient d'intensité, où le regroupement est sans équivoque. La contrainte s'avère utile à la fin de la segmentation lorsque les liens les plus faibles sont rajoutés. L'expérience montre en effet, qu'avec l'algorithme 2D, la segmentation séparée de deux images consécutives peut différer notablement d'une image sur l'autre, à cause de la croissance différente des régions.

Avec cette contrainte, il est alors possible de mettre à profit la continuité temporelle des régions en comparant localement les arêtes connectant I_{t-1} à I_t entre composantes C_i^{t-1}, C_j^t (eq 3). Pour cela, on introduit un critère d'arrêt inter-images $D_{t-1,t}$:

$$D_{t-1,t}(C_i^{t-1}, C_j^t) = \begin{cases} 1 \text{ si } \begin{cases} \|\mu_i - \mu_j\| > \tau_G \\ \text{ou} \\ \max(W_L) > \tau_L \end{cases} \\ 0 \text{ sinon} \end{cases} \qquad (3)$$

$$\tau_G = \max(T_G, p_G \min(\mu_i, \mu_j)) \qquad (4)$$

$$\tau_L = \min(T_L, \mu_i) \qquad (5)$$

Le critère comporte une partie globale et une partie locale, paramétrées par les fonctions de seuil τ_G et τ_L. La partie globale permet de vérifier que la variabilité des composantes est similaire. Idéalement, il faudrait comparer la distribution des pondérations, en prenant par exemple la médiane ou un p-quantile mais leur calcul augmente considérablement la complexité, car elle nécessite tous les éléments du groupe [1]. En pratique, la moyenne des pondérations μ se révéle être un bon indicateur des variations du groupe. μ pouvant varier sur une large échelle, nous mesurons cette similarité en utilisant l'erreur relative entre les moyennes. Pour des régions quasi-homogènes ($\mu \ll 1$), il est préférable de considérer directement la variation absolue. Nous définissons alors τ_G par l'équation 4, où p_G désigne l'erreur relative accordée, et T_G le seuil limite au delà duquel l'erreur relative est appliquée.

La partie locale s'appuie quant à elle sur un voisinage spatio-temporel W_L (fig 2). Il s'agit de comparer la variabilité de ce voisinage à celle de la région C_i formée à partir de I_{t-1}. Nous caractérisons pour cela le voisinage par sa pondération maximale et, similairement à la partie globale, la variabilité du groupe par μ. τ_L est alors définie directement par μ_i, sauf si les pondérations du voisinage dépassent un seuil limite T_L (eq 5), càd que les composantes

ne sont pas regroupées à partir des connections incertaines.

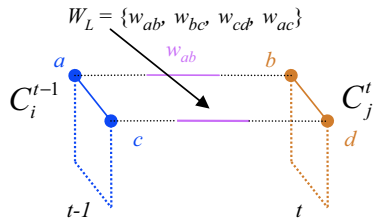


Figure 2 – *Voisinage spatio-temporel de l'algoritme 2D+t*

Le regroupement temporel peut être ainsi effectué de manière locale (une arête pour chaque composante), en gardant les propriétes globales de la segmentation précédente. Une fois effectué, on réalise un post-traitement similaire à celui réalisé dans [1] consistant à regrouper les composantes trop petites et aussi les arêtes liées à $\mathscr{C}_t$ en relâchant le contrainte L. Pour cela, on regroupe ces arêtes en établissant un arbre recouvrant de poids minimum (MST), avec comme critère d'arrêt la taille des composantes.

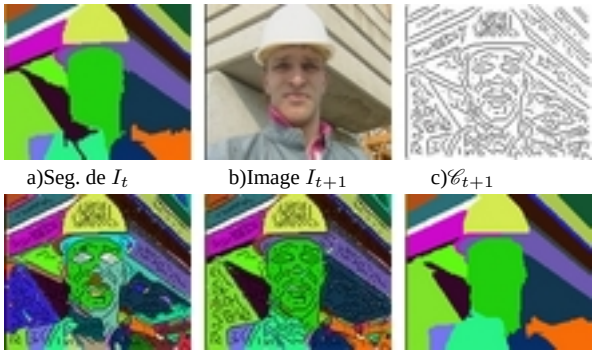


Figure 3 – *Etapes du processus entre S_t et S_{t+1}*

Finalement, la procédure de segmentation comporte les étapes suivantes, illustrées par la figure 3 :

1. Segmenter $I_{t=0}$ avec l'algorithme 2D (eq 1, fig 3a).
2. Construire le graphe de I_{t+1} (fig 3b).
3. Etablir $\mathscr{C}_{t+1}$ à l'aide d'un détecteur de contours (celui de Canny donne une bonne fermeture des contours) (fig 3c).
4. Segmentation partielle de I_{t+1} (eq 1) avec la contrainte L (eq 2, fig 3d).
5. Regroupement temporel (eq 3). Les voisinages W_L sont ordonnés par les arêtes de C_j^{t+1} (fig 3e).
6. Regroupement des composantes trop petites de S_{t+1} en relâchant la contrainte L (fig 3f).
7. $t = t + 1$ et retour à l'étape 2.

On remarque que le regroupement entre I_t et I_{t-1} (fig 3e) permet de fusionner les régions de I_t (fig 3d) en adaptant la finesse de la segmentation à la précédente (fig 3a). Les contours internes (points noirs) sont ensuite supprimés par post-traitement (fig 3a) pour terminer la procédure.

3 Résultats expérimentaux

3.1 Comparaison qualitative

Afin de mettre en évidence les propriétés de la méthode, nous utilisons ici pour w une simple distance L_2 de couleur dans l'espace RGB. Les paramètres de l'algorithme 3D affectent le lissage σ, le nombre de détails K (constante de seuil, eq.1), et la taille minimale des régions m. Pour la figure (4), $\sigma = 0.5$, $K = 300$ et les tailles minimales sont $m_{2D+t} = 200$, $m_{3D} = 2000$ pour les segmentations 2D+t et 3D. Pour la figure (5), $\sigma = 0.8$, $K = 300$, $m_{2D+t} = 200$, $m_{3D} = 6000$. Enfin les fonctions de seuil de l'algorithme 2D+t sont fixées, $\tau_G : (p_G = 0.3, T_G = 10)$ et $\tau_L : (T_L = 5)$. Les tailles des séquences extraites sont respectivement de 10 et 30 images pour les séquences «foreman» et «tennis», d'où le choix de m_{2D+t} et m_{3D} pour la comparaison des segmentations.

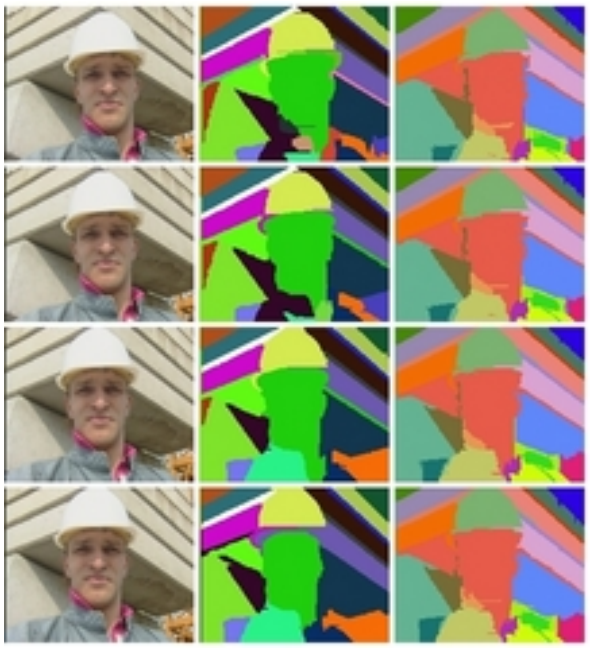
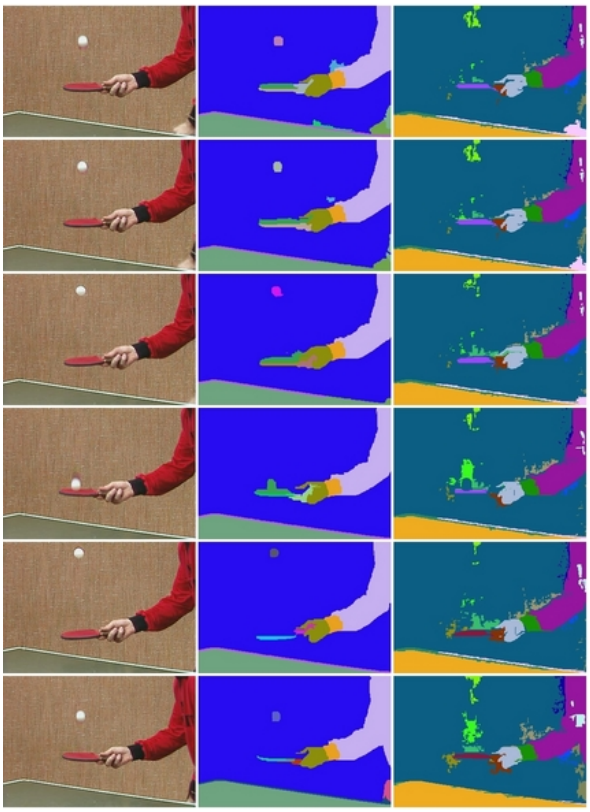


Figure 4 – *Images 0, 1, 5, 9 de la séquence «foreman».*

Les propriétés de l'algorithme 3D sont illustrées sur les figures (4c) et (5c) (en fausses couleurs) :

– Les variations locales sont progressivement éliminées si on augmente K.
– Les régions principales et similaires temporellement se retrouvent dans une même composante (visage, fond) ; elles forment des volumes continus.
– La propagation temporelle des régions permet de retrouver des régions fines à l'échelle spatiale dans la

segmentation finale, par exemple le col rose (fig 4c) mais aussi certaines transitions, par exemple entre le bras et le fond (fig 5c), celles-ci persistant durant la séquence.
– La segmentation est limitée par la grille associée au graphe. S'il n'y a pas de partie commune sur une région (dépacement rapide), celle-ci est perdue (par exemple la balle (fig 5c)).
– Deux objets se rencontrant à un moment donné peuvent se retrouver réunis dans une même composante (par exemple la main et la bordure de la table (fig 5c)).



a)image b) Seg. 2D+t c) Seg. 3D

Figure 5 – *Images 0, 1, 3, 11, 20, 29 de la séquence «tennis», seuil* $\tau_G : (p_G = 0.3, T_G = 10)$.

Par rapport à l'algorithme 3D, la segmentation 2D+t (fig 4b et 5b), s'initialise sur la première image (ligne 1), on ne garde que les régions principales après le pré-traitement. D'où une segmentation plus propre, pour K et σ donnés. Grâce aux contraintes imposées, la segmentation s'adapte aux déplacements des régions (mis à part la balle). Elle reste consistante temporellement et globalement comparable à la section 3D.

La figure (5c), montre aussi la limite de la segmentation et de son caractère *glouton* (ou *greedy*). Nous voyons que lors de leur contact, la balle et la raquette sont réunies ensemble (fig 5b, ligne 4) à cause du post-traitement. La variabilité interne de la région balle-raquette a alors considérablement augmenté (inhomogénéité). Lors de l'image suivante, la contrainte locale devient alors fausse sur certains voisinages (prédominance rouge), ainsi que la contrainte globale vu que la variabilité de l'ensemble rattrape celle du fond. Après le contact, on perd alors la surface de la raquette, rattachée au fond. On pourrait éviter cela avec un seuil global τ_G moins élevé comme le montre la figure 6. Pour l'algorithme 3D, la raquette est considérée comme un objet différent après le contact.

Figure 6 – *Images 11, 20, 29 de la séquence «tennis» en réduisant le seuil* $\tau_G : (p_G = 0.3, T_G = 2.5)$ *après* $t = 11$.

Si les grilles 3D permettent de considérer la continuité temporelle des régions principales, elles ne peuvent pas prendre en compte le déplacement rapide des régions de faible taille. Une solution consiste alors à introduire une phase de prédiction entre S_{t-1} et S_t après la segmentation partielle de I_t (étape 4). En établissant une correspondance entre des points d' une composante de S_{t-1} et ceux d'une autre composante de S_t, on peut établir une connection entre composantes disjointes, et décider de leur regroupement. De bons candidats pour ces marqueurs peuvent être extraits des zones homogènes de I_{t-1}.

3.2 Comparaison des performances

Considérons d'abord la complexité des algorithmes présentés. Dans notre implémentation, les sommets V du graphe sont représentés par une forêt disjointe de n éléments.

Pour l'algorithme 3D, le graphe possède $m = O(n)$ arêtes. Le tri préalable s'effectue donc en $O(n \log n)$. La phase de croissance des régions séquence les opérations du tableau 1.

Opérations	Définition	Nombre d'itérations	Complexité
MakeSet	Créer un singleton	m	$O(m)$
Union	Regrouper 2 ensembles	$m-1$	$O(m)$
FindSet	Trouver le représentant de l'ensemble	$2m$	$O(m\alpha(m))$

Tableau 1 – *Opérations pour le regroupement des composantes.*

Cette 2ème phase a donc une complexité en $O(m\alpha(m))$, où $\alpha(m)$ est une fonction à croissance très lente. La dernière phase est le post-traitement, qui effectue aussi la séquence d'opérations du tableau 1. Finalement, l'algorithme 3D a une complexité en $O(n \log n)$ due au tri des arêtes.

Pour l'algorithme 2D+t, l'ordre de complexité est réduit. En effet, si on considère une séquence de N images, le nombre d'arêtes total m est en $O(N)$. Comme le nombre

d'arêtes m_t utilisées par itération est fixé, le tri s'effectue en $O(1)$, et le cumul sur la séquence en $O(N)$. La complexité totale des tris est donc linéaire avec le nombre d'arêtes, en $O(m)$. Les phases de croissance (4 : segmentation partielle, 5 : regroupement temporel, 6 : post-traitement) décrites dans la section 2.2 ont un séquencement similaire à celle de l'algorithme 3D, avec lors de l'ajout d'une arête, un nombre d'opérations supplémentaires fini. Leur complexité reste donc globalement en $O(m\alpha(m))$ sur toute la séquence. Finalement l'algorithme 2D+t s'effectue en $O(n\alpha(m) + E)$, où E est la complexité de l'étape (3 : détection de contours) de la section 2.2.

Ces algorithmes peuvent donc être considérés comme rapides par rapport à d'autres méthodes par graphes d'ordre $O(n^2)$, comme les *graph-cuts* [9]. En pratique, l'algorithme 3D se révèle d'autant plus rapide que le nombre d'images N est faible. La différence entre les algorithmes proposés dans cet article diminue lorsque N devient élevé, car la phase de tri de l'algorithme 3D s'allonge de même que la charge mémoire augmente. Le tableau (2) illustre les performances des deux algorithmes (temps d'exécution, charge en mémoire). L'implémentation est en C++ et la machine utilisée est un $Pentium\ IV$, 2800 Mhz, 1GB RAM.

		Nombre d'images			
	Alg	10	50	100	150
t(s)	$3D$	3.5	19.4	42.2	64.5
	$2D + t$	4.9	24.9	50.8	73.9
	ratio	0.71	0.78	0.83	0.88
Mem(MB)	$3D$	46	229	460	685
	$2D + t$	30	92	160	230
	ratio	1.5	2.5	2.9	3.0

Tableau 2 – *Performance relative des algorithmes $3D$ et $2D + t$ sur la séquence «tennis» (352x240).*

On voit que les temps d'éxécution de l'algorithme 2D+t et 3D sont comparables, le premier rattrapant progressivement le second. Dans notre implémentation, la charge en mémoire de l'algorithme 3D est due à la fois à la forêt (n sommets) et au graphe (m arêtes), alors que celle de l'algorithme 2D+t est due à la forêt uniquement. Celle-ci peut être rendue indépendante de la taille de la séquence car l'algorithme 2D+t ne nécessite pas la connaissance des sommets à deux instants consécutifs.

4 Conclusion

Les graphes constituent un outil efficace pour la segmentation et le suivi de régions dans les séquences vidéo. Les tailles importantes des volumes vidéo rendent une segmentation directe en volumes délicate. Nous montrons alors qu'il est possible d'obtenir des résultats similaires par une méthode causale, ne prenant en compte qu'une nouvelle image à la fois. Nous imposons pour cela des contraintes spatio-temporelles lors du regroupement des régions afin de conserver la consistence temporelle de la segmentation. A l'avenir, nous pensons qu'une telle méthode peut servir de point de départ pour une segmentation multi-échelles, à partir de graphes d'adjacence sur des régions, afin d'obtenir une description et un modèle des entités présentes dans la séquence.

Références

[1] P.F. Felzenszwalb et D.P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2) :167–181, Septembre 2004.

[2] T. Sikora. Trends and perspectives in image and video coding. *Proceedings of the IEEE : Special Issue on Advances in Video Coding and Delivery*, 93(1) :6–17, Janvier 2005.

[3] H. Xu, A. Younis, et M.R. Kabuka. Automatic moving object extraction for content-based applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(6) :796–812, Juin 2004.

[4] D. Wang et C. Labit. Morphological spatio-temporal simplification for video image segmentation. *Signal Processing : Image Communication*, 11(2) :161–170, Décembre 1997.

[5] M.A. El Saban et B.S. Manjunath. Video region segmentation by spatio-temporal watersheds. Dans *ICIP03*, volume 1, pages 349–352, Barcelone, Espagne, Septembre 2003.

[6] D. DeMenthon et D. Doermann. Video retrieval using spatio-temporal descriptors. Dans *Proceedings of the eleventh ACM international conference on Multimedia*, pages 508–517, Berkeley, CA, Etats-Unis, Novembre 2003.

[7] R. Megret et D. DeMenthon. A Survey of Spatio-Temporal Grouping Techniques. Rapport technique LAMP-TR-094,CS-TR-4403,UMIACS-TR-2002-83,CAR-TR-979, University of Maryland, College Park, 2002.

[8] H. Greenspan, J. Goldberger, et A. Mayer. Probabilistic space-time video modeling via piecewise gmm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3) :384–396, Mars 2004.

[9] J. Shi et J. Malik. Motion segmentation and tracking using normalized cuts. Dans *Procedings of the Sixth International Conference on Computer Vision (ICCV98)*, pages 1154–1160, Bombay, Inde, Janvier 1998.

[10] Y. Weiss. Segmentation using eigenvectors : A unifying view. Dans *Procedings of the Seventh International Conference on Computer Vision (ICCV99)*, volume 2, pages 975–982, Kerkyra, Grèce, Septembre 1999.

[11] C. Fowlkes, S. Belongie, F. Chung, et J. Malik. Spectral grouping using the nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2) :214–224, Février 2004.