Un Algorithme robuste de Segmentation Vidéo pour des Applications Temps Réels

M. El Hassani, D. Rivasseau
Philips Semiconductors
Caen, France

S. Jehan-Besson, M. Revenu
Laboratoire GREYC
Caen, France

D. Tschumperlé, L. Brun
Laboratoire GREYC
Caen , France

M. Duranton
Recherche Philips
Eindhoven,Pays-Bas

Résumé

Dans ce papier, nous proposons un algorithme de segmentation vidéo temps réel conçu pour être stable temporellement. Notre algorithme est basé sur une segmentation spatiale de l'image en régions homogènes qui sont mises à jour au fil de la séquence en utilisant des informations de mouvement. En ce qui concerne la segmentation spatiale, nous proposons une méthode ascendante de segmentation basée sur des fusions successives. Grâce à un ordre de fusion spécifique et à un seuil adaptatif pour le prédicat, la méthode donne de très bons résultats sur des images naturelles (même pour les régions texturées) avec peu de paramètres à régler. De manière à améliorer la consistance temporelle de la segmentation pour une séquence d'images, nous incorporons une information de mouvement via un masque basé sur la détection des changements d'intensités entre deux images. Ce masque est conçu en utilisant à la fois les différences d'intensités entre image successives et la segmentation de l'image précédente. Notre algorithme tourne en temps réel sur un processeur TriMedia pour des séquences de format CIF (Common Intermediate Format).

Mots clefs

Segmentation basée régions, Fusion de régions, Traitement de Séquences Vidéo, Détection du mouvement, Consistance temporelle, Implémentation matérielle, Temps réel.

1 Introduction

La segmentation de vidéos en régions homogènes est cruciale pour de nombreuses applications. Parmi elles, on peut citer l'estimation de mouvement basée régions et la conversion du 2D au 3D où la segmentation est utilisée pour l'estimation de la profondeur. La tendance actuelle se dirige également de plus en plus vers des traitement adaptés au contenu de l'image (filtrage, rehaussement, compression orientée objet) et donc basés sur une segmentation initiale de la vidéo. Toutes ces applications nécessitent l'obtention d'une segmentation précise et stable temporellement. Par ailleurs, les applications multimédia visées nécessitent un algorithme de traitement des séquences en temps réel.

Les méthodes de segmentation spatiale peuvent être divisées en deux catégories, les méthodes basées régions et les méthodes basées contour. Dans la première catégorie [1], les transitions entre pixels sont calculées et les composantes connexes peuvent alors être extraites. Le principal inconvénient de ces approches est que le calcul du gradient est parfois imprécis et très sensible au bruit. De plus il est alors difficile de tenir compte de propriétés statistiques des régions considérées. La seconde catégorie de méthodes (i.e. basées régions) est donc plus souvent utilisée et c'est dans cette catégorie que se situe notre algorithme. Nous nous intéressons plus particulièrement ici aux méthodes ascendantes qui opèrent par fusions successives [2, 3, 4, 5]. Dans ces méthodes, deux points importants sont à considérer : l'ordre de fusion et le critère de similarité.

Lorsque l'on traite de segmentation vidéo, la dimension temporelle doit être ajoutée et la segmentation en régions doit être stable temporellement. De nombreuses approches ont été testées. Quelques auteurs considèrent le temps comme une dimension supplémentaire et la vidéo devient alors un volume 3D [6]. D'autres approches utilisent une information de mouvement comme la détection des changements d'intensités ou les vecteurs mouvement [7, 8]. Nous n'abordons pas ici le suivi d'objet qui consiste à suivre un objet au cours du temps (voir par exemple les travaux [9]).

Dans ce papier, nous proposons une segmentation spatiale basée sur la fusion de régions. Cette fusion est faite dans un ordre spécifique qui est fonction de la différence d'intensité entre les pixels voisins. Par ailleurs, un critère de similarité basé sur les différences de moyennes entre régions est utilisé pour la fusion. La fusion est arrêtée à l'aide d'un seuil adaptatif justifié par une modélisation statistique de l'image. Nous nous sommes basés sur les travaux de [2] en proposant une modélisation statistique de l'image qui conduit à un seuil plus approprié pour la segmentation en temps réel. Notre méthode donne une segmentation très satisfaisante pour des images naturelles (même pour des images texturées) avec peu de paramètres à régler. De manière à améliorer la consistance temporelle, nous proposons d'ajouter l'information de mouvement. Comme l'estimation du mouvement entre deux images est une opération coûteuse, nous avons choisi de combiner un masque de détection des changements d'illumination avec les informations régions contenues dans la segmentation spatiale de l'image précédente. En faisant des comparaisons au niveau pixellique et au niveau région, on obtient un algorithme très

efficace pour un faible coût calcul.
Notre algorithme tourne en temps réel sur des séquences de format CIF sur le processeur TriMedia. De plus les résultats expérimentaux montrent l'applicabilité de notre méthode sur des séquences d'image que ce soit en terme de qualité de la segmentation obtenue ou de stabilité temporelle.
Ce papier est organisé de la manière suivante. L'algorithme de segmentation spatiale est détaillée dans la section 2. L'introduction d'informations temporelles pour améliorer la consistance est détaillée en section 3. Enfin dans la section 4, nous développons l'implantation de notre algorithme. Les résultats expérimentaux sont finalement donnés section 5.

2 Segmentation spatiale

Considérons une image I de largeur W et de hauteur H, $D = \{1....W\} \times \{1....H\}$ est le domaine de l'image. $I(p,n)$ l'intensité du pixel de position $p = (x,y)^T$ dans l'image n.
La segmentation d'une image en régions consiste à trouver une partition pertinente de l'image en m régions $\{S_1, S_2, .., S_m\}$. L'algorithme proposé part du niveau pixellique et procède par fusions successives pour aboutir à la segmentation finale (méthode ascendante). Un ordre spécifique de fusion est utilisé ainsi qu'un seuil adaptatif pour stopper les fusions. Ces deux étapes sont détaillées ci-après et nous donnerons ensuite l'algorithme complet de segmentation.

2.1 Ordre de fusion

L'ordre de fusion est basé sur les poids des transitions comme dans [2, 10]. L'idée est de fusionner d'abord ce qui est similaire avant de fusionner ce qui est différent. Une transition e est un couple de pixels (p,p') en 4-connexité. La similarité entre pixels est mesurée en calculant la distance entre les intensités des deux pixels. Pour des images couleur, la similarité est calculée de la manière suivante :

$$w(p,p') = \sqrt{\sum_{I \in \{Y,U,V\}} (I(p,n) - I(p',n))^2}. \qquad (1)$$

Nous avons choisi l'espace couleur YUV qui est le format couleur utilisé pour le traitement des vidéos ce qui nous évite ainsi une conversion. De plus, nous avons trouvé que l'espace couleur YUV procure une partition de l'image qui est, subjectivement, de qualité meilleure que celle obtenue avec l'espace RGB. L'espace $L^*a^*b^*$ donne des résultats légèrement meilleurs que l'espace YUV, mais au prix d'une implémentation coûteuse [11].
Les transitions sont ensuite triées dans l'ordre croissant de leurs poids w et les couples de pixels correspondant sont traités dans cet ordre pour la fusion. En ce qui concerne l'implémentation, l'image est seulement parcourue deux fois pour ce tri. Le premier parcours permet de calculer le nombre de transitions de même poids, nombre qui est stocké dans une table. Cette table est ensuite utilisée pour allouer la mémoire pour ces transitions. Le second permet de stocker chaque transition dans la partie de la mémoire correspondante.

2.2 Critère de fusion

Etant données deux régions S_1 et S_2, nous voulons savoir si ces deux régions doivent être fusionnées. En conséquence, un critère de similarité entre régions doit être choisi et évalué. En comparant la valeur de ce critère à un seuil, les régions seront fusionnées ou non. Le choix du seuil est souvent difficile. Dans ce papier, nous utilisons un seuil adaptatif qui dépend de la taille de la région considérée. L'utilisation d'un tel seuil est justifiée par le biais d'inégalités statistiques comme dans [2]. Nous proposons ici une interprétation statistique plus simple de l'image qui nous conduit à un critère plus adapté pour une implémentation temps réel. Nous présentons d'abord ici le prédicat de fusion utilisé, puis sa démonstration.

Prédicat de fusion. La moyenne des intensités de la région S_i est calculé de la manière suivante :

$$\overline{S_i} = \frac{1}{|S_i|} \sum_{k=1}^{k=|S_i|} I_i(p_k)$$

où $I_i(p_k)$ est l'intensité du $k^{ème}$ pixel de la région S_i.
Le prédicat de fusion est alors :

$$P(S_1,S_2) = \begin{cases} vrai & \text{si} \quad (\overline{S_1} - \overline{S_2})^2 \leq Qg^2(\frac{1}{|S_1|} + \frac{1}{|S_2|}) \\ faux & \text{sinon} \end{cases} \qquad (2)$$

avec g le niveau maximum de I ($g = 255$ pour les images de composantes en précision 8 bits). LA notation $|S_i|$ représente la taille de la région S_i. Le paramètre Q permet de régler la finesse de la segmentation. Dans les expériences, nous choisissons $Q = 2$ qui donne de bons résultats pour le format vidéo CIF (taille des images 352×288).

Justification statistique du prédicat. Classiquement, l'image I est considérée comme l'observation d'une image parfaite I^* et les pixels sont alors des observations d'un vecteur de variables aléatoires (v.a) noté $\mathbf{X} = (X_1, .., X_n)^T$. Le prédicat de fusion est basé sur l'inégalité de McDiarmid [12] donnée ci-après :

Theorem 2.1 *(L'inégalité de McDiarmid) Soit* $\mathbf{X} = (X_1, X_2, ..., X_n)$ *une famille de* n *v.a. indépendantes avec* X_k *prenant ses valeurs dans un ensemble* A_k *pour chaque* k*. Supposons que la valeur réelle de la fonction* f *definie sur* $\prod_k A_k$ *satisfasse* $|f(x) - f(x')| \leq c_k$ *si les vecteurs* x *et* x' *diffèrent seulement sur leurs* $k^{ème}$ *coordonnée. Soit* $E(f(\mathbf{X}))$ *l'espérance de* $f(\mathbf{X})$ *alors quelque soit* $\tau \geq 0$*,*

$$Pr(|f(\mathbf{X}) - E(f(\mathbf{X}))| > \tau) \leq 2\exp\left(-2\tau^2 / \sum_k c_k^2\right) \qquad (3)$$

Nous nous intéressons ici à deux régions adjacentes S_1 et S_2. Dans ce cas, nous considérons le vecteur de variables aléatoires suivant :

$$(X_1, ..., X_n) \quad = \quad \left(I_1^*(p_1), ..., I_1^*(p_{|S_1|}), I_2^*(p_1), ..., I_2^*(p_{|S_2|})\right)$$

Avec $I_i(p_j)$ l'intensité du $j^{ème}$ pixel de S_i correspondant à l'observation de la variable aléatoire $I_i^*(p_j)$. Dans ce cas, la taille du vecteur de variables aléatoires est $n = |S_1| + |S_2|$. De manière à appliquer le théorème 2.1, nous choisissons comme fonction $f(\mathbf{x}) = \left(\overline{S_1} - \overline{S_2}\right)$. Pour cette fonction, nous trouvons que $\sum_k c_k^2 = g^2 \left(\frac{1}{|S_1|} + \frac{1}{|S_2|}\right)$.
Par inversion du théorème, nous avons avec une probabilité d'au moins $1 - \delta$ ($0 < \delta \leq 1$) :

$$|(\overline{S_1} - \overline{S_2}) - E\left(\overline{S_1} - \overline{S_2}\right)| \leq g\sqrt{Q\left(\frac{1}{|S_1|} + \frac{1}{|S_2|}\right)}$$

avec $Q = \frac{1}{2} * ln(\frac{2}{\delta})$, et $E(\overline{S})$ l'espérance de $\overline{S}$. Si S_1 et S_2 appartiennent à la même région dans I^*, l'espérance $E\left(\overline{S_1} - \overline{S_2}\right)$ sera nulle et le prédicat suit.

2.3 Algorithme de segmentation

L'algorithme 1 donne le traitement complet pour la segmentation spatiale. L'algorithme de fusion utilise la structure de données "UNION-FIND"[13]. La fonction $UNION$ fusionne deux régions en une seule région et la fonction $FIND$ permet d'identifier la région d'appartenance d'un pixel.

Algorithm 1 La segmentation spatiale

Calcul des poids des transitions et de l'histogramme correspondant
Tri par ordre croissant des transitions en fonction de la valeur de leur poids w.
for $i := 1$ to $2|I|$ **do**
  Lecture de la $i^{ème}$ transition : (p_1, p_2) ;
  $S_1 = FIND(p_1)$;
  $S_2 = FIND(p_2)$;
  if $P(S_1, S_2) = True$ **then**
    $UNION(S_1, S_2)$
  end if
end for

3 Amélioration de la consistance temporelle

La qualité d'une segmentation vidéo dépend non seulement de la bonne séparation entre les régions significatives dans l'image, mais aussi de la consistance temporelle de cette séparation. En effet, si dans deux images successives, une même région est segmentée différemment, ce qui peut être dû au bruit, ou à des situations particulières comme l'occlusion, la deocclusion, alors les résultats de la segmentation seront difficilement exploitables dans des applications comme l'amélioration de l'image ou la conversion 2D/3D. Quelques auteurs [8] utilisent les vecteurs de mouvements pour améliorer la consistance temporelle de la segmentation vidéo. Or l'estimation de mouvement est très coûteuse en temps de calcul et n'est pas toujours fiable. Dans ce papier nous utilisons un masque de changement d'intensités (CDM) combiné avec la segmentation spatiale pour améliorer la cohérence temporelle de la segmentation.
Le CDM est calculé en utilisant la différence d'intensités entre les images et la segmentation de l'image précédente. Premièrement, on détecte les variations d'intensité des pixels en utilisant la différence entre deux images consécutives. Ensuite, nous exploitons la segmentation de l'image précédente afin de classifier les pixels au niveau région. Si on considère l'image courante $I(:, n)$ et l'image précédente $I(:, n-1)$, La différence entre les images FD est donnée par $FD(p) = |I(p, n) - I(p, n-1)|$. Cette différence est classiquement seuillée pour distinguer les variations d'intensité des pixels dues aux objets en mouvement de celles dues au bruit.
Soit L Le résultat de seuillage de l'image FD. Un pixel p, avec $L(p) = 1$ est un pixel classifiée comme changeant, c'est à dire que pour ce pixel, la variation d'intensité entre deux images est supérieure à un seuil. Ensuite on utilise la segmentation précédente pour convertir le CDM du niveau pixel au niveau région, ce qui est plus fiable. Pour chaque région dans la segmentation précédente S_i, on calcule $\tau(S_i)$, qui représente le ratio des pixels changeants dans la région S_i. Nous avons $\tau(S_i) = \frac{N_{i,changeant}}{|S_i|}$, ou $N_{i,changeant}$ est le nombre de pixels changeant dans la région. Les pixels sont classifiés par la suite en trois catégories :

$$CDM(p) = \begin{cases} 0 & si\ (\tau(S_i) \leq tr_2). \\ 1 & si\ (\tau(S_i) > tr_2)\ et\ (L(p) = 1). \\ 2 & si\ (\tau(S_i) > tr_2)\ et\ (L(p) = 0). \end{cases} \quad (4)$$

où tr_2 est une constante positive. Plus cette constante est importante, plus la segmentation sera stable mais on risque alors de ne pas segmenter correctement les zones en mouvement. La valeur du seuil tr_2 est donc choisie faible afin de ne pas manquer une région en mouvement. Dans nos expériences, nous avons pris $tr_2 = 0.01$ (i.e. on considère qu'une région a bougé, si elle contient au moins 1% de pixels changeant, sinon elle est considérée comme statique).
Les pixels appartenant à une région statique sont étiquetés en utilisant $CDM(p) = 0$. Les deux autres étiquettes, $1, 2$, concernent les pixels dans les régions qui ont bougé. Selon la valeur de la différence entre images, appelée ici FD, Le pixel est qualifié comme étant changeant $CDM(p) = 1$ ou non changeant $CDM(p) = 2$. Une telle classification est alors utilisée pour segmenter l'image courante. Premièrement, Les régions statiques sont gardées telles qu'elles étaient segmentés dans l'image précédente. Deuxièmement, on applique un algorithme d'extraction de composantes connexes (CCL)[14] afin d'extraire les com-

posantes connexes des pixels ayant $CDM(p) = 2$. Cette deuxième étape construit des germes à partir de la segmentation précédente. Ces germes lient la segmentation précédente à la segmentation courante tout en améliorant la consistance temporelle. Troisièmement on applique la segmentation spatiale algorithme.1 seulement sur les transitions (p, p') qui connectent les pixels changeant (i.e. $CDM(p) = 1$, $CDM(p') = 1$), et ceux connectant les pixels changeant avec les germes construits dans la deuxième étape (i.e. ($CDM(p) = 1$ et $CDM(p') = 2$) ou ($CDM(p) = 2$ et $CDM(p') = 1$)).
Nous montrons un exemple de segmentation avec et sans consistance temporelle dans Fig.1(d) et 1(c). Dans Fig.1(b), nous représentons les différentes étiquettes (i.e. valeurs du CDM) avec des intensités différentes (noir pour $CDM(p) = 0$, rouge pour $CDM(p) = 2$ et blanc pour $CDM(p) = 1$). Dans la section 5, nous proposons le calcul

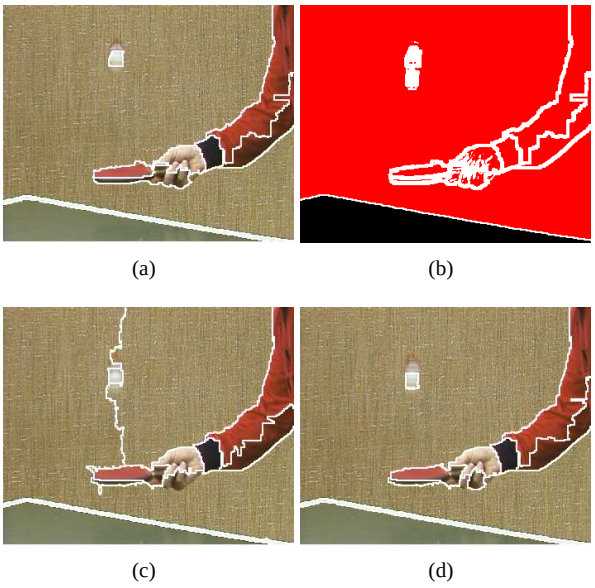


FIG. 1 – *(a) Segmentation de l'image $n-1$. (b) CDM calculé en utilisant la différence d'illumination entre l'image $n-1$ et l'image n et la segmentation de l'image n. (c) Segmentation de l'image n sans utiliser le CDM. (d) Segmentation de l'image n avec l'utilisation du CDM.*

d'une mesure objective de la consistance temporelle. Les résultats montrent une vraie amélioration de la consistance temporelle pour différentes séquences. En plus de cela, La manière dont nous exploitons le CDM réduit aussi les calculs de l'algorithme. Cette réduction des calculs est due au fait que les transitions dans les régions statiques ne sont pas retraitées, et que les transitions connectant les pixels non changeant sont traitées seulement par un algorithme CCL

4 Implémentation de l'algorithme

La figure 2 montre l'algorithme complet de la segmentation vidéo. A part la boucle de fusion de la segmentation spatiale, les autres fonctions de l'algorithme accèdent aux données dans un ordre connu (par exemple de gauche à droite, de haut en bas). Cette régularité d'accès aux données, limite le nombre de défauts de cache puisque La mémoire cache exploite la localité spatiale et temporelle des données. Contrairement à cela, dans la boucle de fusion, la structure de données UNION-FIND est non prédictible et cause donc un nombre de défauts cache important ce qui ralentit l'exécution de l'algorithme. Afin de limiter le nombre de défauts de cache, nous avons investigué deux optimisations qui sont indépendantes du type de la mémoire cache.

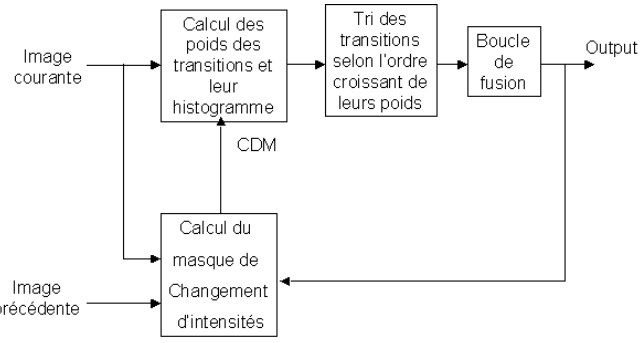


FIG. 2 – *L'algorithme complet de la segmentation vidéo*

La première optimisation concerne la façon dont on étiquette les régions pendant le processus de segmentation. Il y a deux méthodes pour gérer cela. Dans la première, on génère des nouveaux labels qui représentent les régions et donc optimiser l'utilisation des labels. Mais cette première méthode doit utiliser un autre tableau ou sera stocké la correspondance entre les labels générés et les pixels de notre image. Dans cette première méthode nous accédons à deux structures de données qui ont des organisations différentes. Dans la deuxième méthode, chaque pixel dans l'image peut être le représentant d'une région (on peut choisir celui dont l'adresse est la minimale). Nous avons vérifié que la deuxième méthode cause moins de défauts de cache. Donc les données sont stockées dans un tableau A de taille $|I|$, chaque élément $A(p)$ stocke la moyenne des couleurs $(\overline{Y}, \overline{U}, \overline{V})$ et le cardinal $|S|$. La deuxième optimisation Concerne la structure de données UNION-FIND. Elle est encodée séparément du tableau A cité précédemment grâce à un tableau F où $F(p)$ contient le père de p. La solution alternative consisterait à stocker les deux structures dans une seule, mais cela impliquerait le stockage de données inutiles dans le cache lors des opérations FIND.

Nous avons expérimenté ces optimisations sur le processeur TriMedia[15]. La mémoire cache de ce processeur est de taille 128 $KOctet$, 4-associative, avec des blocs de taille 128 $Octet$. L'algorithme de remplacement utilisé est LRU. Nous avons une réduction moyenne de 3 $Mcycles$ du temps d'exécution de l'algorithme. Ces optimisations sont utiles pour plusieurs segmentations qui utilisent la

structure de données UNION-FIND.
Il y a une quantité importante de parallélisme de données (DLP) dans notre algorithme (calcul des valeurs de transitions, calcul du masque de changement entre images). Ceci permet d'accélérer l'algorithme en traitant des données en parallèle si l'architecture cible dispose des ressources suffisantes. Le coeur du TriMedia est une architecture VLIW disposant de plusieurs unités fonctionnelles. Ces unités fonctionnelles peuvent traiter quatre octets en parallèle (mode SIMD). Le parallélisme d'instructions (ILP) est extrait par le compilateur, tandis que le parallélisme de données peut être exploité en utilisant des intrinsèques et le déroulage de boucle. Nous nous sommes servis de ces optimisations pour exploiter le (DLP) qu'offre notre algorithme.

5 Les résultats expérimentaux

Dans cette section, nous présentons des résultats expérimentaux de notre algorithme porté sur le processeur TriMedia et évalué pour des séquences CIF (images de taille 352*288). Le tableau 3 montre les résultats suivants :

1. Mesure de la consistance temporelle :
   Nous avons utilisé ici une mesure classique. Considérons la segmentation de l'image précédente $SEG(n-1)$ et la segmentation de l'image courante $SEG(n)$, nous trouvons la correspondance entre les régions dans $SEG(n-1)$ et celles dans $SEG(n)$. Deux régions ($S_{i,n-1} \in SEG(n-1)$ et $S_{j,n} \in SEG(n)$ correspondent si elles ont un maximum de recouvrement $Overlap(i,j) = |S_{i,n-1} \cap S_{j,n}|$. Nous additionnons le nombre de pixels des zones de recouvrement des régions qui correspondent dans toute l'image. La consistance temporelle est alors le pourcentage de ce nombre par rapport à la taille de l'image.
2. Temps d'execution :
   Nous donnons le nombre de Mega cycles que prend l'exécution de l'algorithme sur TriMedia. Comme nous l'avons déjà noté, l'utilisation du CDM réduit les calculs de l'algorithme. Le cas critique de l'algorithme (temps de calcul le plus long pour une image) correspond à l'exécution de la segmentation spatiale sans CDM. Ce cas critique correspond par exemple au traitement de la première image de la séquence où l'information CDM n'est pas disponible, ou au traitement d'une image après un changement de scène où l'information donnée par le CDM ne réduit pas les calculs. Pour cette raison, nous donnons ici les temps d'exécutions correspondant à ce cas critique. Si on intègre l'algorithme de consistante temporelle, le temps de calcul sera réduit.

Lorsqu'on utilise le CDM, la mesure de consistance temporelle est élevée. Visuellement cela se traduit par une segmentation en régions plus stable d'une image à l'autre. La

Séquence	Akiyo	Paris	Tennis Table	Mobile
Consistance temporelle (Sans CDM)	0.88	0.89	0.73	0.84
Consistance temporelle (Avec CDM)	0.98	0.97	0.92	0.92
Nombre de Mcycles par image	15.68	16.59	15.84	16.20

FIG. 3 – *Les mesures expérimentales de notre algorithme*

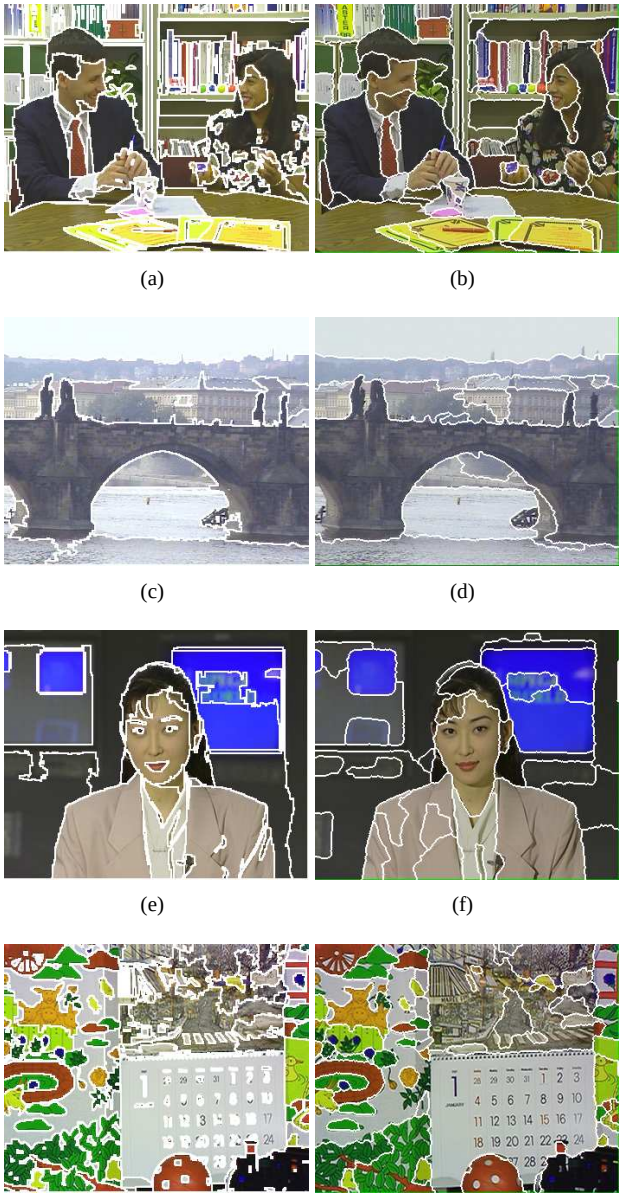
segmentation reste cependant précise comme le montre la figure 4. A titre de comparaison, nous montrons aussi les résultats de segmentation obtenus avec l'algorithme JSEG [7] qui est maintenant une référence dans le domaine de la segmentation. L'algorithme JSEG procède d'abord à une quantification suivie d'une mesure de similarité à plusieurs échelles, ce qui augmente considéralement sa complexité et le nombre de paramètres à régler. Son approche multiéchelle explique les bons résultats obtenus dans les zones texturées (figure.4(b)). Par contre, notre algorithme donne de meilleurs résultats dans les zones homogènes (figure.4(e)) et il extrait également mieux les petites régions. Ce dernier point est important particulièrement lorsque l'application visée est l'amélioration d'image.
Au niveau de l'éxecution de notre algorithme, le processeur TriMedia utilisé dans notre expérimentation peut fonctionner avec une fréquence égale à $450\ MHz$. Une telle fréquence nous permet de traiter plus de 25 images par seconde, ce qui est suffisant pour une exécution temps réel.

6 Conclusion

La conception d'algorithmes pour le traitement des vidéos nécessite une exécution de ceux-ci en temps réel. Pour cette raison, nous avons choisi d'utiliser des méthodes ayant un temps de calcul raisonnable. Nous avons proposé une méthode de segmentation vidéo consistante temporellement. Notre méthode donne des résultats précis, et d'une grande consistance temporelle. Elle est basée sur une segmentation spatiale statistique et un masque de détection des changements d'illumination. Elle fonctionne en temps réel sur un seul processeur. Des résultats expérimentaux démontrent l'applicabilité de cette méthode. Nos recherches futures concernent la mise en oeuvre de l'algorithme en temps réel sur des séquences HD (Haute Définition), et l'exploitation des résultats de segmentation pour l'amélioration d'image.

Remerciements

Les auteurs remercient P. Meuwissen, O.P. Gangwal et Z. Chamski pour leurs suggestions constructives.



(a) (b) (c) (d) (e) (f) (g) (h)

FIG. 4 – *Résultats de segmentation pour les séquences Paris, Bridge, Akiyo et Mobile. Figures a, c, e : Résultats avec notre segmentation. Figures b, d, f : Résultats avec l'algorithme JSEG*

Références

[1] G. Iannizzotto et L. Vita. Fast and accurate edge-based segmentation with no contour smoothing in 2-d real images. *IEEE Transactions on Image Processing*, 9, Issue 7 :1232 – 1237, 2000.

[2] R. Nock et F. Nielsen. Statistical region merging. *IEEE Transactions on PAMI*, 24, 2004.

[3] L. Vincent et P. Soille. Watersheds in digital spaces : an efficient algorithm based on immersion simulations. *IEEE Transactions on PAMI*, 13, Issue 6 :583–598, 1991.

[4] Jianibo Shi et J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on PAMI*, 22, Issue 8 :888 – 905, 2000.

[5] E. Sharon, A. Brandt, et R. Basri. Fast multiscale image segmentation. *IEEE Conference on CVPR*, 1 :70 – 77, 2000.

[6] H.-Y. Wang et K.-K. Ma. Automatic video object segmentation via 3d structure tensor. *ICIP*, 1, 14-17 :153–156, 2003.

[7] Y. Deng et B.S. Manjunath. Unsupervised segmentation of colour-texture regions in images and video. *IEEE Transactions on PAMI*, 23, Issue 8 :800 – 810, 2001.

[8] D. Wang. Unsupervised video segmentation based on watersheds and temporal tracking. *IEEE Transactions on CSVT*, 8, ISSUE 5 :539 – 546, 1998.

[9] F. Moscheni, S. Bhattacharjee, et M. Kunt. Spatio-temporal segmentation based on region merging. *IEEE Transactions on PAMI*, 20, Issue 9, :897 – 915, 1998.

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

[11] C.Connolly et T.Fliess. A study of efficiency and accuracy in the transformation from rgb to cielab color space. *IEEE Transactions on Image Processing*, 6(7) :1046 – 1048, 1997.

[12] M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, et B. Reed. Probabilistic methods for algorithmic discrete math. *Springer Verlag*, 20, Issue 9, :1 – 54, 1998.

[13] C. Fiorio et J. Gustedt. Two linear time union-find strategies for image processing. *Theoretical Computer Science*, 154 :165–181, 1996.

[14] K. Wu, E. Otoo, et A. Shoshani. Optimizing connected component labeling algorithms. *Proceedings of SPIE*, 5747 :1965–1976, 2005.

[15] pnx1500 databook. *available at http ://www.tcshelp.com/public_files.html*.