Algorithme rapide d'estimation de mouvement

Fast Motion Estimation Algorithm

par Christophe ALEXANDRE, Han VU THIEN,

Laboratoire Electronique et Communications
CNAM, 292 rue Saint Martin 75141 PARIS cedex 03

résumé et mots clés

Nous proposons dans cet article un algorithme d'estimation de mouvement ayant le même niveau de robustesse que la méthode de recherche intégrale et donnant des images compensées de même qualité, mais beaucoup plus rapide. De plus, il comporte un paramètre permettant de faire varier la vitesse de la recherche sans modifier les dimensions de la fenêtre de recherche.

Estimation de mouvement, Recherche intégrale, Filtrage, Décimation.

abstract and key words

This paper describes a new motion estimation algorithm. It is designed to have the same level of robustness and image quality as the full search algorithm, but to be much faster than this latter. Moreover, a speed parameter is included in our method which can control search rapidity but does not affect the size of the search window.

Motion estimation, Fullsearch, Filtering, Decimation.

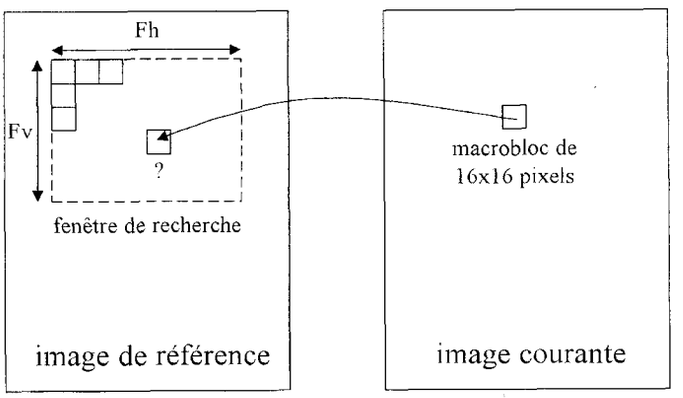
1. introduction

L'estimation de mouvement par appariement de bloc (*block–matching*) consiste, connaissant un macrobloc dans une image, à trouver le macrobloc qui lui ressemble le plus dans une image de référence. La position des deux macroblocs étant connue, on en déduit un vecteur de déplacement. Les critères de ressemblance entre deux macroblocs sont généralement les suivants :

• l'erreur quadratique : c'est la somme des carrés de la différence de luminance entre chaque pixel.

• la différence en valeur absolue : c'est la somme des valeurs absolues de la différence de luminance entre chaque pixel.

La recherche du macrobloc de référence (16 × 16 pixels) se fait à l'intérieur d'une fenêtre de recherche dont les dimensions, pour un codeur MPEG–2 [1], sont fonctions des paramètres $f_$code horizontal et $f_$code vertical. Cette recherche peut se faire en mode image ou en mode trame. Par exemple, un $f_$code 3–2 indique une fenêtre de largeur 64 pixels et de hauteur 32 pixels sur une image ou 16 pixels sur une trame (voir : figure 1 et tableau 1).



Fv : hauteur de la fenêtre
Fh : largeur de la fenêtre

Figure 1. – Estimation du mouvement d'un macrobloc.

Les dimensions de la fenêtre sont choisies en fonction du déplacement des motifs entre l'image courante et l'image de référence. Il existe, pour chaque séquence vidéo, une taille de fenêtre optimale qu'il n'est pas nécessaire de dépasser afin d'éviter des calculs inutiles. Pour une séquence d'images dont les mouvements sont

Tableau 1. – Taille de la fenêtre de recherche en fonction des paramètres f_code.

$f_$code	Fh	Fv	
		mode image	mode trame
1	16	16	8
2	32	32	16
3	64	64	32
4	128	128	64
5	256	256	128

lents (séquence « mobile&calendar »), un $f_$code horizontal égal à 3 et un $f_$code vertical égal à 2 sont suffisants (voir en annexe l'illustration des différentes séquences utilisées dans cet article). Pour une séquence rapide (séquence de sport type « football »), ce couple de paramètres peut atteindre les valeurs 5 et 4. La qualité de l'image compensée se dégrade très brutalement si la taille de la fenêtre de recherche devient inférieure aux valeurs adéquates.

La norme MPEG propose plusieurs méthodes pour estimer le déplacement d'un macrobloc. La méthode de recherche intégrale (RI) [2] donne, dans le domaine de la télévision, les meilleurs résultats en termes de robustesse et de qualité d'image avec une charge de calcul très importante [3][4]. La diminution de la taille de la fenêtre est le seul moyen d'augmenter la vitesse de la recherche, ce qui dégrade fortement la qualité des images compensées dans les séquences ayant des mouvements rapides. La mise au point d'algorithmes d'estimation de mouvement plus efficace que la méthode RI a fait l'objet de nombreuses recherches ces dernières années. Deux grandes familles de méthodes sont à distinguer, les algorithmes pyramidaux et les algorithmes pel–récursifs. Dans le domaine de la télévision, la robustesse de l'algorithme est une qualité essentielle à cause de la variété des images traitées ce qui élimine d'emblée les méthodes pel–récursives. Une méthode pyramidale, c'est–à–dire une méthode estimant le mouvement à partir d'une pyramide d'images de résolutions différentes, semble être le meilleur moyen d'obtenir un algorithme performant.

Le temps de traitement considérable nécessaire à la compression de séquences vidéo au format télévision numérique studio (plusieurs jours sur une station de travail HP9000–755) avec une méthode RI nous a conduit à chercher un algorithme d'estimation de mouvement ayant les caractéristiques suivantes :

1. **robustesse de la recherche intégrale** (compte tenu de la diversité des séquences à traiter en télévision).

2. **adaptation du temps de calcul au temps disponible pour le traitement ou à la puissance de l'ordinateur** (la qualité des images s'accroît quand le temps disponible augmente, mais le traitement est toujours possible très rapidement en cas d'urgence).

3. **dégradation *prévisible* et modérée de la qualité de l'image en fonction du temps de calcul.** (Ce critère nous conduit plutôt à privilégier une solution simple d'un point de vue conceptuel).

Nous proposons dans cet article un algorithme d'estimation de mouvement de type pyramidal simplifié remplissant les trois critères cités précédemment. Il n'est pas nécessairement plus rapide que les meilleurs algorithmes pyramidaux déjà étudiés, mais il nous semble mieux adapté à la compression de séquences vidéo avec un codeur MPEG–2.

2. les algorithmes proposés par la norme MPEG

2.1. algorithme de recherche intégrale (RI)

Le meilleur macrobloc est recherché systématiquement sur toute la fenêtre dans l'image de référence. Plusieurs critères d'appariement des macroblocs sont utilisables. L'erreur quadratique moyenne donne les meilleurs résultats, mais on préfère utiliser le critère de la différence absolue qui est plus rapide à calculer. Si deux macroblocs ou plus, dans l'image de référence, donnent la même erreur, le macrobloc qui présente la distance minimale avec le macrobloc courant est utilisé.

2.2. algorithme de recherche logarithmique (RL)

Cette méthode [2] permet de trouver un macrobloc de référence très rapidement en procédant par étapes. La première étape porte sur la taille initiale de la fenêtre de recherche. On prend 9 points sur cette fenêtre, puis on compare le macrobloc à compenser avec chacun des 9 macroblocs de référence et on retient la position du macrobloc qui présente le minimum d'erreur. Pour l'étape suivante, on divise par deux la plage de recherche que l'on centre sur le macrobloc précédemment défini. Le processus s'arrête quand la plage de recherche est égale à un macrobloc. La figure 2 illustre cette méthode par un exemple. Les chiffres « 1 », « 2 » ou « 3 » représentent les pixels centraux des macroblocs de référence. La première étape porte sur les macroblocs « 1 » qui sont espacés de 4 pixels. La meilleure position est utilisée comme centre pour la deuxième étape composée des macroblocs « 2 ». La zone de recherche est divisée par deux lors d'un changement d'étape. Après la troisième étape, la zone de recherche est égale à un macrobloc, et on considère le macrobloc 3 encerclé comme le meilleur macrobloc.

Cette méthode est effectivement rapide [5], mais elle détermine rarement le meilleur macrobloc de référence. La qualité de l'image compensée est fortement dégradée par rapport à la méthode RI.

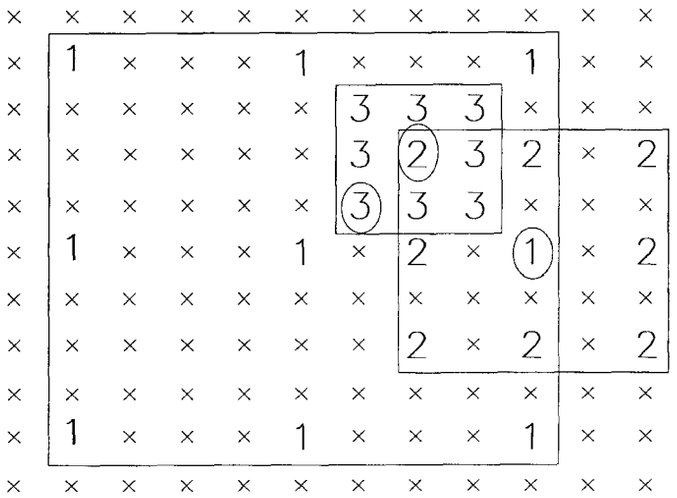


Figure 2. – Recherche logarithmique.

2.3. amélioration de la recherche par l'interpolation au « demi-pixel »

L'estimation d'un macrobloc trouvé par un algorithme de recherche au pixel près (RI ou RL) est affinée par une recherche au demi–pixel. Cela revient à déplacer le macrobloc d'un demi–pixel dans les huit directions possibles. Une interpolation linéaire est utilisée avec la convention de l'arrondi à l'entier le plus proche. Considérons les 4 pixels voisins ayant les valeurs A, B, C et D de la figure 3. Les valeurs des pixels interpolés horizontalement sont données par $h1 = (A+B)/2$ et $h2 = (C+D)/2$, les valeurs des pixels interpolés verticalement sont données par $v1 = (A+C)/2$ et $v2 = (C+D)/2$. La valeur du pixel central c est donné par $c = (A+B+C+D)/4$. La recherche au demi-pixel permet d'améliorer le rapport signal sur bruit crête d'une séquence codée MPEG–2 d'environ 0,5 à 1 dB.

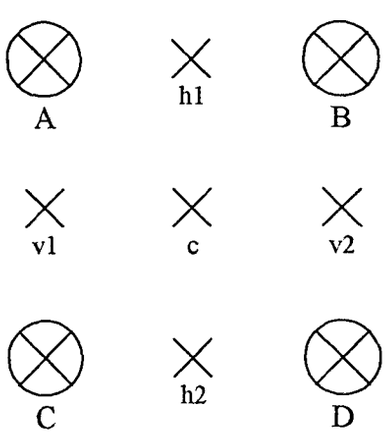


Figure 3. – Recherche au demi-pixel.

3. nouvelle méthode de recherche

3.1. présentation

La partie de l'algorithme RI qui demande le plus de calculs est la comparaison entre le macrobloc courant et les macroblocs de l'image de référence appartenant à la fenêtre de recherche. Pour diminuer cette charge de calculs, nous décidons de placer une première étape de recherche intégrale sur des images de définition la plus faible possible afin de présélectionner les meilleures solutions. Une décimation d'un facteur 4 a été choisie car un facteur 8 entraîne une perte d'information qui ne permet plus d'estimer le mouvement. La diminution d'un facteur 4 des dimensions de l'image entraîne la même réduction sur la taille du macrobloc et de la fenêtre de recherche. La quantité de calculs sera donc 256 fois moins élevée pour une RI basse résolution que pour une RI pleine résolution (fenêtre de recherche et macrobloc 16 fois plus petits, soit un gain de 16 × 16 en charge calcul). Au lieu de chercher seulement un macrobloc de référence sur l'image basse résolution, nous allons retenir les N meilleures solutions (appelées *essais*). Ces N solutions vont ensuite être essayées sur les images pleines résolutions. Cette méthode est schématisée à la figure 4. Le paramètre N nous servira à ajuster la vitesse de recherche de l'algorithme.

Il est à noter que cet algorithme comportait initialement une phase de recherche intermédiaire sur une image de résolution $1/2$. Cette phase a été éliminée car elle ne diminuait pas le temps de calcul de manière significative.

3.2. filtrage et décimation quart de bande

Nous cherchons à réduire les dimensions d'une image d'un facteur quatre en conservant au mieux l'information de mouvement. Comme cette information se trouve essentiellement dans les basses fréquences, l'image est filtrée par un filtre passe–bas quart de bande, puis décimée d'un facteur 4. La composante de luminance étant la seule utilisée dans le traitement d'estimation de mouvement, elle est la seule à être filtrée. Le filtrage vertical se fera sur l'image et sur chaque trame prise séparément. L'image filtrée et décimée est appelée *image basse résolution*. Ce processus est schématisé à la figure 5.

Nous allons utiliser le même filtre passe–bas pour le filtrage horizontal et vertical. Ce filtre a une grande importance pour l'estimation de mouvement. Pour déterminer un filtre efficace, nous considérerons les aspects suivants : la précision des calculs, la réjection spectrale maximale, le traitement des transitions ainsi que l'ondulation dans la bande passante.

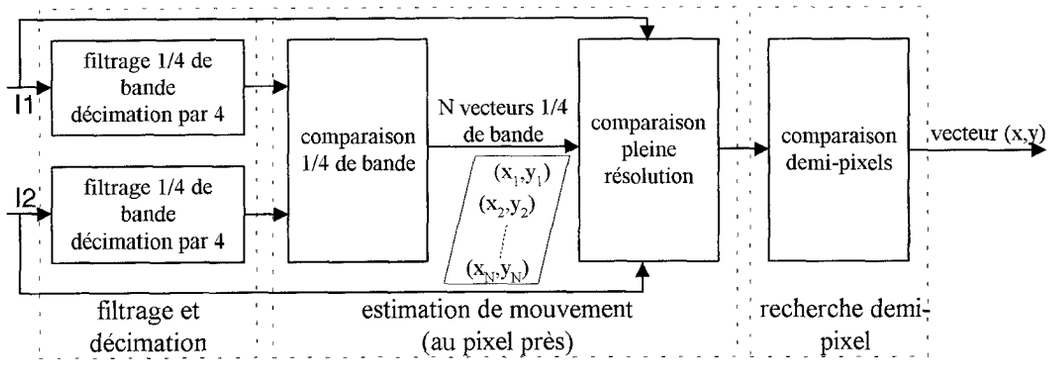


Figure 4. – Méthode de recherche basse résolution.

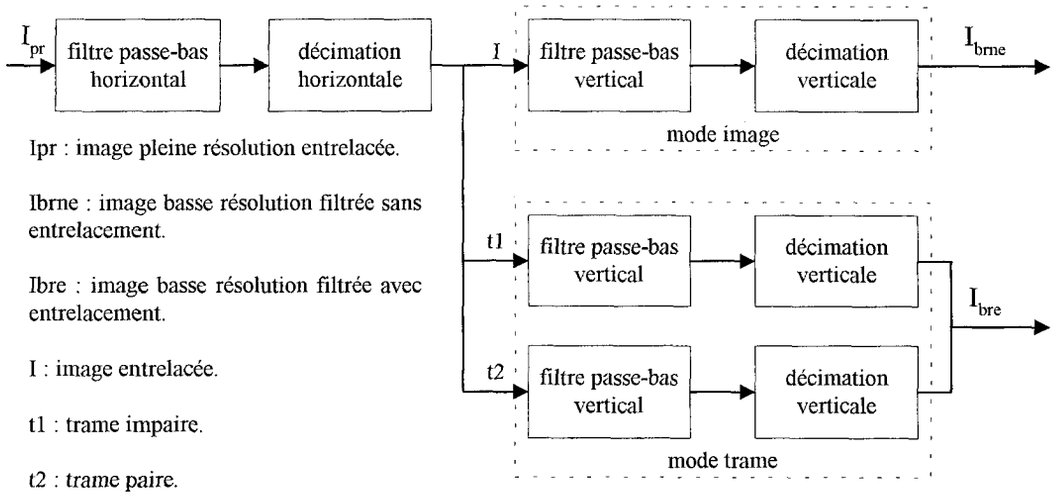


Figure 5. – Filtrage - décimation quart de bande.

3.3. estimation de mouvement

Une recherche intégrale est appliquée sur l'image et sur chacune des trames quart de bande. Les macroblocs trouvés sur la plage de recherche basse résolution, aux coordonnées x et y, sont ensuite ramenés sur la plage de recherche pleine résolution, aux coordonnées $4x$ et $4y$. Une recherche pleine résolution de ± 2 pixels autour de chaque macrobloc permet alors de définir l'emplacement des macroblocs ressemblant le plus au macrobloc originel.

La localisation du meilleur macrobloc est rendue possible par la mémorisation de N meilleures solutions sur la plage de recherche basse résolution. Le temps de recherche basse résolution est fonction du nombre d'essais N et des paramètres $f_$code horizontaux et verticaux, tandis que le temps de recherche pleine résolution n'est fonction que du nombre d'essais N.

Nous avons défini une formule exprimant N en fonction des dimensions de la fenêtre de recherche afin d'obtenir des résultats proches de ceux calculés par la méthode RI. Pour cela, nous avons calculé combien d'essais N étaient nécessaires pour obtenir une dégradation de rapport signal sur bruit inférieure à 0.5 dB (seuil de visibilité des défauts) en fonction des paramètres $f_$code horizontaux et verticaux, et ce pour les 5 séquences utilisées dans cet article (voir : annexe). Nous avons déterminé la formule suivante :

$$N = \max \cdot (1, 2^{(f\_\text{code\_horizontal} + f\_\text{code\_vertical} - 3)})$$

soit :

Tableau 2. – Nombre d'essais en fonction de la taille de la fenêtre de recherche.

f_code_horizontal	1	1	2	3	3	4	4	5	5	6	6
f_code_vertical	1	2	2	2	3	3	4	4	5	5	6
N	1	1	2	4	8	16	32	64	128	256	512

A partir de N, nous avons défini le paramètre N' afin de privilégier la qualité de l'estimation du mouvement ($M1$) ou la rapidité de la recherche ($M4$).

Tableau 3. – Définition du paramètre N'.

Sélecteur	$M1$	$M2$	$M3$	$M4$
N'	$2 \cdot N$	N	$N/2$	1

L'estimation de mouvement est finalement complétée par une recherche au demi–pixel.

4. implantation et comparaison avec la recherche intégrale

4.1. filtre passe-bas

Nous avons cherché un filtre passe–bas $1/4$ de bande qui donne l'estimation de mouvement la plus proche possible de la méthode RI (voir : §4.3) pour un codeur MPEG–2. Le processus de compression MPEG–2 est très complexe et la détermination a priori des caractéristiques du filtre permettant d'obtenir la meilleure qualité d'image après codage est difficile. Nous avons donc pris comme base de départ un filtre classique en télévision : un filtre pondéré FIR à phase linéaire d'ordre impaire ayant une accentuation dans les fréquences élevés pour améliorer la résolution. Puis nous avons procédé à différents essais afin de fixer précisément les paramètres suivants :

• type de filtre : le filtrage bi–dimensionnel est moins rapide que le filtrage mono–dimensionnel et n'améliore pas l'estimation de mouvement.

• nombre de coefficients : l'augmentation du nombre de coefficients du filtre améliore le gabarit spectral, mais la réponse temporelle entraîne des traînées sur les transitions. On obtient une estimation de mouvement plus efficace avec un filtre à 31 coefficients qu'avec un filtre à 9 coefficients à condition d'utiliser une fenêtre de pondération.

• fenêtre de pondération : les meilleures estimations de mouvement sont obtenues avec l'application d'une fenêtre de Hamming sur les coefficients du filtre car cette pondération diminue les traînées sur les transitions. Un filtre pondéré à 31 coefficients semble être le meilleur compromis pour des séquences d'images variées.

• accentuation : une accentuation d'un dB au milieu de la bande passante du filtre augmente la résolution de l'image filtrée et améliore la précision de l'estimation de mouvement.

• précision des calculs : une précision suffisante est obtenue quand les calculs sont effectués avec des entiers codés sur 16 bits.

Compte tenu de ces différents essais, nous avons finalement sélectionné le filtre à 31 coefficients pondéré par une fenêtre de Hamming suivant :

Tableau 4. – Coefficients du filtre (entiers sur 16 bits).

$H(0)$	$= H(30)$	$= -54$
$H(1)$	$= H(29)$	$= -42$
$H(2)$	$= H(28)$	$= -3$
$H(3)$	$= H(27)$	$= 66$
$H(4)$	$= H(26)$	$= 130$
$H(5)$	$= H(25)$	$= 94$
$H(6)$	$= H(24)$	$= -148$
$H(7)$	$= H(23)$	$= -623$
$H(8)$	$= H(22)$	$= -1172$
$H(9)$	$= H(21)$	$= -1423$
$H(10)$	$= H(20)$	$= -919$
$H(11)$	$= H(19)$	$= 632$
$H(12)$	$= H(18)$	$= 3116$
$H(13)$	$= H(17)$	$= 5928$
$H(14)$	$= H(16)$	$= 8164$
	$= H(15)$	$= 9050$

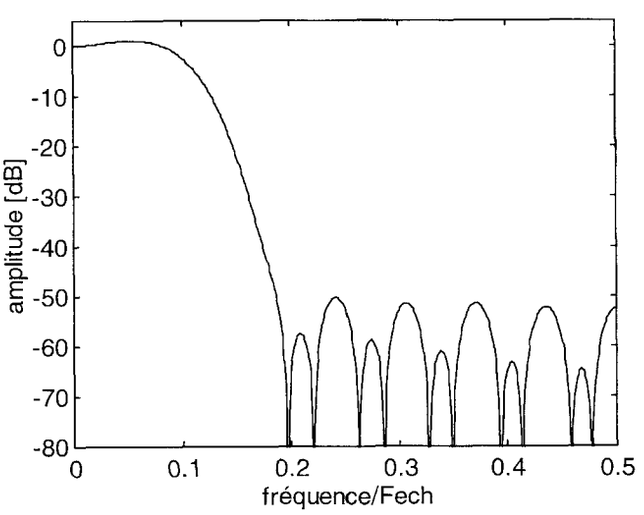


Figure 6. – Réponse en fréquence du filtre passe-bas.

4.2. critère de comparaison

Nous avons testé l'influence des critères de comparaison (erreur quadratique et erreur absolue) entre macroblocs sur la qualité des images compensées. Le critère de l'erreur quadratique a été retenu pour la recherche basse résolution car il permet d'améliorer

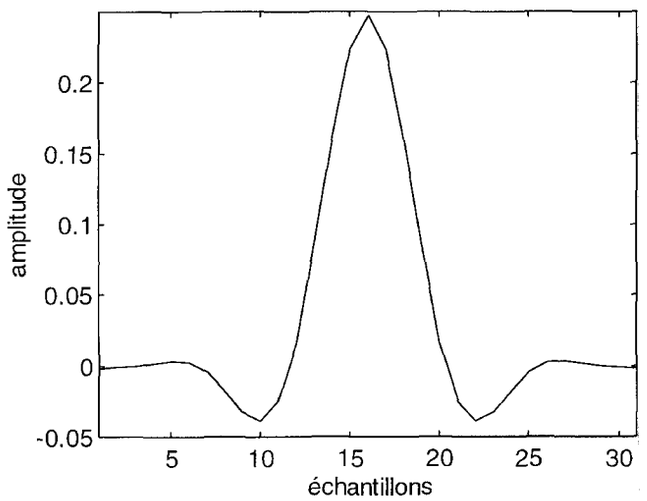


Figure 7. – Réponse impulsionnelle du filtre passe-bas.

la qualité de l'image compensée sans trop pénaliser le temps de traitement. Pour la recherche pleine résolution à partir des solutions calculées en basse résolution, le nombre de comparaison des pixels est très important. On garde le critère de l'erreur absolue qui permet un gain de temps non négligeable.

4.3. comparaison avec la recherche intégrale

Nous avons comparé notre méthode et la méthode RI en regardant la qualité des images compensées ainsi que le nombre de vecteurs de mouvement qui diffèrent entre les deux méthodes. Le rapport signal sur bruit crête (PSNR) a été utilisé pour comparer la qualité des images. En utilisant les séquences mobile&calendar (vitesse lente) et football (vitesse rapide), nous avons compensé des images P (images prédites) à partir d'images I (images de références codées par un traitement de type JPEG). Ces images P sont situées à une distance de trois images des images de références I comme c'est le cas dans les groupes d'images généralement utilisés dans MPEG (voir : figure 8).

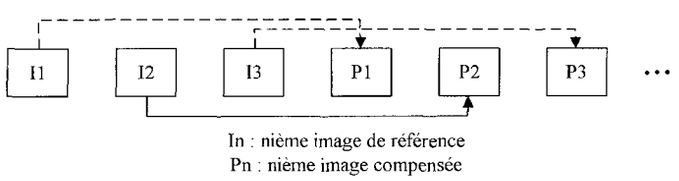


Figure 8. – Compensation d'images P.

Les courbes 1a et 2a des figures 9 et 10 montrent la convergence de la qualité d'image de notre méthode vers celle de la méthode RI en fonction de N. Sur les courbes 1b et 2b, nous avons représenté le nombre de macroblocs compensés qui diffèrent entre les deux méthodes (nous les appellerons macroblocs erronés pour la suite de cet article). La position de N (64 en $f_$code 5–4 et 4 en $f_$code 3–2) est repérée en trait continu sur ces figures. Une dégradation très nette est observée pour $N = 1$. Avec la valeur par défaut de N (sélecteur = $M2$), l'image compensée par notre méthode est visuellement identique à l'image obtenue avec la méthode RI, mais des macroblocs compensés différents subsistent encore. Nous avons comparé visuellement ces macroblocs avec ceux trouvés par la méthode RI. Ils diffèrent très peu, ce qui explique la faible dégradation apportée aux séquences vidéo codées MPEG–2 dans ce mode.

Nous avons ensuite compensé avec les deux méthodes de recherche les quatre séquences tests suivantes : mobile&calendar, tennis, football et basket. Cette compensation a été réalisée à partir des images originales avec une distance de 3 images entre image I et P. Les courbes 3a à 3d de la figure 11 montrent que les deux méthodes donnent des images compensées de qualité comparable quelle que soit la séquence d'images testée. Les quelques différences parfois constatées entre les courbes ne sont pas le signe de dégradations visibles lorsque l'on observe les images obtenues.

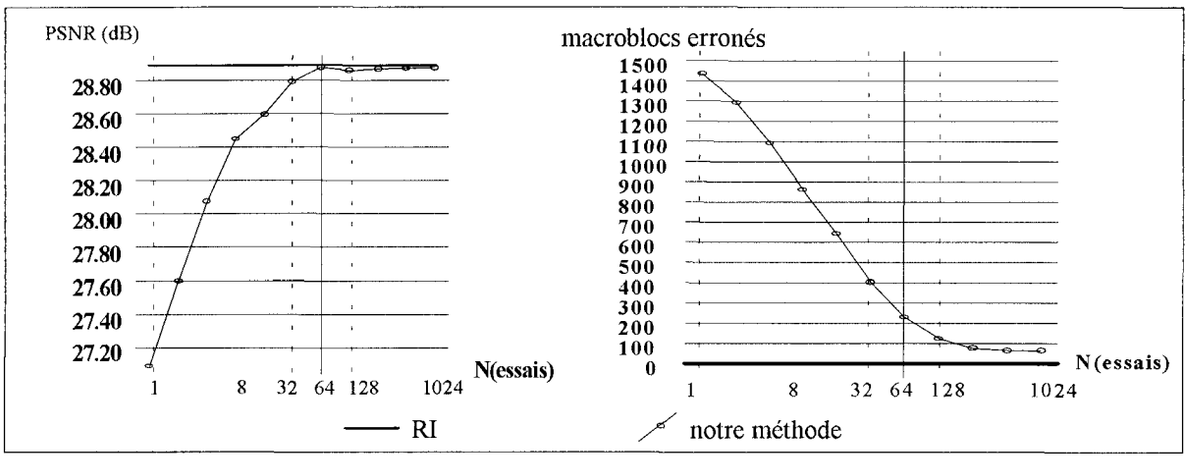
5. application de la méthode basse résolution au codeur MPEG-2

Les différents tests auxquels nous nous sommes livrés au paragraphe précédent nous laissent penser que nous avons obtenu un algorithme d'estimation de mouvement robuste et efficace. Nous devons maintenant le vérifier en l'implantant dans un codeur MPEG et en mesurant sa vitesse.

Pour cela, nous avons implanté dans un codeur vidéo MPEG–2 (Test Model 5) notre méthode ainsi que les algorithmes de recherche intégrale (RI) et de recherche logarithmique (RL). Une séquence test composée d'images mobile&calendar, flower garden, tennis et football a été appliquée à l'entrée du codeur. Nous avons testé les trois méthodes de recherche en comparant la qualité des images décodées et le temps de codage. La séquence test a été codée dans les deux modes suivants : $f_$code 3–2 et débit de 6Mbit/s, $f_$code 5–4 et débit de 3,5Mbit/s.

La moyenne des courbes de PSNR en luminance obtenues avec les différentes méthodes est présentée dans le tableau 5. Pour notre méthode, nous avons utilisé les quatre sélecteurs ($M1$, $M2$, $M3$ et $M4$) correspondant aux différentes valeurs du paramètre N.

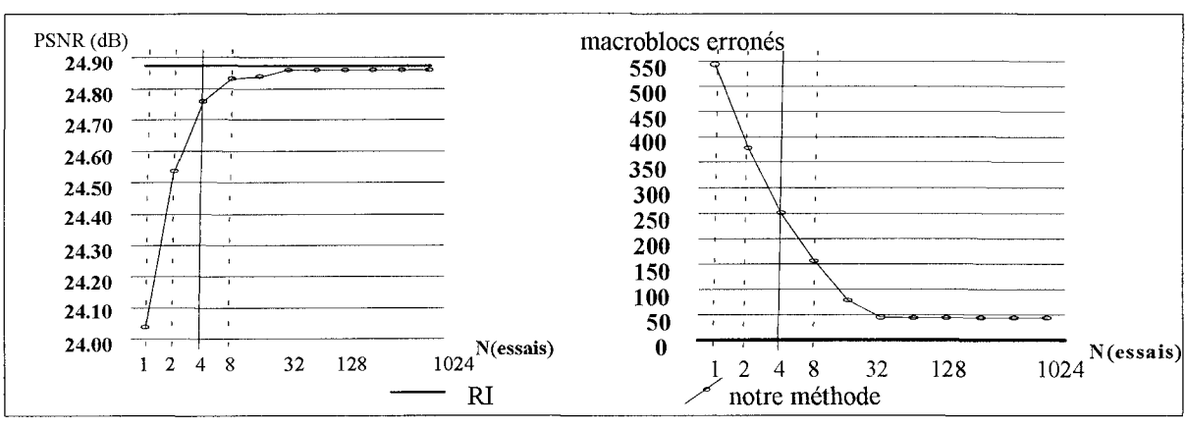
En prenant le temps de codage de la séquence codée avec la méthode RI comme référence, le tableau 6 montre les gains de vitesse d'exécution apportés par notre méthode et par la méthode logarithmique sur une station de travail.



1a : Convergence du Peak SNR

1b : Convergence des macroblocs erronnés

Figure 9. – Séquence football avec un f_code 5-4.



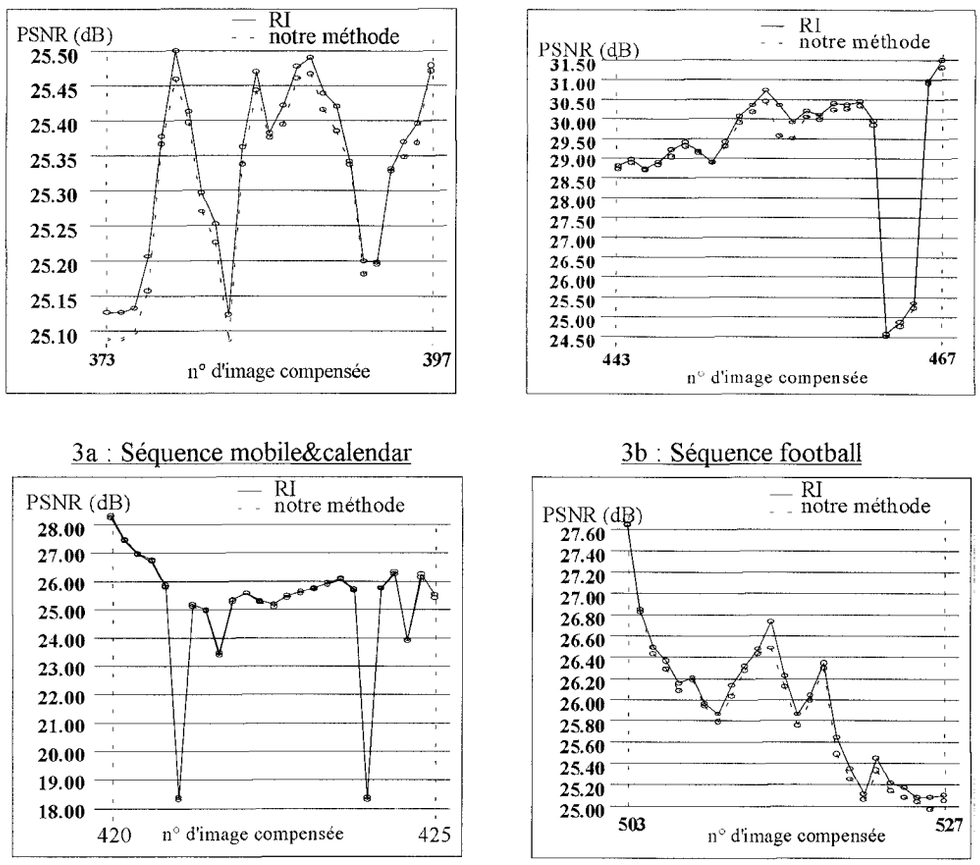
2a : Convergence du Peak SNR

2b : Convergence des macroblocs erronés

Figure 10. – Séquence mobile& calendar avec un f_code 3-2.

Tableau 5. – PSNR pour différentes séquences test.

	mobile		flower		tennis		football	
	6 Mbit/s	3,5 Mbit/s	6 Mbit/s	3,5 Mbit/s	6 Mbit/s	3,5 Mbit/s	6 Mbit/s	3,5 Mbit/s
RI	29.5	26.5	31.5	27.5	34	32	35	32.5
M1	29.4	26.5	31.4	27.5	33.9	32	34.9	32.5
M2	29.3	26.4	31.3	27.4	33.8	31.9	34.8	32.4
M3	29.2	26.3	31.2	27.3	33.8	31.8	34.7	32.3
M4	29	25.5	31	27	33.5	31.2	34.5	31.8
RL	26.5	24	28.5	24.5	32.5	29	34	28.5



3a : Séquence mobile&calendar

3b : Séquence football

3c : Séquence basket

3d : Séquence tennis

Figure 11. – PSNR en luminance pour les méthodes RI et basse résolution (M2).

Tableau 6. – Vitesses d'exécution.

	recherche intégrale	notre méthode			recherche logarithmique	
$f_$code	Tps de codage [s]	essais N	Tps de codage [s]	**gain**	Tps de codage [s]	gain
3–2	30297 (8h25)	1 ($M4$)	4910 (1h22)	**6.2**	5169 (1h26)	5.8
		2($M3$)	5316 (1h28)	**5.7**		
		4 ($M2$)	6556 (1h49)	**4.6**		
		8 ($M1$)	9306 (2h35)	**3.3**		
5–4	395782 (109h56)	1 ($M4$)	9904 (2h45)	**40**	5554 (1h32)	71
		32 ($M3$)	39055 (10h51)	**10.1**		
		64 ($M2$)	64150 (17h49)	**6.1**		
		128 ($M1$)	106446 (29h34)	**3.7**		

La méthode logarithmique est une méthode rapide mais la qualité d'image obtenue n'est pas satisfaisante. Cette méthode présente peu d'intérêt et on lui préférera notre méthode. Avec la valeur $N = 1$, notre méthode est un peu moins rapide que la méthode logarithmique pour un $f_$code 5–4 (ou un peu plus rapide pour un $f_$code 3–2) mais la qualité de l'image compensée est nettement supérieure. Pour obtenir une image de qualité RI, notre méthode sera utilisée avec un sélecteur $M2$ ou $M3$, $M2$ pour un maximum de sécurité et $M3$ pour un maximum de vitesse. Le sélecteur $M1$ est peu utilisé car l'amélioration de la qualité d'image par rapport à $M2$ est trop faible pour pouvoir compenser sa lenteur.

6. conclusion

Nous avons décrit dans cet article un algorithme d'estimation de mouvement aussi robuste que la méthode de recherche intégrale

et donnant des images compensées de qualité identique. Nous avons implanté cet algorithme dans un codeur MPEG–2 que nous utilisons avec satisfaction au laboratoire pour coder toutes nos séquences depuis environ 2 ans. En ajustant le sélecteur de vitesse, nous pouvons coder des séquences vidéo avec des fenêtres de recherche réalistes et un niveau de qualité correspondant au temps disponible pour réaliser le codage.

BIBLIOGRAPHIE

[1] Organisation Internationale de Normalisation, ISO/IEC JTC1/SC29/WG11, DIS 13818–2 vidéo, Genève, 1995.

[2] Arun N. Netravali, Barry G. Haskell, « Digital pictures, representation and compression », Plenum Press, New York, 1988.

[3] P. Pirch, T. Komarek, « VLSI architecture for block–matching algorithms », *Visual communications and image processing '88 Proc.*, vol. 1001, 1988, pp. 882–891.

[4] K.M. Yang, M.T. Sun, L. Wu, « A family of VLSI designs for the motion compensation block–matching algorithm », *IEEE Trans. Circuit Syst.*, vol. 36, n°10, Oct. 1989, pp. 1317–1325.

[5] L.G. Chen, Y.S. Jehng, T.D. Chiueh, « VLSI design of motion estimator for HDTV application », *International workshop on HDTV '92 Proc.*, vol. I, Nov. 1992, Tokyo, pp. 361–368.

Manuscrit reçu le 4 Septembre 1996.

LES AUTEURS

Christophe ALEXANDRE

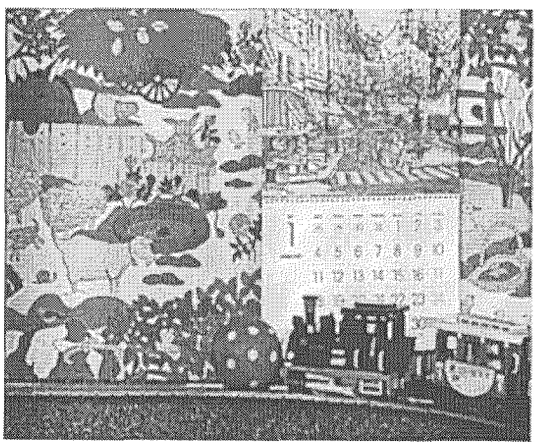
Est docteur de l'Université et ingénieur CNAM en électronique. Il enseigne l'électronique au laboratoire Electronique et Communications du CNAM Paris.

Han VU THIEN

Han Vu Thien est ingénieur Sup' télécom et docteur d'état. Il est Professeur des Universités et dirige le laboratoire Electronique et Communications du CNAM Paris.

ANNEXE

Illustration des séquences traitées

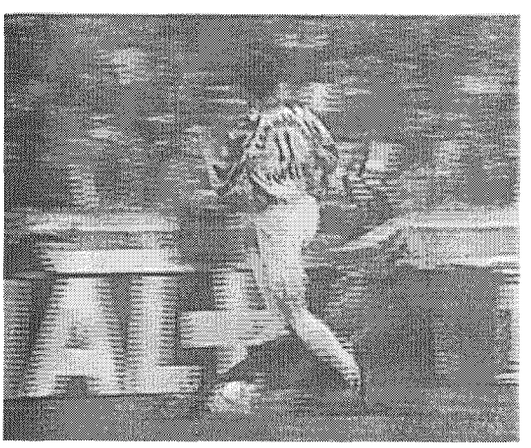


Mobile and Calendar

Flower Garden

Tennis Table

Basket



Football