REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT D'ELECTRONIQUE
OPTION : TELECOMMUNICATIONS

MEMOIRE DE MAGISTER

Présenté par :
BAILICHE Mohamed Amine
Ingénieur d'état C.U. de Médéa

Etude Comparative des Algorithmes d'Estimation de Mouvements dans l'Image Vidéo par la Technique de "Block Matching"

Président :	AKSAS Rabia	Pr	ENP.
Rapporteurs :	MEHENNI Mohamed	Pr	ENP.
	LAIDI Kamel	CC	C.U. de Médéa.
Examinateurs :	AIT DAOUD Sihem	CC	INI.
	BOUSBIA-SALAH Hichem	Dr	ENP.
	MOUSSAOUI Aicha	CC	ENP.

Octobre 2006

ملخص

الهدف من عملنا هذاهو دراسة مقارنة بين خوارزميات تقدير الحركة، المستعملة في صور الفيديو، باستخدام طريقة الأشكال المتوافقة. تحديد الحركة هو مرحلة جد مهمة في تقليص كمية صور الفيديو. باستعمال التداخل الزمني بين صورتين متتابعتين من نفس لقطة الفيديو نستطيع تقليص حجم الذاكرة اللازمة لتخزين الصورة دون تضييع المعلومات. في هذا الإطار قمنا بتطبيق مختلف خوارزميات البحث المستعملة في إطار طريقة الأشكال المتوافقة، و هي : 'خوارزمية البحث الكامل'، 'خوارزمية البحث على شكل الماس'، 'خوارزمية البحث على ثلاثة مراحل'، 'خوارزمية البحث على ثلاثة مراحل الجديدة'، 'خوارزمية البحث على أربعة مراحل'، 'خوارزمية البحث عن الأصغر'. فعالية الخوارزمية هي عبارة عن توفيق بين نسبة الإشارة على التشويش و الوقت المستغرق في الحساب.

الكلمات المفاتيح:

تقدير الحركة، صور الفيديو، طريقة الأشكال المتوافقة.

Résumé

L'objectif de notre travail consiste en une étude comparative des algorithmes d'estimation de mouvement, utilisés dans les images vidéo, par la méthode de mise en correspondance par bloc « block matching ». L'estimation de mouvement est une étape très importante pour la compression des images vidéo. En exploitant la redondance temporelle entre deux frames successives de la séquence nous pouvons efficacement réduire la masse mémoire nécessaire pour le stockage de l'image, sans perte d'informations. Dans ce contexte, nous avons étudié et appliqué les différents algorithmes de recherche utilisés par la méthode de mise en correspondance par blocs à savoir : l'algorithme full Search 'FS', Diamond Search 'DS', Tree Step Search 'TSS', New Tree Step Search 'NTSS', Four Step Search 'FSS', Minimum Search 'MS'. La performance d'un algorithme est un compromis entre le Rapport Signal sur Bruit et le temps de calcul.

Mots clés :

Estimation de Mouvement, image vidéo, mise en correspondance par blocs.

Abstract

The purpose of this work consists in a comparative study of motion estimation algorithms, used in video images by Block Matching Technique. Motion estimation is a very important step in video image compression. By exploiting a temporal redundancy between two successive sequence frames we can efficiently reduce the necessary memory space for image storage, without information loss. In this context, we have studied and applied the different algorithms of search used in Block Matching technique. These algorithms are: full Search algorithm 'FS', Diamond Search algorithm 'DS', Tree Step Search algorithm 'TSS', New Tree Step Search algorithm 'NTSS', Four Step Search 'FSS', Minimum Search algorithm 'MS'. The performance of an algorithm is a compromise between Signal to Noise Ratio and computing time.

Keys words :

Motion Estimation, vidéo image, Block Matching.

Remerciements

Au terme de ce mémoire je tiens à remercier tout naturellement en premier lieu ***DIEU Le Tout Puissant*** *qui nous a donné la force, le courage et la patience de bien mener ce travail.*

Ce travail a été réalisé sous la direction de Mr. ***MEHENNI Mohamed*** *notre promoteur, et Mr.* ***LAIDI Kamel*** *notre co-promoteur, qu'ils trouvent ainsi l'expression de ma profonde reconnaissance pour leurs aides, leurs encouragements et leurs précieux conseils durant le déroulement de ce travail.*

Je tiens à exprimer mes sincères remerciements à tous ***les enseignants de PG Electronique option Télécommunications*** *à l'Ecole Nationale Polytechnique, EL-HARRACH.*

Je remercier Mr. ***AKSAS****, qui a voulu assurer la présidence de notre jury ainsi que Mme.* ***AIT DAOUD****, et Mr.* ***BOUSBIA*** *et Mme* ***MOUSSAOUI*** *d'avoir accepté l'évaluation de ce modeste travail.*

Mes sentiments vont également à toutes les personnes qui ont contribué de prés ou de loin à la réalisation de ce modeste travail.

Dédicaces

Au delà des personnes, des lieux et des temps, je dédie ce travail avec ma profonde conviction, à tous ceux qui ont toujours cru sans doute à la science. Cette lumière qui éclaire les esprits et leurs permet de transcender les limites installées par les sociétés et les cultures dans leurs périodes de dégénérescence.

A ceux qui savent concrétiser leurs idées et travaillent pour réussir.
A ceux qui ne portent que du bien pour les autres.
A ceux qui ne vivent que pour la vérité.

Je dédie ce travail à mes ***chers parents****.*
A mes frères Fethi et Redha, à mes sœurs Faiza, Samia et Hadjer.
A mes chères nièces Nesrine et Yasmine, et à leur père Mohamed

A toute ma grande famille à Médéa, Blida, et Alger, surtout : Youcef, Karim, Hocine, Hamza, Bilal, Abdennour, Krimo, Bilal.

A l'esprit de mon chèr oncle ***Abd Elkerim****.*

A tous mes amis de la promotion électronique 2003 du centre universitaire de Médéa sans exception.

A tous mes amis de la promotion PG électronique 2004 surtout 'télécommunications' de l'ENP EL-HARRACH.

A tous mes amis de la résidence universitaire BOURAOUI, surtout : Samir, Mohamed, Ismail et Halim.

A tous mes amis : Hichem, Boubaker, Mustapha, Boualem, Hamza, Djamel, Toufik, les deux Raouf et les deux Sidali, Abd El-Moghni, Kader, Nourredine, Mourad, Dadou, Kakou, Zohir, Mohamed, Sid-Ahmed, Ilyas, Hammido et Redha.

Et à tous ceux qui me connaissent.

e-mail : bailiche_amine@yahoo.fr

SOMMAIRE :

Chapitre I :

Notions de base sur l'image et la compression numérique

Introduction générale :

Dans les applications multimédia l'exigence principale est la vitesse de traitement et la compression sans sacrifie dans la qualité de l'image. Dans le but de traiter une séquence vidéo de haute qualité nous avons besoin d'un algorithme d'estimation (qui est un composant essentiel pour la majorité des codeurs vidéo MPEG-1, MPEG-2, MPEG-4, H.261, H.263,…) [38], [13], [25] qui augmente la vitesse de traitement.

La technique la plus populaire adoptée pour l'estimation de mouvement entre deux frames est la technique de mise en correspondance par blocs " Block Matching Algorithms " [41], dans laquelle un bloc de taille N × M de la frame actuelle (généralement 16 × 16) est comparé avec un bloc correspondant dans une zone dite zone de recherche dans la frame précédente. Trois éléments principaux doivent être prise en considération dans la technique BMA :

- Critère de mise en correspondance.
- Zone de recherche.
- Technique de recherche.

Cette dernière est la plus importante, qui joue un rôle critique dans la performance de l'algorithme BMA.

Dans la technique BMA deux problèmes essentiels doivent être pris en considération :

- Un problème de la recherche du minimum global (problème d'optimisation).
- Un problème de temps de calcul.

L'algorithme "Full Search : FS" aboutit à la solution optimale par une recherche exhaustive dans la zone de recherche de la frame précédente, mais au prix de la vitesse (temps de calcul) ; ce qui lui empêche d'être appliqué en temps réel. Un nombre d'algorithmes de recherche sont appliqués afin de réduire le temps de calcul et fournissent un vecteur de mouvement (ou de déplacement) sous-optimal. Ces algorithmes trouvent seulement un minimum local ; par conséquent la qualité de codeur vidéo vas diminue.

Notre travail entre dans ce cadre. Nous avons fait une étude comparative des différents algorithmes de recherche par la technique BMA.

La performance d'un algorithme est jugée selon deux critères [14] :

- Le PSNR : "Peak Signal to Noise Rate", le rapport pic de signal sur bruit.
- Le temps de calcul de chaque algorithme.

Pour élaborer ce travail, nous avons organisé notre thèse comme suit :

Dans le chapitre I on va donner des notions de base concernant les images numériques, la compression vidéo, la norme MPEG, et le rôle important de l'estimation de mouvement dans la compression/décompression MPEG.

Le chapitre II nous donne une notion sur les différentes méthodes d'estimation de mouvement, les méthodes différentielles (Méthode de Horn et Schunk, méthodes différentielles avancées..), les méthodes récursives, les méthodes statiques, et finalement un bref sur les méthodes de mise en correspondance par blocs.

Dans le chapitre III on discute les méthodes de mise en correspondance par blocs, et les six algorithmes les plus utilisés dans le domaine de la compression des séquences vidéo.

Le chapitre IV, le chapitre final inclus les résultats pratiques de notre travail.

Et finalement une conclusion générale de notre travail.

Chapitre I :

Notions de Base sur l'Image et la Compression Numérique

I.1. Introduction :

Chaque séquence vidéo est une combinaison de plusieurs frames (images), notre travail est une application sur les séquences vidéo, donc sur les images inclus dans les images vidéo.

Commençons par un petit rappel et des notions de base sur l'image numérique, ce chapitre donne toutes les clés permettant de comprendre les principes de la compression numérique appliquée à la vidéo, technique complexe qui, en s'imposant chaque jour davantage, a quasiment banalisé la manipulation des images vidéo numériques dans les différents domaines, la vidéo conférence, les multimédias, la téléphonie visuelle,…

Nous décrirons donc les normes de codage des images JPEG, MPEG-l, MPEG-2.

I.2. Notion de base sur l'image numérique :

I.2.1. Définition de l'image :

L'image est une représentation d'une personne ou d'un objet par la peinture, la sculpture, le dessin, la photographie, le film, etc.

C'est aussi un ensemble structuré d'informations qui, après affichage sur l'écran, ont une signification pour l'œil humain.

Elle peut être décrite sous la forme d'une fonction I(x, y) de brillance analogique continue, définie dans le domaine borné, tel que x et y sont les coordonnées spatiales d'un point de l'image et I est une fonction d'intensité lumineuse et de couleur. Sous cet aspect, l'image est inexploitable par la machine, ce qui nécessite sa numérisation.

I.2.2. Les images numériques :

Il existe deux sortes d'images numériques : les images matricielles et les images vectorielles.

Dans une image vectorielle les données sont représentées par des formes géométriques simples qui sont décrites d'un point de vue mathématique.

Par exemple, un cercle est décrit par une information de type (cercle, position du centre, rayon). Ces images sont essentiellement utilisées pour réaliser des schémas ou des plans.

Les logiciels de dessin industriel fonctionnent suivant ce principe ; les principaux logiciels de traitement de texte ou de PAO (Publication Assistée par Ordinateur) proposent également de tels outils.

Ces images présentent deux avantages : elles occupent peu d'espace en mémoire et peuvent être redimensionnées sans perte d'informations.

Une image matricielle est formée d'un tableau de points ou de pixels. Plus la densité des points est élevée, plus le nombre d'informations est grand et plus la résolution de l'image est élevée.

Corrélativement, l'espace occupé en mémoire et la durée de traitement seront d'autant plus grands.

Les images vues sur un écran de télévision ou une photographie sont des images matricielles. On obtient également des images matricielles à l'aide d'un appareil photo numérique, d'une caméra vidéo numérique ou d'un scanneur.

I.2.3. Les technologies d'affichage :

L'image s'affiche sur un écran (appelé aussi moniteur), il s'agit d'un périphérique de sortie permettant de fournir une représentation visuelle. Ces informations proviennent de l'ordinateur, mais de façon indirecte. En effet, le processeur n'envoie pas directement les informations au moniteur, mais traite les informations provenant de sa mémoire vive (RAM), puis les envoie à une carte graphique qui est chargée de convertir les informations en impulsions électriques qu'elle envoie au moniteur.

Les moniteurs des ordinateurs sont la plupart du temps des tubes cathodiques, c'est-à-dire un tube en verre dans lequel un canon à électrons émet des électrons dirigés par un champ magnétique vers un écran sur lequel sont disposés de petits éléments phosphorescents (luminophores) constituent des points (pixels) émettant de la lumière lorsque les électrons viennent les heurter.

I.2.4. La notion de pixel :

Une image est constituée d'un ensemble de points appelés pixels (pixel est une abréviation de *PICture ELement*). Le pixel représente ainsi le plus petit élément constitutif d'une image numérique. L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image :

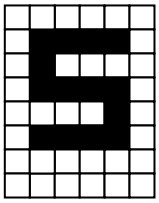


Figure I.1 : notion de pixel

Étant donné que l'écran effectue un balayage de gauche à droite et de haut en bas, on désigne généralement par les coordonnées [0,0] le pixel situé en haut à gauche de l'image, cela signifie que les axes de l'image sont orientées de la façon suivante :

- L'axe X est orienté de gauche à droite
- L'axe Y orienté de haut en bas, contrairement aux notations conventionnelles en mathématiques, où l'axe Y est orienté vers le haut.

I.2.5. Définition et résolution :

On appelle définition le nombre de pixels constituant l'image, c'est-à-dire sa dimension informatique (le nombre de colonnes de l'image que multiplie son nombre de lignes). Une image possédant 640 pixels en largeur et 480 pixels en hauteur aura une définition de 640 par 480 pixels, notée 640 × 480.

La résolution, terme souvent confondu avec la définition, détermine par contre le nombre de points par unité de surface, exprimé en points par pouce (PPP, en anglais DPI pour Dots Per Inch) ;

un pouce représente 2.54 cm. La résolution de 300 dpi signifie donc 300 colonnes et 300 rangés de pixels sur un pouce carré ce qui donne donc 90000pixels sur un pouce carré. La résolution de référence de 72 dpi nous donne un pixels de 1"/72 (un pouce divisé par 72) soit 0.353 mm, correspondant à un point pica (unité typographique anglo saxonne).

I.2.6. Le codage de la couleur :

une image est donc représentée par un tableau à deux dimensions dont chaque case est un pixel. Pour représenter informatiquement une image, il suffit donc de créer un tableau de pixels dont chaque case contient une valeur. La valeur stockée dans une case est codée sur un certain nombre de bits déterminant la couleur ou l'intensité du pixel, on l'appelle profondeur de codage (parfois profondeur de couleur). Il existe plusieurs standards de codage de la profondeur :

- **Bitmap noir et blanc :** en stockant un bit dans chaque case, il est possible de définir deux couleurs (noir ou blanc).
- **Bitmap 16 couleurs ou 16 niveaux de gris :** en stockant 4 bit dans chaque case, il est possible de définir 2^4 intensités de pixels, c'est-à-dire 16 dégradés de gris allant du noir au blanc ou bien 16 couleurs différentes.
- **Bitmap 256 couleurs ou 256 niveaux de gris :** en stockant un octet dans chaque case, il est possible de définir 2^8 intensités de pixels, c'est-à-dire 256 dégradés de gris allant du noir au blanc ou bien 256 couleurs différentes.
- **Palette de couleurs (colormap) :** grâce à cette méthode, il est possible de définir une palette, ou table des couleurs, contenant l'ensemble des couleurs pouvant être contenues dans l'image, à chacune desquelles est associé un indice. Le nombre de bits réservé au codage de chaque indice de la palette détermine le nombre de couleurs pouvant être utilisées. Ainsi en codant les indices sur 8 bits il est possible de définir 256 couleurs utilisables c'est-à-dire que chaque case du tableau à deux dimensions représentant l'image va contenir un nombre indiquant l'indice de la couleur à utiliser. On appelle ainsi image en couleurs indexées une image dont les couleurs sont codées selon cette technique.
- **True color :** cette représentation permet de présenter une image en définissant chacune des composantes (RGB : rouge, vert, bleu). Chaque pixel est représenté par un entier comportant les trois composantes, chacune codée sur un octet, c'est-à-dire au total 24 bits (16 millions de couleurs). Il est possible de rajouter une quatrième composante permettant d'ajouter une information de transparence ou de texture, chaque pixel est alors codé sur 32 bits.

I.2.7. Taille d'une image :

Pour connaître la taille (en octets) d'une image, il est nécessaire de compter le nombre de pixels que contient l'image, cela revient à calculer le nombre de cases du tableau, soit la hauteur de celui-ci que multiplie sa largeur. La taille (ou poids) de l'image est alors le nombre de pixels que multiplie la taille (en octets) de chacun de ces éléments.

Voici le calcul pour une image 640 × 480 en true color :
Nombre de pixels : 640 × 480 = 307200.
24 bits / 8 =3 octets.

Le poids de l'image est ainsi égal à :

307200 × 3 = 921600 octets.
921600 / 1024 = 900 Ko.

Voici un tableau de quelques exemples

Définition de l'image	Noir et blanc (1 bit)	256 couleurs (8 bits)	65000 couleurs (16 bits)	True color (24 bits)
320 × 200	7.8 Ko	62.5 Ko	125 Ko	187.5 Ko
640 × 480	37.5 Ko	300 Ko	600 Ko	900 Ko
800 × 6000	58.6 Ko	468.7 Ko	937.5 Ko	1.4 Mo
1024 × 768	96 Ko	768 Ko	1.5 Mo	2.3 Mo

Tableau 1 : poids des différents types d'images.

Cela met en évidence la qualité de mémoire vidéo que nécessite votre carte graphique en fonction de la définition de l'écran (nombre de points affichés) et du nombre de couleurs. L'exemple montre ainsi qu'il faut une carte ayant au minimum 4 Mo de mémoire vidéo afin d'afficher une résolution de 1024 × 768 en true color.

I.2.8. Transparence :

La transparence est une caractéristique permettant de définir le niveau d'opacité des éléments d'une image, c'est-à-dire la possibilité de voir à travers l'image des éléments graphiques située derrière celle-ci.

Il existe deux modes de transparence :

La transparence simple s'applique pour une image indexée et consiste à définir parmi la palette de couleurs une des couleurs comme transparence.

La transparence par couche alpha (ou canal alpha, en anglais alpha channel) consiste à rajouter pour chaque pixel de l'image un octet définissant le niveau de transparence (de 0 à 255). Le processus consistant à ajouter une couche transparente à une image est généralement appelé alpha blinding.

I.3. But de la compression numérique :

Si la compression numérique fait beaucoup parler d'elle depuis quelques années, il ne faut pas oublier que l'idée de réduire la quantité d'information du signal vidéo n'est guère nouvelle. Depuis les débuts de l'histoire de la télévision, on a toujours cherché à exploiter les caractéristiques psychovisuelles de l' œil humain pour restreindre à des valeurs raisonnables l' encombrement du signal vidéo: réduction du nombre d'images transmises par seconde tout d'abord, par rapport à la scène lumineuse continue captée par l'objectif de la caméra; réduction de la définition verticale de chaque image aussi, inhérente à la structure en lignes; limitation de la résolution horizontale également, le nombre de points par ligne étant directement lié à la bande passante électronique;

diminution de la définition de l'image encore, due cette fois à la structure en triplets de luminophores de l'écran du récepteur; réduction de la quantité d'informations de chrominance transmises enfin, compte tenu de l'incapacité de l' œil à discerner des différences de couleurs dans les détails finis.

Apparue au début des années 1980 la numérisation des images vidéo conformément à la norme 4:2:2 engendre des volumes de données gigantesques, donc des débits extrêmement élevés au regard de la capacité relativement limitée des supports de stockage et, surtout, de transmission existants. Pour donner un ordre d'idée, une image 4:2:2 quantifiée sur 8 bits occupe un espace de près de 830 kilo-octets (en ne considérant que la partie active), ce qui porte la seconde à 21 mégaoctets. Un CD-Rom de 650 Mo ne peut pas contenir plus de 30 s de vidéo non compressée, et un DVD de 4,7 Go même pas 4 mn. La réduction, du débit du signal vidéo est un passage obligé pour autoriser 'le stockage d'une -grande quantité d'images sur une bande magnétique ou un support informatique, et pour permettre la diffusion de ces images numériques dans un canal de fréquence conventionnel. En plus de ces économies en matière de stockage et de transmission, la compression des images ouvre la voie à des fonctionnalités totalement nouvelles dans les applications courantes qui bénéficient des avantages inhérents au numérique. Parmi eux, citons l'accès aléatoire à n'importe quelle image d'un programme, l'accès simultané au même contenue par plusieurs utilisateurs, ainsi que le transfert plus rapide que le temps réel. Bien que relativement récentes, les techniques de réduction de débit sont déjà très largement employées dans les domaines de l'enregistrement - broadcast et domestique -, la diffusion - câble, satellite, voie terrestre -, le montage non-linéaire et les applications multimédia.

Dans les systèmes 625/50, une image vidéo 4:2:2 échantillonnée sur 8 bits occupe un espace mémoire de 829440 octets (seule la partie active de l'image est prise en compte) :

720 (Y) + 360 (Cr) + 360 (Cb) = 1440 octets/ligne, soit 829 440 octets pour 576 lignes.

- Une seconde (25 images) occupe 20,7 mégaoctets.
- Une minute occupe 1 ,24 gigaoctets.
- Une heure occupe 74,6 gigaoctets.

On retiendra qu'un espace mémoire de 1 Go peut contenir 47 secondes d'images 4:2:2 sur 8 bits à plein débit.

Le signal vidéo numérique renferme un volume de données beaucoup trop important pour etre transmis ou enregistré tel que par des machines économiquement abordables.

La compression a pour but de trouver la manière la plus rationnelle de coder les images (en termes d'encombrement), tout en préservant au maximum leur contenu.

I.4. La compression numérique : comment :

L'art de la compression numérique en vidéo est de supprimer certaines informations de l'image et d'en simplifier d'autres, tout en faisant en sorte que les modifications apportées échappent le plus possible à la perception humaine. Cela est d'autant plus difficile que le débit final que l'on cherche à obtenir est faible. Dans le cas d'une image fixe, les techniques de compression s'appuient sur une analyse du contenu de l'image et tirent profit de son organisation interne afin d'en éliminer les données redondantes (une donnée redondante pouvant être déduite à partir des informations restantes). Par exemple, une image comporte forcément des plages uniformes plus ou moins grandes, composées de pixels identiques que l'on peut coder de manière compacte. Dans le cas

d'une séquence vidéo animée, la compression peut exploiter le fait qu'il existe très souvent une grande similitude entre plusieurs images successives. La plupart du temps, la vitesse des mouvements est largement inférieure à la fréquence de rafraîchissement des images. Là aussi, une économie de données peut être réalisée. Par ailleurs, il faut savoir que le système visuel humain n'exploite pas la totalité des informations présentes sur une image. Il est moins sensible aux fins détails de l'image (qu'ils soient fixes ou en mouvement) qu'aux plages uniformes. De plus, il possède un pouvoir de perception bien plus faible dans les détails de couleurs que dans les détails de luminosité. Par conséquent, il existe sur l'image vidéo une grande quantité d'informations auxquelles notre œil n'accorde que peu d'importance et pour lesquelles il peut se contenter d'une représentation approximative.

I.4.1. Compressions sans pertes et avec pertes :

La compression vidéo fait appel à une variété d'algorithmes de codage qui exploitent les différents types de redondance de l'image. Le choix et l'allocation de ces algorithmes se fait en fonction des applications visées et des débits souhaités.

On distingue deux grandes catégories d'algorithmes de compression [38], [03] :

Les algorithmes dits «sans pertes» (lossless en anglais) effectuent un traitement totalement transparent, permettant de retrouver intégralement les données d'origine à l'issue de la décompression. Malheureusement, ils ne conduisent à des taux de compression très faibles, en tout cas insuffisants pour la plupart des applications vidéo.

Les algorithmes dits «avec pertes» (lossy en anglais) aboutissent à des taux de compression nettement supérieurs, mais imposent de négliger certaines informations de l'image, en tenant compte de sa nature et de notre perception visuelle. Si elle se fait dans des proportions limitées, l'élimination de ces informations peut passer inaperçue pour un observateur moyen; on parle alors de compression virtuellement transparente. Si, en revanche, la réduction de débit doit être réalisée dans des facteurs élevés, le prix à payer est l'apparition d'artéfacts et distorsions plus ou moins visibles.

La compression numérique élimine les données redondantes, simplifie ou supprime certaines informations peu importantes pour notre œil, et utilise des systèmes de codage mieux adaptés (plus efficaces).

Les dégradations engendrées par une compression trop poussée sont visuellement différentes des défauts pouvant affecter un signal analogique. Les principales sont les suivantes:

- *effet de bloc:* une structure rappelant une mosaïque carrée apparaît à certains endroits de l'image (du même type que celle que l'on; peut voir en lecture rapide avec une cassette numérique). Ce sont les blocs de pixels (voir plus loin) qui deviennent visibles, à cause de taux de compression excessif ;
- *effet de halo :* une sorte de frange apparaître sur les contours des objets, particulièrement visible sur les textes incrustés, du fait d'un débit trop faible;
- *effet de bluring :* les détails sont moins nets, les contours moins marqués, avec parfois un effet de traînée. Là encore, une compression trop élevée est en cause;
- *bruit de quantification:* un effet de neige ou de vitre sale vient polluer l'image, mais pas uniformément. Ce défaut est généralement lié à un problème local de conversion analogique/ numérique;

- *effet «de moustique» (Mosquito noise) :* du bruit apparaît sur les transitions d'éléments en mouvement, sous la forme de petits points noirs ou blancs qui miroitent autour des objets, rappelant des moustiques. Ce défaut est dû à des erreurs de quantification entre pixels voisins.

I.4.2. Les redondances de l'image vidéo :

I.4.2.1. La redondance spatiale :

Toute plage uniforme sur une image renferme des pixels identiques. Il est donc inutile de coder séparément chacun de ces pixels puisqu'un seul peut les caractériser tous. Il suffit de transmettre deux données, l'une représentant la valeur du pixel,' l'autre étant le facteur de répétition. Nous allons voir comment une technique comme la Transformée en Cosinus Discrète (souvent notée DCT) peut mettre en évidence cette redondance spatiale à l'intérieur de chaque image.

I.4.2.2. La redondance temporelle :

Dans une séquence vidéo, il existe une très forte corrélation entre les images successives. Il suffit en effet de visionner sur un magnétoscope une séquence au ralenti pour constater que très' peu d'éléments changent d'une image à l'autre. Les techniques d'estimation de mouvement permettent de coder une image par rapport à sa voisine, en prévoyant ses changements et ne transmettant que les informations relatives aux éléments qui se sont déplacés. L'élimination des redondances temporelles : peut conduire à des taux de compression très élevés. En revanche; le codage s'applique dans ce cas non plus à des images isolées, mais à, des groupes d'images rendues indissociables, parce que décrites les unes en fonction des autres. Il 'est donc bien adapté à la diffusion d'un flux continu d'informations, mais se prête difficilement au montage, surtout si les groupes d'images sont longs.

I.4.2.3. La redondance subjective :

L'exploitation de la redondance subjective fait appel à la notion de codage perceptuel, tirant parti des faiblesses de la vision humaine. Elle consiste à coder avec un nombre de bits limité les éléments de l'image jugés les moins significatifs pour notre œil. Cette pondération psychovisuelle est obtenue lors de la phase de quantification non linéaire, qui introduit des pertes de résolution

I.4.2.4. La redondance statistique :

Il s'agit d'une notion purement mathématique: si certains codes reviennent plus fréquemment que d'autres, autant leur réserver les mots les plus courts. Cette opération, appelée codage entropique, n'entraîne aucune perte.

La compression en vidéo met en œuvre une variété d'outils de codage s'appuyant sur les trois principaux suivants.

- Il est inutile de répéter un à un les pixels qui sont identiques sur une image.
- Si une image est très semblable à sa voisine, il suffit de ne transmettre que leurs différences.
- Certaines informations peu ou pas pertinentes pour notre système visuel peuvent etre codées plus grossièrement, voire supprimées.

Pour exploiter les différents types de redondance observées en vidéo, les méthodes employées sont :

- redondance spatiale : transformée en Cosinus Discrète (DCT).
- redondance temporelle : estimation de mouvement.
- redondance subjective : quantification.
- redondance statistique : codage entropique.

I.5. Quelques ordres de grandeurs :

Dans le cas d'une image fixe, on peut typiquement supprimer jusqu'à 70 % des informations sans toucher à sa qualité. On dit alors qu'un taux de compression de 3: 1 est totalement transparent. Si l'on tolère quelques pertes, généralement peu décelables par un œil non averti, le taux de compression peut atteindre 10: 1.

Dans le cas d'une séquence vidéo, on peut obtenir des facteurs de compression nettement supérieurs en tenant compte de la forte parenté entre images contiguës. Par exemple, pour les applications de diffusion, il est courant de recourir à des taux de compression compris entre 15: 1 et 40: 1. Le DVD compresse le signal vidéo dans un facteur d'environ 25. En studio cependant, les exigences plus sévères en matière de qualité du signal imposent de limiter le taux de compression à 5: l, et de conserver un accès individuel à chaque image pour le montage.

Quant aux applications multimédia (CD-Rom, transmission sur réseaux,...), elles se contentent d'une qualité d'image inférieure à celle du VHS, obtenue après la suppression de plus de 99 % des informations.

Les débits avant compression :

Pour un grand nombre d'applications, le débit initial du signal vidéo (270 Mbits/s sur 10 bits) est déjà sensiblement allégé avant même qu'intervienne le processus de compression. D'une part, on ne prend en compte que les données concernant la partie visible de l'image, les instants de suppression horizontale et verticale pouvant être remplacés en numérique par .un simple motif. Le débit du signal passe ainsi de .270 à 207 Mbits/s. D'autre Part, on accepte souvent de travailler avec une quantification sur 'T!bits"; au lieu de 10, ce qui abaisse à l66 Mbits/s le débit du signal utile avant compression.

Par ailleurs, le signal vidéo n'est pas toujours traité dans sa structure 4:2:2, c'est-à-dire avec 720 points de luminance et 360 points de chrominance sur chaque ligne. Si l'on s'interdit formellement de toucher à "la luminance, un sous-échantillonnage d'ordre deux est souvent mis en œuvre pour réduire de moitié la résolution de la chrominance, soit dans le sens vertical, soit dans le sens horizontal. Comprenons bien qu'il s'agit ici d'une suppression systématique et aveugle de données de couleur, qui ne tient aucunement compte du contenu de 'l'image, contrairement aux algorithmes de compression dont toute la force est d'être adaptifs. La qualité résultante est logiquement inférieure à celle d'une image 4:2:2, mais elle s'avère somme toute satisfaisante pour un bon nombre d'applications, comme la diffusion, la distribution grand public, ou le reportage d'actualité (en revanche.. les travaux en studio de production et, a fortiori, de postproduction exigent que soit maintenue la structure 4:2:2 sur 8, voire, de préférence, 10 bits). Lorsque ce sous-échantillonnage est effectué en horizontal, on ne compte plus que 180 points de chrominance par ligne. Le signal est alors de structure 4: 1: 1. C'est par exemple le cas des formats d'enregistrement DV /DVCAM en 525/60 et DVCPRO25. Quand le filtrage est réalisé dans le sens vertical, une ligne sur deux se retrouve totalement exempte d'échantillons de chrominance ; en fait, un seul des deux signaux de différence de couleurs est codé en alternance sur chaque ligne, comme en SECAM. Le signal est alors de structure 4:2:0. Ce schéma est mis en œuvre dans les systèmes de

diffusion numérique, le DVD, ainsi que dans les formats d'enregistrement; DV/DVCAM en 625/50. Dans les deux cas, le débit utile du signal vidéo avant compression est abaissé de 25 %, passant de 166 Mbits/s (4:2:2) à 124 Mbits/s (4:2:0 et 4:1:1). Précisons cependant qu'il est vivement déconseillé de combiner entre eux ces équipements utilisant des structures d'échantillonnage différentes. Car si un signal 4: 1: 1 ou 4:2:0 conservera sa qualité lors d'un traitement en 4 : 2 :2, la mise en cascade d'équipements 4 :1 :1 et 4 :2 :0 donnera moins qu'un signal « 4 :1 :0 »…

Pour donner un ordre d'idée, les débits obtenus après compression sont de 4 à 8 Mbits/s en diffusion (4:2:0; 8 bits), 4,5 Mbits/s en moyenne pour le DVD vidéo (il s'agit ici d'un débit variable, avec un maximum de 9,8 Mbits/s), 25 Mbits/s en enregistrement DV (4:2:0 : ou 4: 1 : 1, 8 bits), et 50 ou 100 Mbits/s en production et postproduction broadcast haut de gamme (4:2:2, 8 ou 10 bits).

Il faut aussi savoir que la relation qualité/débit n'est pas linéaire et que, par exemple, une séquence diffusée à 8. Mbits/s n'est pas d'une qualité deux fois supérieure à celle d'une séquence à 4 Mbits/s. Les industriels ont réalisé des tests comparatifs s'appuyant sur des panels d'utilisateurs, qui révèlent en substance que l'amélioration qualitative au delà de 6 Mbits/s n'est perçue que par une minorité de téléspectateurs.

Pour interpréter de manière juste un débit compressé ou un taux de compression, il faut bien connaître les paramètres de signal source. Un signal à 25 Mbits/s peut être obtenu avec un taux de compression de 8 : 1 si le signal source est le 4 :2 :2 sur 10 bits. Mais il peut aussi résulter d'une compression de facteur seulement 5 :1, si le signal de départ est le 4 :1 :1 ou le 4 :2 :0 sur 8 bits (c'est le cas du DV).

I.6. Les normes de compression vidéo (historique) :

En un peu plus de dix ans, quatre standards de compression numérique des images ont été définis par des groupes de travail spécialisés, formés sous les auspices de l'ISO (*International Standard Organization*): JPEG (*Joint Photographics Experts Group*), MPEG-l (*Motion Pictures Experts Group*), MPEG-2 et MPEG-4.

I.6.1. JPEG et M-JPEG :

Apparue en 1989, JPEG est une norme de compression des images fixes, conçue à l'origine pour le monde de l'impression et de la photocomposition. JPEG accepte n'importe quelle définition d'image et exploite les uniformités présentes à l'intérieur de chacune d'elle; le codage est dit « intra-image ». Or, comme les circuits intégrés de codage JPEG étaient disponibles bien avant tout autre circuit de compression, les fabricants d'équipements vidéo se sont rapidement intéressés à ce standard, même s'il ne leur était pas destiné au départ. Partant du principe qu'une séquence vidéo n'est qu'une succession rapide d'images fixes, ils ont développé des systèmes JPEG capables dé compresser/décompresser en temps réel 25 ou 30 images par seconde. Ces solutions, baptisées M-JPEG (Motion-JPEG), se sont vite répandues à partir du début des années 90, notamment dans les stations de montage non-linéaire et dans le domaine de l'enregistrement haut de gamme. Cependant, les modifications ayant permis de passer du JPEG au M-JPEG, ainsi que la synchronisation du son, n'ont jamais fait l'objet d'une normalisation. Les fabricants ont donc développé sans concertation des solutions propriétaires, si bien que les fichiers générés par des équipements de marques différentes sont très souvent incompatibles entre eux.

L'échange des programmes M-JPEG n'est possible qu'à travers un cycle de

compression/décompression du signal. Depuis, est apparue le format d'enregistrement DV, qui utilise une compression de type M-JPEG, mais dont l'algorithme a été totalement normalisé à l'échelle mondiale. Initialement développé pour le grand public avec un débit de 25.Mbits/s, le DV a progressivement évolué vers des débits supérieurs (50 et 100 Mbits/s), donnant naissance à une famille de systèmes de compression adaptés à tous les besoins de la chaîne de production et postproduction broadcast, en définition standard comme en haute définition: DV, DVCAM, DVCPRO25, DVCPRO50 et DVCPRO-HD.

I.6.2. MPEG-1:

Apparue en 1992, MPEG-l est une norme de compression des images animées à faible résolution, destinée aux applications multimédia. Fruit de quatre années de travaux au sein du comité MPEG, le MPEG-1 a été conçu pour coder un programme audio/vidéo dans une qualité tout juste comparable à du VHS. Son débit maximal a en effet volontairement été bridé à 1,5 Mbits/s (dont 1,15 Mbits/s pour la vidéo), afin d'être compatible avec le stockage sur un Compact Disc (CD-Rom, CD-Vidéo), qui peut ainsi contenir 74 mn de programme. Pour parvenir à un débit aussi bas, MPEG-l travaille sur une résolution d'image réduite à 1/4 seulement de celle d'une image standard (360 x 288), et ne prend en compte qu'une trame sur deux (l'autre est simplement répétée). Il effectue ensuite un codage intra-image de type JPEG à l'intérieur de chaque image, qu'il complète par un formidable codage inter-image. C'est ce dernier qui, en exploitant les redondances temporelles entre plusieurs images successives (rendues de ce fait indissociables) permet d'obtenir des taux de compression aussi élevés. L'une des applications phare du MPEG-l est le CD-Vidéo, qui na pas véritablement percé en Europe, mais qui a rencontré un grand succès dans d'autres pays comme la Chine ou plusieurs millions de lecteurs ont été vendus. MPEG-l est aujourd'hui notamment utilisé comme format de consultation dans plusieurs applications broadcast, mais il tend à être remplacé par MPEG-4.

I.6.3. MPEG-2 :

Normalisé en 1994, MPEG-2 est devenu le standard de compression de référence pour tous les secteurs de diffusion/distribution audiovisuelles, et a permis à l'industrie grand public de migrer avec succès de la technologie analogique à la technologie digitale. Il n'est pas exagéré de dire que toute image affichée sut un écran de télévision aujourd'hui a été au moins une fois codée en MPEG2 au cours de son cheminement. MPEG- 2 reprend la philosophie et l'ensemble des techniques de base de MPEG-l (compressions spatiale de type JPEG et temporelle), avec des outils optimisés pour le faire sérieusement monter en puissance en termes de débit, donc de qualité d'image. Tout d'abord, la résolution de l'image d'entrée s'étend de la définition standard (720 x 576J à la haute définition (1920 x 1080), avec un balayage entrelacé ou progressif. Le débit vidéo s'échelonne typiquement de 4 à 50.Mbits/s en SDTV, et peut atteindre 300 Mbits/s en HDTV. Du coup, MPEG3, qui devait se consacrer à part entière à la haute définition, a été totalement absorbé par MPEG-2, qui possède tous les outils nécessaires. MPEG-2 a d'abord été étudié pour la diffusion numérique ; il est aujourd'hui mondialement utilisé par tous les opérateurs de télévision numérique et constitue également le standard des DVD- Vidéo. Par la suite, MPEG-2 a été décliné en une version « studio », baptisée MPEG-2 422P@ML (ou MPEG-2 422), répondant aux exigences plus sévères de la production/postproduction broadcast et permettant de travailler en mode intra-image, comme en M-JPEG, pour permettre des manipulations et traitements complexes et précis des images.

I.6.4. MJPEG-4 :

Normalisée en 2000, MPEG-4 se distingue fondamentalement de MPEG-l et MPEG-2 par son approche orientée objet et l'intégration, dans une norme unique, de toutes les composantes liées à l'interactivité. Ce standard est capable de coder individuellement lès différents éléments média d'une image (les « objets»audio, Vidéo, graphiques...) en les faisant bénéficier du traitement le mieux adapté à leur nature. Ces objets sont alors transmis séparément sur tout type de réseau, afin de permettre à l'utilisateur final d'interagir sur la composition de la scène audiovisuelle qu'il reçoit. En particulier, le traitement de la vidéo, qui a jusqu'à présent concentré l'essentiel des développements de la norme, est le plus performant de tous les standards actuels. Il reprend les grands principes déployés par MPEG-2; mais avec des outils plus sophistiqués et des techniques inédites, offrant un codage plus raffiné et aboutissant à une efficacité de compression très sensiblement supérieure. MPEG-4 est un standard extrêmement riche et complexe, pouvant fournir des débits allant de quelques Kbits/s à plusieurs centaines de Mbits/s.

Son champ d'application est, on l'imagine, très vaste, allant du streaming sur Internet à l'enregistrement de la vidéo à haute définition, en passant par le multimédia mobile.

JPEG	Initialement codage d'images fixes. A été décliné pour la vidéo, considéré comme une succession d'images fixes, en plusieurs versions dites M-JPEG (non normalisé) et DV (normalisé). Débit types de 25 à 100 Mbits/s.
MPEG-1	Codage vidéo en qualité VHS pour le CD-Rom. Débits de 1.15 Mbits/s.
MPEG-2	Codage vidéo en qualité broadcast standard et haute définition. Débits de 4 à 300 Mbits/s.
MPEG-4	Codage orienté objet appliqué individuellement à chaque élément média d'une scène audiovisuelle (audio, vidéo, 2D, 3D). Débits de quelques Kbits/s à plusieurs centaines de Mbits/s.

Tableau 2 : différents standards de compression.

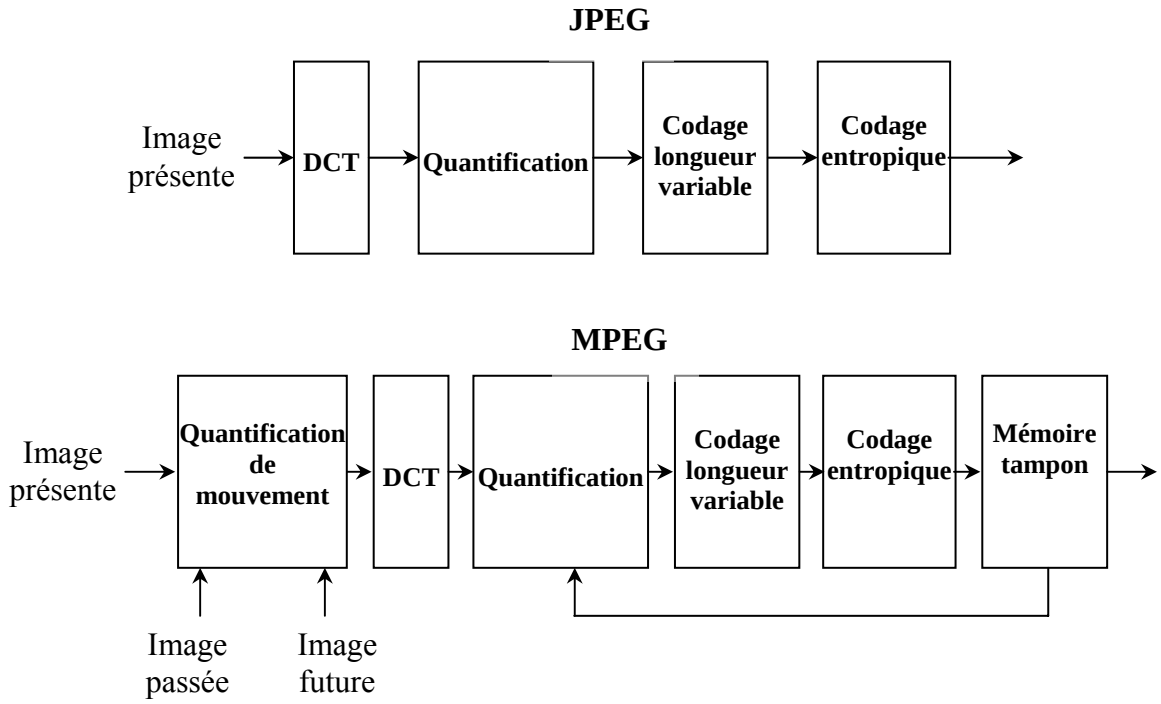


Figure I.2 : principe des compressions JPEG et MPEG.

La compression JPEG (et sa déclinaison vidéo M-JPEG) traite indépendamment chaque image en éliminant ses redondances spatiales : le codage est dit « intra-image ».

Les compressions MPEG-1 et MPEG-2 traitent les images par groupes, en exploitant leurs redondances temporelles, c'est-à-dire les données communes à plusieurs images successives : le codage est dit « inter-image ». MPEG-2 peut cependant travailler uniquement en mode intra-image pour les applications de studio. MPEG-4 reprend les principes de MPEG-1 et 2, mais avec des outils plus performants adaptés à tous les débits, et avec une approches orientée objet.

I.7. MPEG-1 : La compression vidéo pour le multimédia :

MPEG-l est un standard de compression de séquences vidéo animées, associées à un son stéréo synchrone. MPEG-l fournit un débit d'environ 1,5 Mbits/s (dont 1,15 Mbits/s, pour l'image), totalement calibré pour le CD- Rom. Avec une capacité de 650 Mo, celui-ci offre ainsi une capacité de 74 minutes de programme codé en MPEG-I. Pour parvenir à un taux de compression aussi 'élevé, MPEG-I reprend les techniques de JPEG pour ce qui est de l'élimination des redondances internes à chaque image, mais fait également appel à d'autres procédés, dont certains très radicaux. Tout d'abord, il travaille à partir d'une image source au format SIP (*Source Intermediate Format*), caractérisée par une réduction de moitié des résolutions spatiale et temporelle. La taille de l'image est d' 1/4 d'écran et l'affichage s'effectue à 25 trames par seconde en balayage progressif. D'autre part, MPEG-l cherche à tirer pleinement parti des fortes similitudes entre les images. Il se contente de coder 'intégralement une image de temps en temps et d'indiquer, le reste du temps, uniquement quels éléments ont bougé.

Mis à part le choix du format d'entrée, toutes les techniques de compression utilisées par MPEG-l ont été reprises par MPEG-2, avec cependant des paramètres différents garantissant une meilleure qualité.

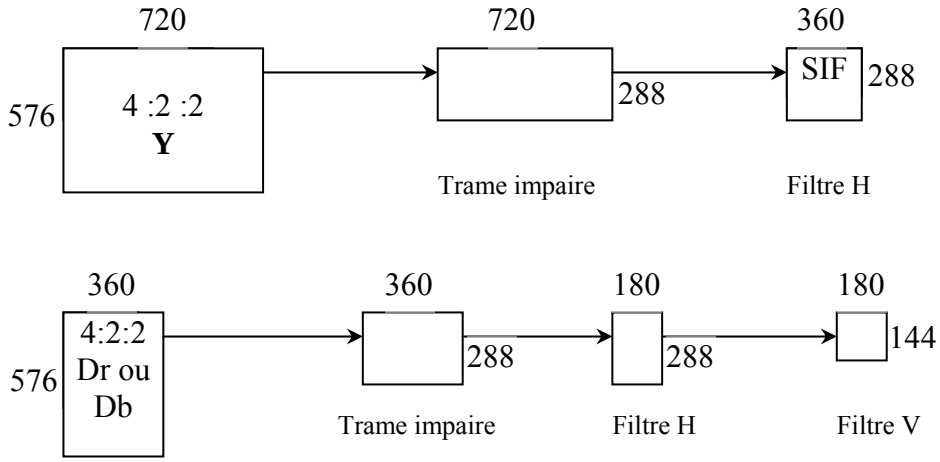
En plus de l'analyse interne des images mise en œuvre dans JPEG, MPEG-1 travaille sur une

image réduite à 25 % de sa taille d'origine, ne traite qu'une trame sur deux, et élimine les redondances temporelles entre plusieurs images successives.

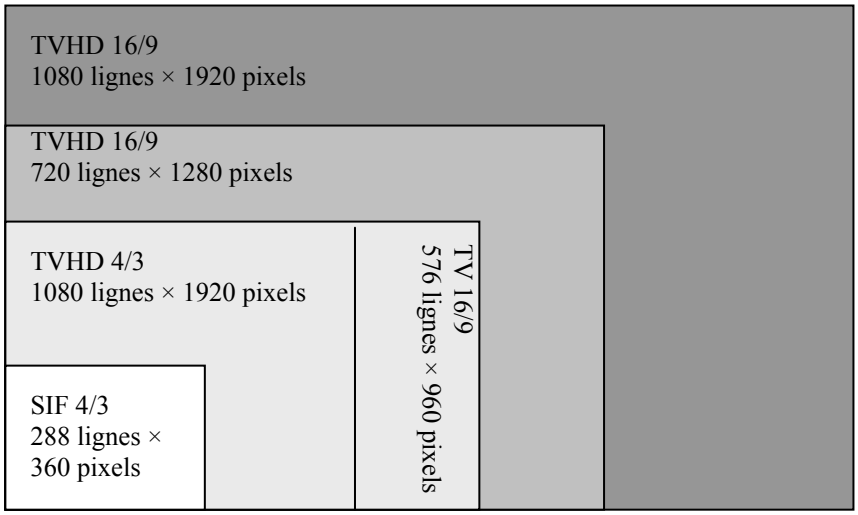
I.7.1. Le format source SIF (Source Intermediate Format) de MPEG-1 :

Les dégradations engendrées par le format SIF, sont déjà très sévères, avant même qu'intervienne la compression MPEG-l.

Tout d'abord, il ne sait pas gérer le balayage entrelacé et supprime d'emblée une trame sur deux. La trame conservée sur chaque image est affichée deux fois de suite lors du décodage pour que l'œil ait une impression de continuité. D'autre part, le format SIF élimine un point sur deux sur chaque ligne, en luminance comme en chrominance. Il est donc caractérisé par une résolution de 360 pixels x 288 lignes et une fréquence trame de 25 Hz. Le débit du signal SIF avant compression est de 31,5 Mbits/s et son niveau de codage, en langage numérique, est le 2: 1 :0. Le sacrifice est certes drastique, mais indispensable vers l'objectif visé: Gardons tout de même à l'esprit que, l'on ne cherche pas à faire mieux que du VHS.



Conversion du format 4 :2 :2 vers le format SIF



Rapport de grandeurs entre les différents formats d'images

Figure I.3 : le format source SIF utilisé par MPEG-1 comparé aux autres formats TV.

I.7.2. Les « GOP » en MPEG :

Après avoir réduit les redondances spatiales à l'intérieur de chaque image au moyen de l'algorithme JPEG, la compression MPEG se poursuit en s'appliquant à des groupes d'images appelés GOP (Group Of Pictures): Les GOP se composent d'une combinaison de trois types d'images:

- l'image I (Intra) : elle est codée en mode intra-image en JPEG. Elle est entièrement décrite par elle-même, sans aucune référence à d'autres, et contient tous les éléments nécessaires à sa reconstruction. Elle constitue de ce fait le point d'accès pour le décodage. La fréquence d'occurrence des images I dans une séquence MPEG conditionne la précision de l'accès aléatoire. Les images I bénéficient cependant d'un faible taux de compression et sont donc assez volumineuses;
- l'image, P (Prédite) : elle est prédite à partir d'une image passée I ou P. Elle est codée uniquement à l'aide de vecteurs mouvement indiquant les déplacements de ses éléments par rapport à l'image de référence. Une image P est typiquement trois fois moins volumineuse qu'une image I, mais peut transmettre des erreurs car elle sert également de référence. Il faut attendre l'arrivée d'une image I pour tout remettre à plat et relancer un nouveau processus de prédiction;
- l'image B (Bidirectionnelle) : elle est construite, à l'aide de vecteurs mouvement, par interpolation bidirectionnelle entre les images passées ou futures I ou P voisines. Elle offre le taux de compression le plus élevé, mais ne propage pas d'erreur, car elle n'est jamais utilisée en référence. Une image B est typiquement six fois moins volumineuse qu'une image I. C'est grâce aux images B que l'on peut faire chuter le débit d'un flux MPEG.

Un GOP commence toujours par une image I et se termine à la dernière image précédant la prochaine image I. Il peut se résumer à une seule image I, ou alors être composé d'une combinaison d'images I et P, I et B, ou encore I, B et P.

La figure I.4 donne l'exemple d'un GOP contenant les trois types d'images I, P, B. On y relève deux paramètres fondamentaux: l'intervalle M séparant deux images Prédites et l'intervalle N entre deux images Intra (c'est la longueur du GOP). Les valeurs de M et N normalisées pour la diffusion sont : M = 3 et N = 12. Dans cette configuration, seule une image sur douze est transmise intégralement, soit deux par seconde; toutes les autres se réfèrent à leurs voisines.

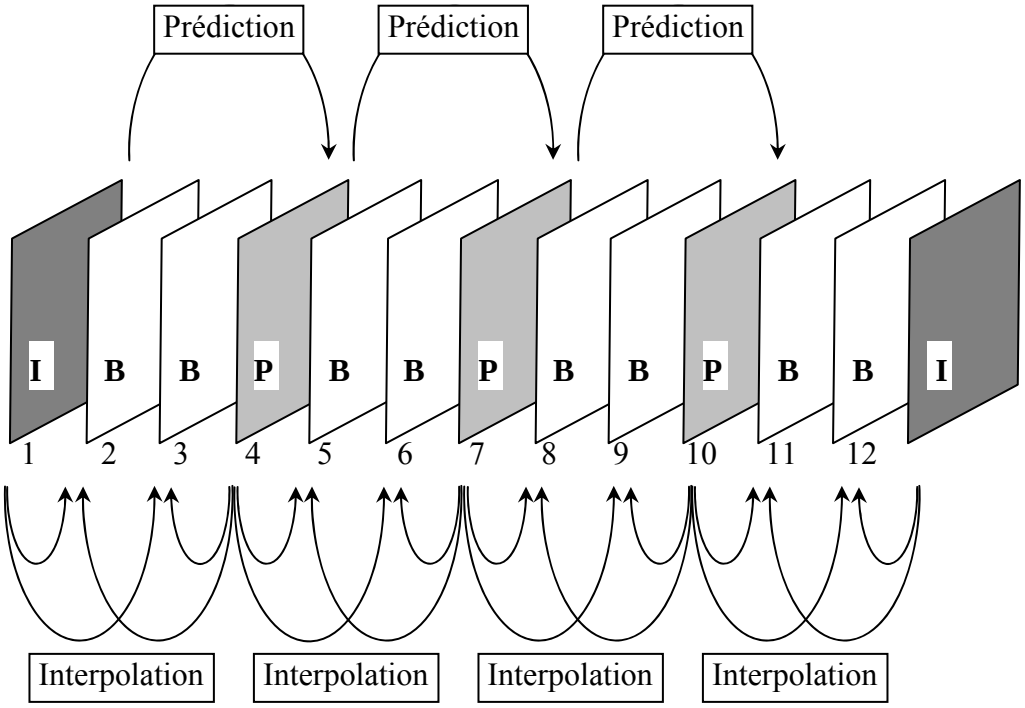


Figure I.4 : organisation type d'un GOP de 12 images pour la diffusion. M=3, N=12**.**

Le nombre d'images Prédites séparant deux images Intra est ici assez élevé; ce qui implique de soigner le processus d'estimation de mouvement. La séquence type avec M = 3 et N=12 est la suivante :

I B B P B B P B B P B B I B B P B B...
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17...

Notons que pour qu'une image B puisse être calculée, il faut que les images I et P dont elle dépend aient préalablement été reçues par le décodeur et conservées temporairement en mémoire. D'où la nécessité de modifier l'ordre de transmission des images par rapport a leur ordre naturel d'analyse. De ce fait, un retard de l'ordre de la durée du Gap est toujours introduit lors de la décompression. Cela donne, en reprenant notre exemple :

I P B B P B B P B B I B B P B B P B...

0 3 1 2 6 4 5 9 7 8 12 10 11 15 13 14 17 16...

L'inconvénient majeur découlant d'un tel système de codage par longs groupes d'images est qu'il ne permet pas un accès direct à chaque image. On ne peut pas briser un GOP, car cela aurait pour conséquence de désolidariser une image P ou B de celle(s) servant à leur construction. Comme seul l'accès aux images I est autorisé, l'unité d'accès aléatoire vidéo en MPEG est le GOP. Il est par conséquent impossible - ou disons qu'il serait extrêmement complexe - d'effectuer du montage sur un programme ainsi codé, à moins de s'imposer des points d'entrée/sortie correspondant uniquement aux images I, soit, dans le cas présent, deux par seconde. En revanche, un GOP long présente l'avantage de permettre un taux de compression plus élevé qu'un GOP court pour une même qualité d'image. Autrement dit, à débit égal, un GOP long donne une meilleure qualité d'image qu'un GOP court.

Dans les applications de diffusion et de distribution sur DVD, le GOP est respectivement de 12 et 15 images. En enregistrement broadcast cependant, il ne dépasse pas 2 images (format Betacam SX de Sony) et se réduit même, dans tous les autres cas, à une seule image.

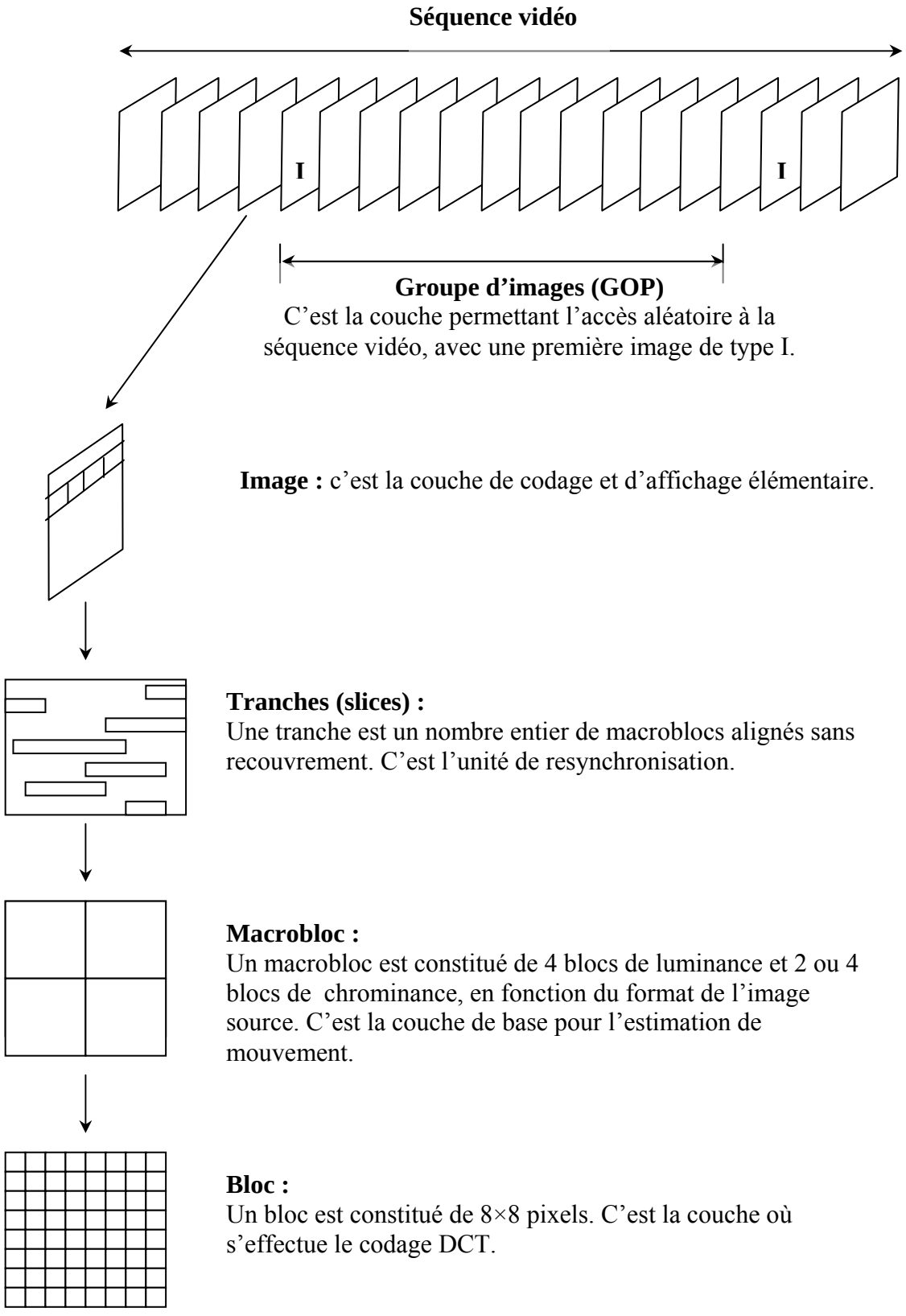
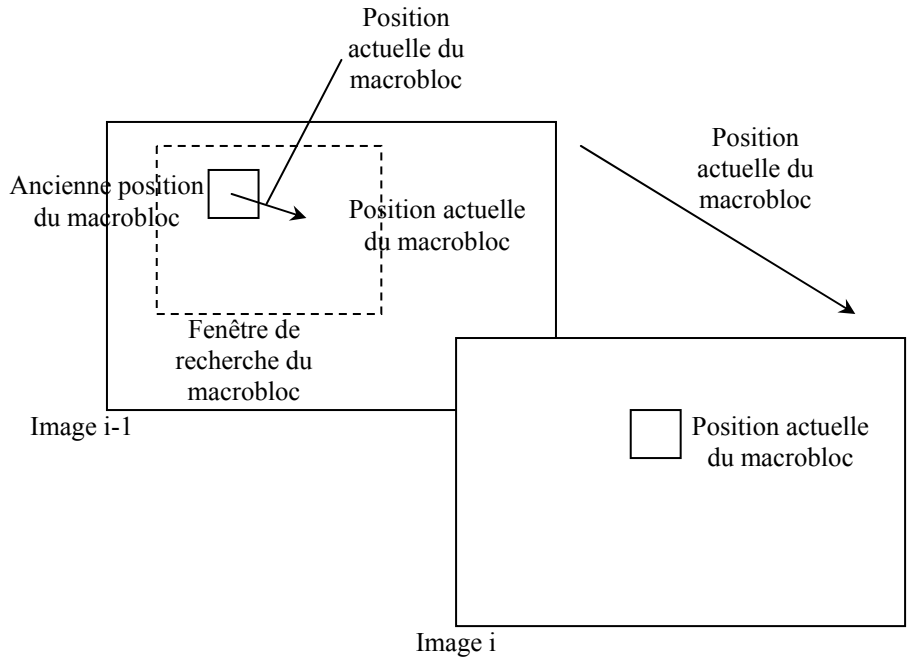


Figure I.5 : structure d'une séquence vidéo MPEG

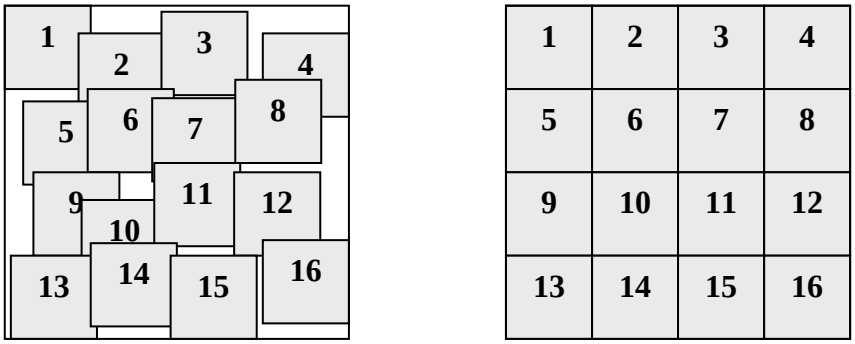
I.7.3. L'estimation de mouvement :

Le principe de l'estimation de mouvement consiste à construire une image de prédiction à partir d'une image précédente et d'informations relatives aux déplacements subis par ses composantes. L'estimation de mouvement n'est pas effectuée sur chaque point de l'image, ni même sur les blocs 8 x 8 de la DCT, mais sur des macroblocs, c'est afin de réduire au minimum la richesse d'informations à transmettre. Un macrobloc est typiquement constitué de l'association de six blocs : quatre pour la luminance et deux pour la chrominance. Si, entre une image actuelle et celle qui la précède, deux macroblocs semblables sont repérés, mais à des emplacements légèrement différents; il suffit de transmettre une seule fois ce macrobloc et d'indiquer quelle est sa nouvelle position sur l'image actuelle. Cette information de position - amplitude et direction - est donnée, par un vecteur mouvement.

La technique la plus répandue pour former le vecteur mouvement est celle du 'block matching' [03], que l'on traduit par correspondance des blocs. L'estimateur de mouvement compare l'image d'entrée, que nous considèrerons comme la nouvelle image « i », avec l'image précédente « i-l», conservée en mémoire. Cette comparaison consiste à examiner un à un les macroblocs de l'image «i », afin de voir si ils existaient sur l'image « i-l ». La figure I.6 illustre ce principe avec un macrobloc sur l'image « i », que l'on cherche à localiser sur l'image «i-l ». Une exploration est alors réalisée dans toutes les directions possibles à l'intérieur d'une fenêtre de recherche, afin d'identifier le macrobloc qui lui ressemble le plus. Lorsqu'il est repéré, la différence de position spatiale du macrobloc entre les deux images permet de déterminer les coordonnées du vecteur mouvement. Son amplitude représente la vitesse du déplacement; sa direction indique celle de la translation. Les macroblocs occupant la même place sur les deux images sont ignorés, ce qui diminue la quantité d'information à coder.



(a)



Macroblocs de l'image précédente i-1 utilisée pour prévoir la nouvelle image i

Image i prédite, après application des vecteurs mouvement sur les macroblocs de l'image i-1

(b)

Figure I. 6: (a) détection et **(b)** estimation de mouvement en MPEG.

Toutes les correspondances étroites de macroblocs sont combinées pour générer une image prédite, exactement comme le ferait le décodeur. Sauf qu'ici, cette image prédite est comparée avec la « vraie» image « i », afin de produire des données de différence visant à compenser les erreurs ou imprécisions de l'estimateur de mouvement. Les vecteurs mouvement et les données de différence sont transmis avec l'image « i-l ». Ils signaleront au décodeur comment il devra déplacer les macroblocs de cette image « i-l » pour construire l'image « i ». L'estimation de mouvement repose sur un concept d'apparence simple; c'est pourtant la phase la plus complexe du codage MPEG.

L'estimation de mouvement se décompose en cinq étapes [03] :

1. recherche des macroblocs semblables entre une image i et l'image précédente i-1.
2. calcul des vecteurs mouvement caractérisant les déplacements des macroblocs.
3. construction d'une image prédite en utilisant ces vecteurs de mouvements.
4. comparaison de cette image prédite avec la vraie nouvelle image pour générer des données d'erreurs de prédiction.
5. codage et transmission des vecteurs et des données d'erreurs de prédiction.

I.7.4. La régulation du débit :

Le volume des données issues du processus de compression varie fortement en fonction du contenu des images. Or, un débit constant est impérativement requis lorsque le signal numérique compressé doit être enregistré sur un magnétoscope. Celui-ci utilise en effet des éléments mécaniques (scanner, moteurs, etc.) tournant à vitesse régulière (les enregistreurs sur disques s'accommodent, quant à eux, d'un débit variable). Pour maintenir à une valeur fixe le débit du signal compressé, le codeur intègre une boucle de régulation utilisant une mémoire tampon qui agit sur les tables de quantification, comme le montre la figure I.7. Lorsque la mémoire tampon est proche de la saturation, un signal d'alerte est envoyé au quantificateur pour qu'il réduise la précision des coefficients dans le but d'abaisser le débit binaire instantané. À l'inverse, si la mémoire tampon est proche du niveau minimal de fonctionnement, le quantificateur pourra

augmenter la précision des coefficients.

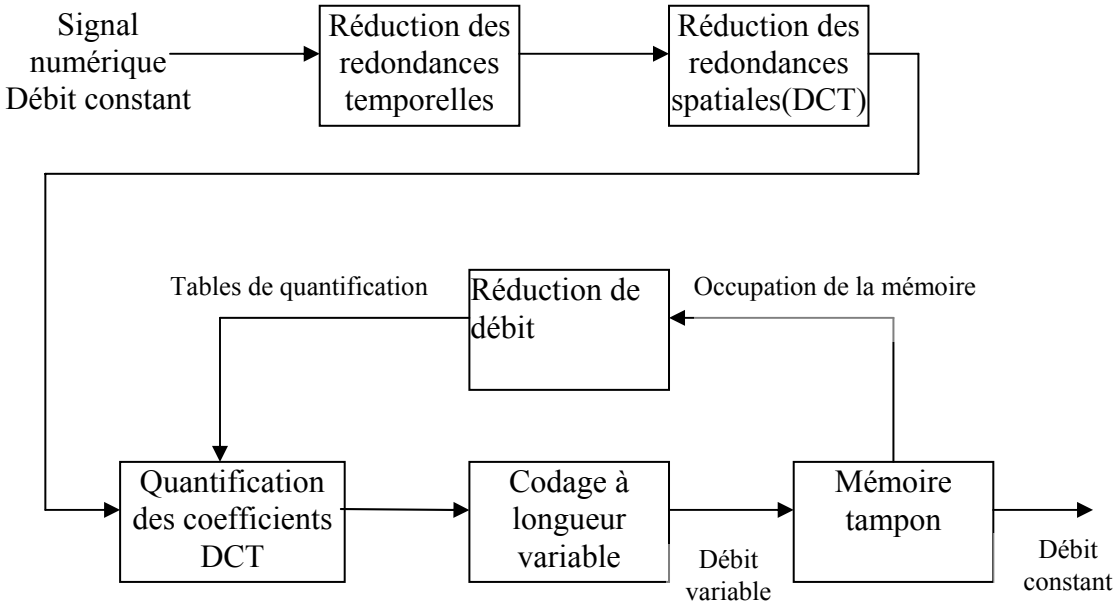


Figure I. 7 : principe de la boucle de régulation

I.7.5. Le codeur MPEG :

L'ordre des images est modifié par l'intermédiaire de mémoires de trame. Pour corriger les erreurs amenées par le calcul des vecteurs mouvement, une boucle assez complexe est introduite : les opérations de DCT et de quantification sont appliquées sur les images Intra - avec la boucle de régulation de débit ; puis les fonctions inverses sont réalisées (quantification inverse, DCT inverse, introduction des vecteurs mouvement) pour que l'image prédite par l'estimateur soit comparée avec la vraie nouvelle image, macrobloc par macrobloc. Une image de référence est alors produite, qui permettra au décodeur de rectifier les erreurs de prédiction éventuelles.

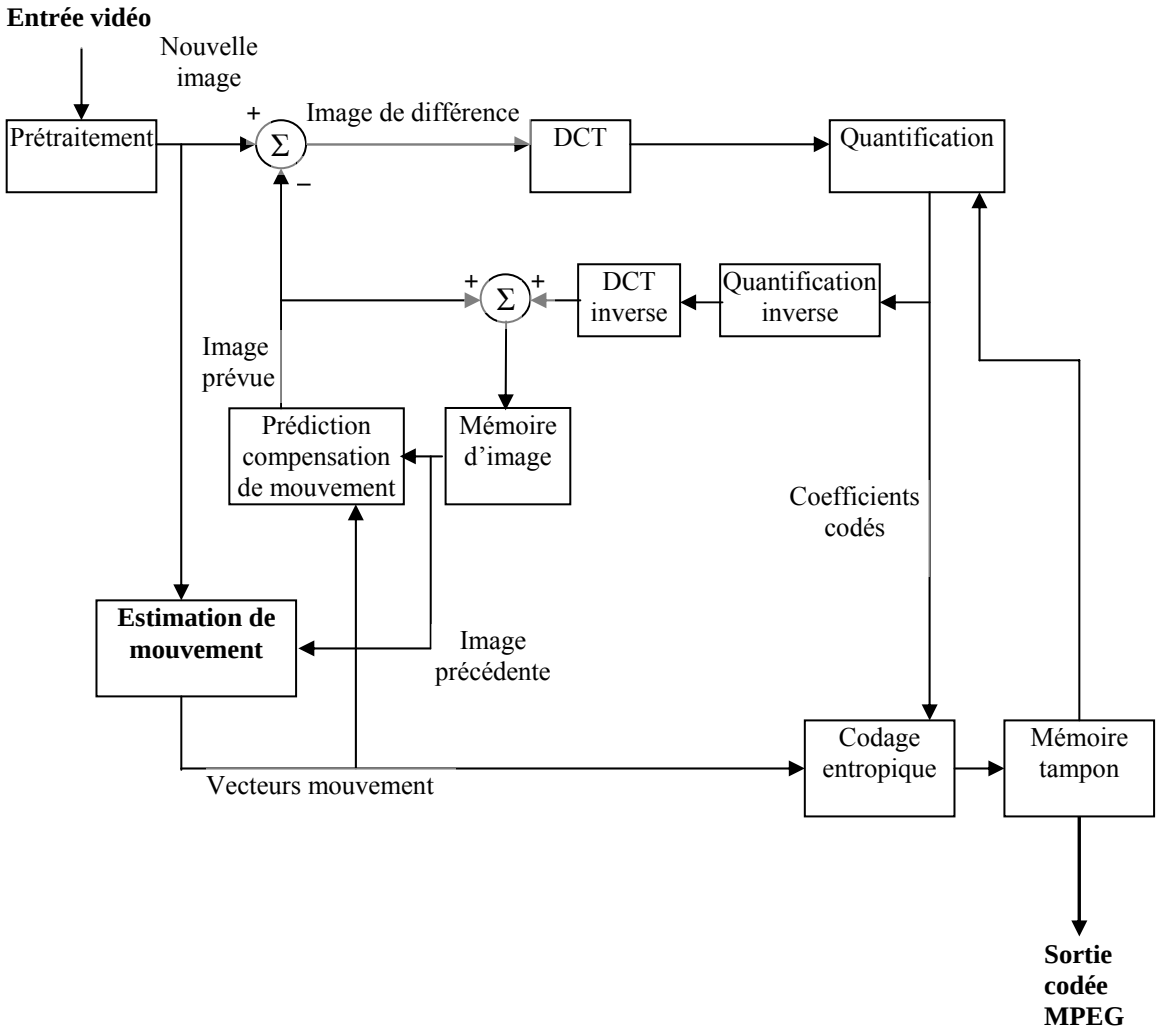


Figure I. 8 : synoptique du codeur MPEG.

I.7.6. Le décodeur MPEG :

Si le codeur est particulièrement complexe et onéreux, le décodeur est en revanche considérablement plus simple (il n'y a pas d'estimation de mouvement à faire) et relativement bon marché, ce qui est indispensable pour sa production à grande échelle sur le marché grand public. Après démultiplexage, les images Intra sont décodées - décodage à longueur variable, quantification inverse avec des modes programmés, DCT inverse. Les vecteurs de mouvement et les données de différence sont utilisés pour fabriquer les images Prédites à partir des images de référence. Les images Bidirectionnelles sont alors calculées, puis les images sont replacées dans leur ordre naturel.

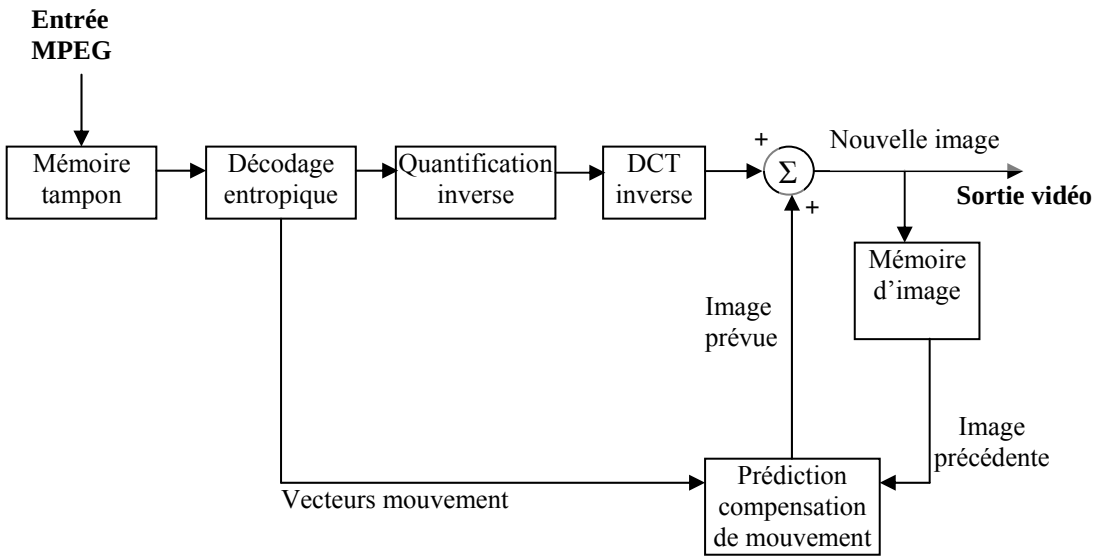


Figure I. 9 : synoptique du décodeur MPEG.

I.8. MPEG-2 : La compression vidéo Broadcast :

MPEG-2 est un standard générique de codage audio/vidéo, unique à l'échelle internationale et indépendant des applications et des supports de stockage ou de transmission. Il s'agit en réalité d'une famille de standards, qui reprend les techniques de base de MPEG-I, mais en les adaptant aux exigences de l'industrie audiovisuelle broadcast, notamment en termes de résolution spatiale et temporelle. À l'origine, MPEG-2 a été conçu pour couvrir toutes les applications de distribution d'images et de sons par satellite, câble et voie terrestre, ainsi que sur support enregistré de type DVD. Par la suite, la norme a été étendue pour prendre en compte les particularités des applications broadcast en production et post-production - montage à l'image, effets spéciaux, incrustations chromakey, multigénération, liaisons de contribution, etc. Précisons cependant que la norme MPEG-2 se contente de décrire les outils de compression des données, ainsi que la syntaxe du signal de transmission. La technique mise en œuvre dans le codeur pour fabriquer ce signal est laissée au soin du constructeur et peut constamment évoluer. La qualité de l'image dépend donc non seulement du débit et de la nature de l'image d'origine, mais aussi de la qualité du codeur et du décodeur.

Par rapport à MPEG-1, les nouveautés introduites par MPEG-2 sont les suivantes:

- support de formats d'image d'entrée supérieurs, jusqu'à la haute définition ;
- traitement du balayage entrelacé (50 trames/seconde) et du balayage progressif (50 images complètes/seconde Le support de l'entrelacé n'est pas sans apporter des complications, notamment, en ce qui concerne les vecteurs mouvement. Ces derniers peuvent être déterminés sur une trame ou sur une image, une décision étant prise indépendamment pour chaque macrobloc en fonction de l'importance des mouvements;
- codage hiérarchique permettant de transmettre différents niveaux de qualité pour une même image, avec compatibilité descendante entre les niveaux;
- compatibilité avec MPEG-1 (un décodeur MPEG-2 doit pouvoir décoder un signal MPEG-1).

MPEG-2 est une famille de standards adoptée à l'échelle mondiale pour la distribution

d'images et de sons numériques grand public, mais aussi pour la production broadcast, qu'elle soit standard ou à haute définition.

I.9. Conclusion :

Dans la compression numérique MPEG l'estimation de mouvement joue un rôle très important, plutôt un rôle principal, ainsi que pour tout les normes de compression, MPEG-2, MPEG-4, H.261, et H.263.

On va étudier l'estimation de mouvement dans le chapitre suivant et les techniques (ou bien les méthodes) les plus utilisées pour cette étape, incluent dans la compression vidéo.

Chapitre II :

Les Techniques d'Estimation de Mouvement

Chapitre II :

Les Techniques d'Estimation de Mouvement

II.1. Introduction :

La détection et l'estimation du mouvement dans des séquences temporelles d'images bidimensionnelles (2D) et tridimensionnelles (3D) est un des problèmes fondamentaux en traitement et analyse d'images. Ce domaine de recherche récent est à l'origine de nombreux problèmes ouverts pour lesquels il n'existe aucune solution satisfaisante et générale.

L'œil humain perçoit et traite en continu le flux de données contenu dans une séquence temporelle d'images. De nombreuses études sur le mouvement ont mis en évidence l'existence de structures spécialisées dans la détection du mouvement dans les yeux de chats et de grenouilles. Il semble que de telles structures existent aussi chez l'homme.

On distingue chez l'homme deux types de vision pour percevoir le mouvement:

- Une vision focalisée, au centre de l'image, qui analyse et estime le mouvement;
- Une vision périphérique, qui détecte seulement le mouvement dans le reste du champ visuel.

La capacité à discerner les objets et en particulier leur mouvement, et à naviguer dans l'espace tridimensionnel 3D est une caractéristique presque universelle dans le monde animal. L'intégration d'un tel système de vision dans une machine ou un robot est un problème non résolu. Même s'il est relativement facile d'assembler un système de vision artificielle, il s'est avéré très difficile d'obtenir une capacité visuelle sur une machine, même avec un degré très limité et une variété très large de détecteurs de mouvement et de systèmes visuels.

On note deux grands groupes de chercheurs en vision. Un premier groupe étudie la vision humaine et animale pour comprendre le fonctionnement du système biologique, ses limitations et sa diversité. Les chercheurs qui composent ce groupe sont des neurophysiologistes, physiciens et psycho-physiciens, qui ont prouve l'existence de certaines cellules corticales dédiées à la détection et à l'estimation du mouvement [01]. Le deuxième groupe inclut des spécialistes en informatique et des ingénieurs qui étudient la vision artificielle, afin de développer des systèmes de vision capables de reconnaître et poursuivre des objets, estimer leur vitesse et leur trajectoire. Les connaissances et les résultats obtenus par les neurophysiologistes, les physiciens et les psycho-physiciens aident à la conception des systèmes de vision artificielle faits par les ingénieurs et les spécialistes en informatique. Dans le même temps, les résultats issus de la vision artificielle offrent un cadre conceptuel et technique pour la modélisation de la vision biologique. Les résultats obtenus en combinant systèmes d'acquisition d'images et systèmes avancés de traitement de l'information en sont un exemple (réseaux neuronaux). La confrontation entre les concepts développés par la vision artificielle et une meilleure connaissance des mécanismes de la vision biologique sont source de progrès mutuels.

Les sources potentielles d'images perçues ou acquises puis traitées par les systèmes visuels biologiques et artificiels, sont nombreuses et les applications variées. À titre d'exemple nous mentionnons:

- Les images aériennes avec la détection et le suivi des masses nuageuses ou l'étude de l'évolution des cultures agricoles.
- Les scènes de télévision numérique avec la compression d'images utilisant l'information de mouvement.
- La poursuite de cibles pour les applications militaires ou la régulation du trafic automobile.
- La biologie avec la cinétique des cellules, ou la médecine avec l'analyse du mouvement du coeur.
- La robotique avec l'inspection visuelle d'une scène animée.

Ces exemples mettent en évidence la diversité des applications dans lesquelles l'estimation du mouvement à partir de séquences temporelles d'images est d'une importance majeure.

II.2. Mouvement réel, mouvement apparent et mouvement estimé :

Les images représentent souvent la projection de scènes réelles 3D. C'est pourquoi le mouvement observé (ou mouvement apparent) dans une séquence temporelle d'images représente le plus souvent la projection du mouvement 3D dans le plan image. Une image peut également représenter une coupe d'un objet 3D. Dans ce cas, le champ de mouvement observe à l'intérieur de cette coupe pourra représenter à la fois un déplacement dans le plan image et également la projection sur ce plan d'un mouvement 3D. Le but de l'estimation de mouvement est d'estimer le champ de mouvement 2D ou 3D à partir d'une séquence temporelle d'images 2D ou 3D dont le contenu varie en fonction du temps. On doit différencier:

- Le mouvement réel;
- Le mouvement apparent (ou observé);
- Le mouvement estimé.

II.2.1. Champ de mouvement réel et champ de mouvement apparent :

Le mouvement (ou déplacement) réel anime la scène réelle, dans l'espace réel 3D. Ce mouvement réel est observé soit par l'œil humain soit par un système de prise de vue. Le second fournit un enregistrement du mouvement observé ou apparent sous la forme d'une séquence temporelle d'images. Souvent, Le champ de mouvement réel n'est pas directement mesurable dans une séquence d'images 2D ou 3D si l'on s'appuie seulement sur l'intensité des pixels. Dans ce cas, on observe en fait des changements de la distribution spatiale d'intensité lumineuse. Le mouvement ainsi perçu est appelé champ de "mouvement apparent" ou "flux optique" qui, en général, est différent du champ réel de mouvement.

Le champ de mouvement ou de déplacement apparent représente, en général, la projection (photographique ou projective) du mouvement réel 3D dans le plan image. C'est pourquoi, le champ de mouvement apparent s'appelle aussi "mouvement projeté". Il représente une approximation du mouvement réel (figure II.1).

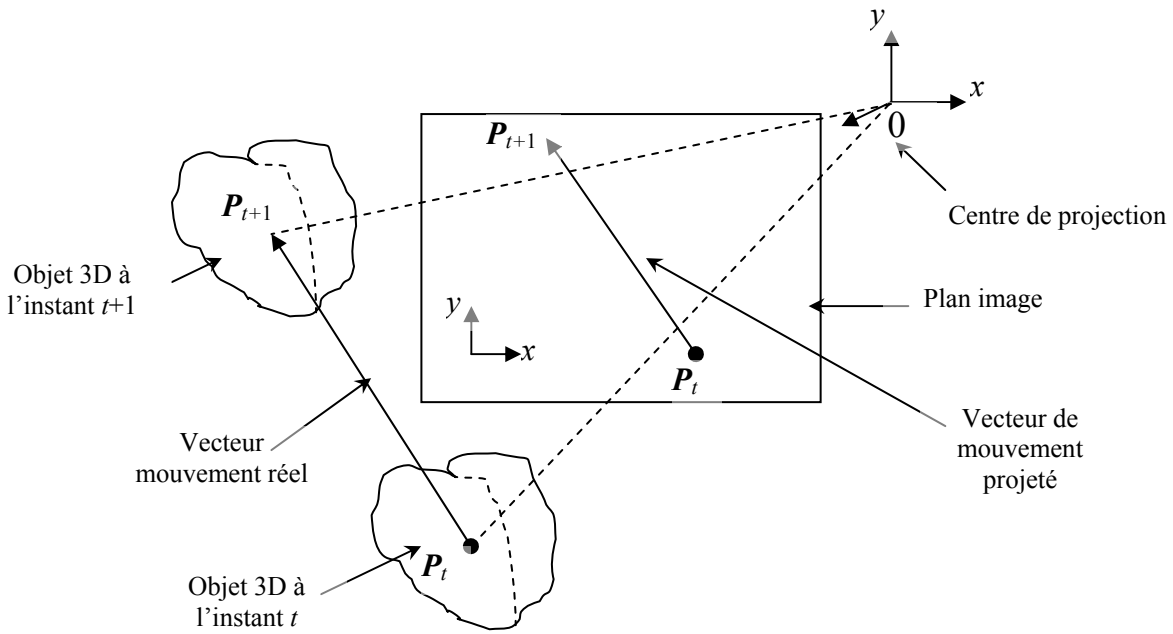


Figure II.1. Illustration des mouvements réel et apparent, dans un système optique de prise de vues.

On suppose que Le point P_t d'un objet réel 3D, à l'instant t, devient Le point P_{t+1} à l'instant t+1. La projection perspective des points P_t et P_{t+1} dans le plan image est notée P_t et P_{t+1}, respectivement avec $I_t(x,y)$ et $I_{t+1}(x,y)$ leur intensité. Sur la figure II.1 on note que le déplacement apparent $\overline{p_t\, p_{t+1}}$ correspond à la projection perspective du mouvement réel $\overline{P_t\, P_{t+1}}$. Tous les vecteurs déplacement dont les extrémités appartiennent aux droites pointillées correspondant à l'opération de projection ont le même vecteur 2D déplacement apparent (ou projeté). L'illustration présentée sur la figure II.1 est valable dans le cas optique, c'est-à-dire pour une séquence d'images acquise avec une caméra vidéo. En imagerie médicale, le problème est plus ou moins compliqué selon le type d'imagerie utilisée. Sur la figure II.2 est présenté un exemple en tomographie X.

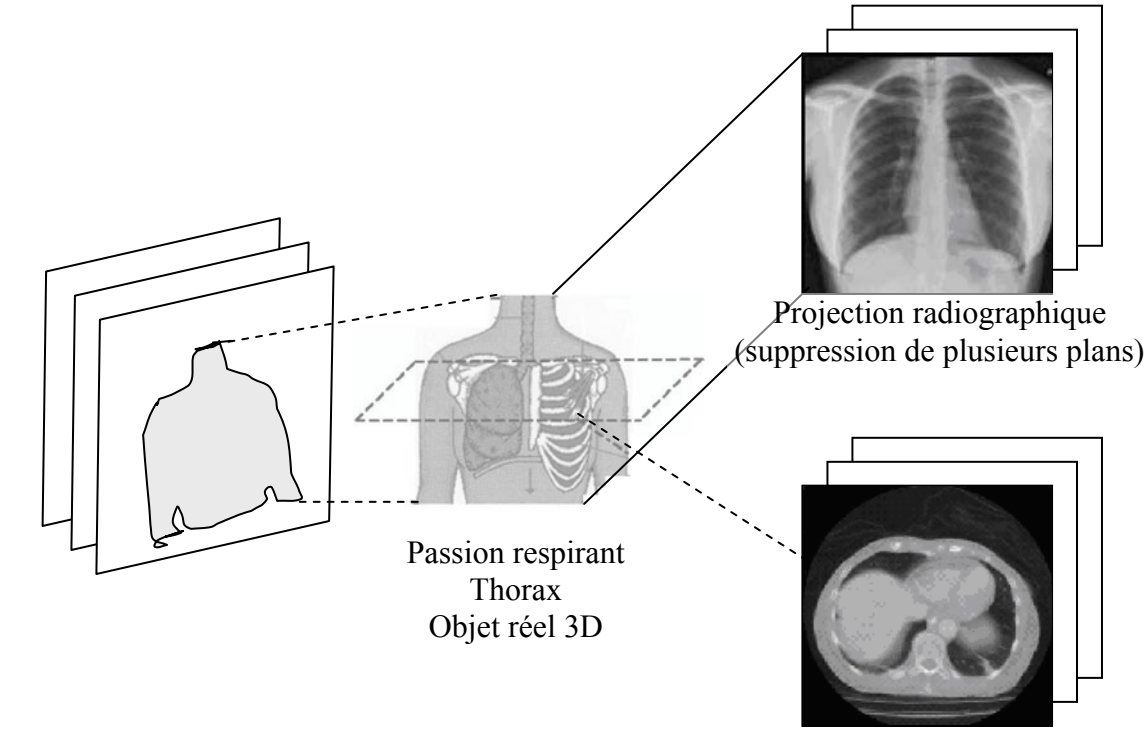


Figure II.2. Illustration comparative d'une acquisition de séquences d'images en optique et en rayons X

L'acquisition par une caméra vidéo d'une séquence d'images d'un thorax en mouvement fournit une projection perspective du thorax. Dans le cas du rayonnement X, on obtient selon le type d'imagerie utilisée: soit une coupe tomographique 2D du thorax, soit une projection radiographique intégrale du thorax. Dans ce cas, le mouvement projeté peut représenter la résultante du mouvement de plusieurs objets. On pourrait par exemple obtenir un mouvement résultant nul, du fait de combinaisons antagonistes de plusieurs organes en mouvement.

Dans le cas continu, le champ de déplacement apparent (ou projeté) est défini pour tous les points $\boldsymbol{p}(x,y,t)$ du plan image par une fonction vectorielle de déplacement, $\boldsymbol{d}(x,y,t)$, à valeur en $\boldsymbol{R}^2$.

Dans le cas discret, le champ de vecteurs déplacement entre les instants t_k et $t_{k+\Delta t}=t_k+k.\Delta t$, où $k \in \boldsymbol{Z}$ et Δt est le pas discret d'échantillonnage temporel, constitue une représentation échantillonnée de la fonction définie dans le cas continu:

$$\boldsymbol{d}(i, j, t_k) = \boldsymbol{d}(x, y, t)\Big|_{[x,\ y,\ t\ ]=[i,\ j,\ t_k\ ]} \qquad \text{(II.1)}$$

En conclusion, Le champ de déplacement représente la totalité des vecteurs déplacement, $\boldsymbol{d}(i, j, t_k)$ où $(i,j,t_k) \in \Lambda^3$, Λ^3 étant un sous-espace discret de l'espace (2D+temps).

II.2.2. Champ de mouvement apparent et champ de mouvement estimé :

Le vecteur déplacement estimé $d(\boldsymbol{p})=(d_x(\boldsymbol{p}),d_y(\boldsymbol{p}))$ correspondant au déplacement du point $\boldsymbol{p}=(x,y)$ du plan image, est déterminé à partir du champ de mouvement apparent, c'est-à-dire des variations locales d'intensité lumineuse $I_t(x,y)$ entre les instants t et $t+\Delta t$, où Δt est la distance en temps inter-images, dans le cas continu le vecteur vitesse estimé $v=(v_x(\boldsymbol{p}), v_y(\boldsymbol{p}))$ est défini comme la variation temporelle du déplacement par unité de temps, $(v_x,v_y)=(d_x /dt,d_y /dt)$. Ceci explique pourquoi une séquence temporelle d'images ne permet que d'estimer le champ de mouvement (déplacement ou vitesse) apparent observable dans la séquence, et non le champ de vitesse réel. On nomme champ de déplacement (respectivement de vitesse), le champ de vecteurs déplacement (respectivement de vitesse) estimé avec une méthode quelconque.

En optique par exemple, la différence entre le champ de vitesse estimé et le mouvement apparent, peut exister si:

- Le gradient spatial d'intensité est trop faible. Pour que le mouvement réel soit observable, il faut que la variation de niveaux de gris (ou des couleurs) soit suffisamment grande dans les régions où il y a mouvement. L' exemple classique qui met en évidence l'importance de ce facteur, est celui d'une sphère ayant une distribution uniforme d'intensité (éclairage homogène), qui effectue un mouvement de rotation autour de son axe propre, dans une scène à éclairage constant. Même si en réalité il y a mouvement, celui ci ne peut être observé à partir d'une séquence temporelle d'images à niveaux de gris, parce qu'aucune variation apparente de niveaux de gris n'intervient pendant le mouvement.
- L'illumination de la scène varie. Un mouvement observable dans une séquence d'images d'intensité ne correspond pas toujours à un mouvement réel. La même sphère immobile soumise à une illumination variable au cours de la séquence génère un mouvement apparent (c'est-à-dire une variation d'intensité) artificiel.

II.3. Estimation de mouvement :

Une des hypothèses qui doit être faite pour estimer le mouvement à partir du champ de mouvement apparent est que l'intensité image reste constante au cours du mouvement ou qu'elle varie d'une manière prédictible d'une image à l'autre [05].

L'hypothèse de conservation de l'intensité lumineuse en chaque point le long de la trajectoire du mouvement peut s'exprimer par l'équation DFD des différences entre les images déplacées (en anglais DFD=Displaced Frame Difference), c'est-à-dire entre les images aux instants t et $t+\Delta t$, avec $\Delta t = \pm 1$:

$$\boldsymbol{DFD}=I(x+d_x y+dy\ t+\Delta t)-I(x,y,t)=0 \qquad \text{(II.2)}$$

où $I(x,y,t)$ est l'intensité du point image $p=(x,y)$ à l'instant t et $d(p)=(d_x(p),d_y(p))$ est le vecteur déplacement correspondant entre les instants t et $t+\Delta t$. On observe que:

a) si les composantes de d ne sont pas entières, le calcul de la DFD en chaque pixel, nécessite une étape d'interpolation;

b) si les composantes de d correspondent à la valeur réelle du déplacement, alors l'erreur d'estimation, donc la DFD, est nulle en chaque pixel.

L'estimation du mouvement réel à partir du mouvement apparent peut être abordée de deux manières différentes:

a) l'estimation des vecteurs déplacement dans Le plan image:

$$d(x,y,t)=(d_x(x,y,t),dy(x,y,t)) \quad \text{(II.3)}$$

estimés entre les images à t et $t+1$;

b) I' estimation des vecteurs vitesse:

$$v(x,y,t)=(v_x(x,y,t),v_y(x,y,t)) \; . \quad \text{(II.4)}$$

Les vecteurs déplacement (respectivement vitesse) estimés peuvent varier en espace et en temps.

a) Estimation des vecteurs déplacement: L'estimation peut être vue comme un problème d'estimation de mouvement direct ou inverse ("avant" ou "arrière"), selon que l'estimation est réalisée entre les instants t et $t+1$ ou entre les instants t et t-1 (figure II.3).

Dans le cas de l'estimation directe (avant) du mouvement, le problème se pose de la manière suivante: si on connaît les échantillons spatio-temporels $I_t\,(x,y)$ et $I_{t+1}(x,y)$, qui sont liés par la relation (hypothèse de conservation de l'intensité):

$$I(x,y)= I_{t+1}(x+d_x(x,y,t),\, y+d_y(x,y,t)), \quad \text{(II.5)}$$

on doit trouver le vecteur de déplacement:

$$d_t(x,y)=(d_{xd}(x,y,t),\, d_{yd}(x,y,t)) \quad \text{(II.6)}$$

Dans le cas de l'estimation inverse (arrière) du mouvement où les vecteurs de déplacement sont définis entre les instants t et t-1, on a la relation:

$$I_t(x,y)= I_{t-1}(x-d_{xi}(x,y,t),\, y-d_{yi}(x,y,t)), \quad \text{(II.7)}$$

En estimation de mouvement on utilise généralement l'estimation inverse (arrière). L'estimation avec compensation directe (avant) du mouvement est classiquement utilisée dans la compression prédictive de séquences d'images. Les valeurs prises par les déplacements d_x, d_y sont souvent réelles, ce qui nécessite une étape d'interpolation pour l'estimation du mouvement.

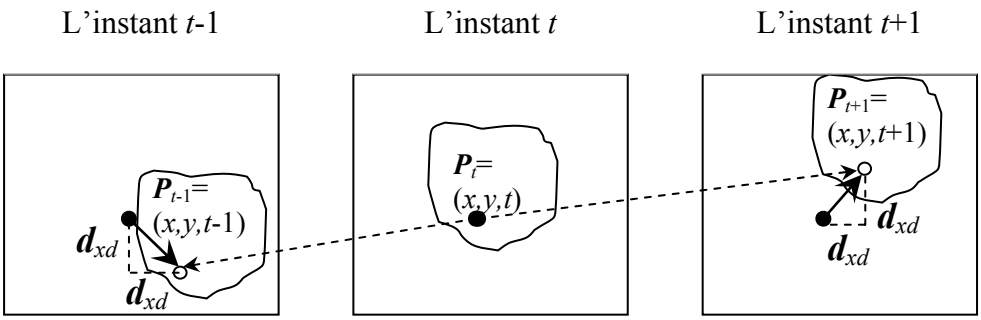


Figure II.3. Estimation directe et inverse des vecteurs déplacement.
Dans Le cas de l'estimation directe on a note: $d_d(x,y,t)=(d_{xd}(x,y,t), d_{yd}(x,y,t))$.
Dans Le cas de I 'estimation inverse on a note: $d_i(x,y,t)=(d_{xi}(x,y,t), d_{yi}(x,y,t))$.

b) Estimation des vecteurs vitesse: étant donnes les échantillons $I_t(x,y)$, on doit déterminer les vecteurs vitesse $v(x,y,t)$. On observe que si la vitesse reste constante dans l'intervalle Δt entre deux images et si Δt est petit, alors la vitesse estimée peut être assimilée au déplacement:

$$\boldsymbol{v}_t(x,y) = \boldsymbol{d}_t(x,y) / \Delta t \qquad \text{(II.8)}$$

Pour un mouvement accéléré, on doit prendre en compte plus de deux images afin d'estimer correctement le champ de vitesse.

En conclusion, le champ de mouvement estimé peut être caractérisé par le champ de vecteurs vitesse ou par le champ de vecteurs déplacement (ou de correspondance). Le champ de vitesse estimé et le champ de déplacement sont identiques quand l'échantillonnage temporel de la séquence est constant. L'estimation du mouvement par la perspective de l'estimation du champ de vecteurs vitesse ou par l'estimation de vecteurs déplacement (ou de correspondance) représente donc deux approches équivalentes, quand on connaît le pas d'échantillonnage temporel de la séquence. Dans la suite on s'intéressera à l'estimation du champ de vecteurs déplacement.

L'estimation de mouvement est très sensible au bruit présent dans les images qui peut être interprète comme étant le résultat d'un mouvement dans la scène réelle.

L'estimation du mouvement est un problème mal-posé qui nécessite l'ajout de contraintes.

Un problème est dit mal-pose s'il n'a pas une solution unique et/ou la solution n'est pas unique:

- Le problème d'existence de la solution peut être illustre dans le cas "d'occultation" ou de mouvement 3D multiples projetés;
- Le problème d'unicité de la solution est lié au problème d'ouverture.

II.3.1. Problème d'occultation :

L'occultation concerne le recouvrement ou non-recouvrement d'une surface (d'habitude le fond d'une image), dû à la translation ou à la rotation d'un objet dans le champ visuel.

Le concept de fond couvert ou découvert est illustré sur la figure II.4. Sur la figure II.4, l'objet représenté en pointillé effectue un mouvement de translation dans la direction x, entre les instants t et $t+1$, correspondants aux images I_t et I_{t+1} de la séquence. La région représentée en pointillée

dans I'image I_t représente La région du fond qui sera couverte par l'objet dans l'image I_{t+1}. Par conséquent, pour les pixels de cette région il ne sera pas possible de déterminer un correspondant dans l'image I_{t+1}.

La région représentée en pointillés dans l'image I_{t+1} représente la région du fond qui sera découverte par l'objet en mouvement. Par conséquent, les pixels de cette région n'auront aucun correspondant dans l'image I_t.

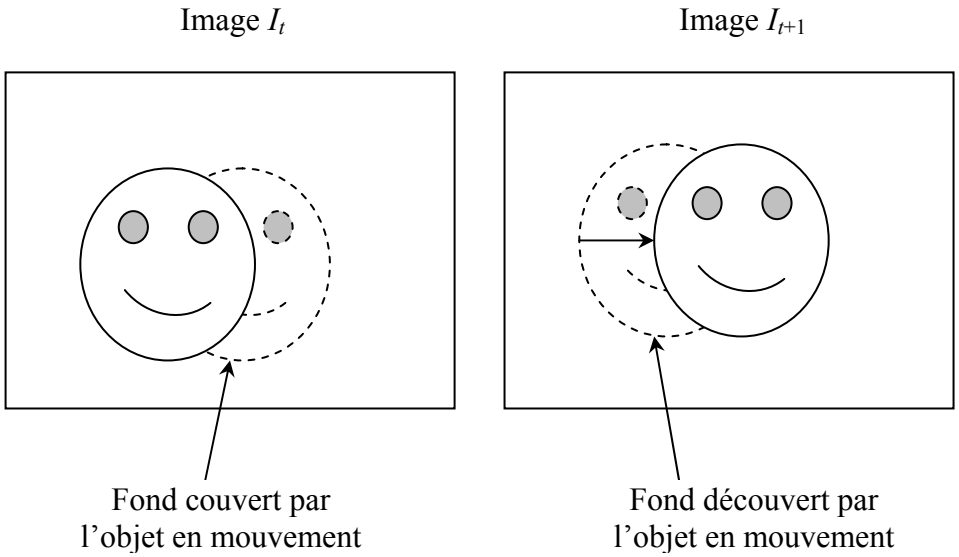


Figure II.4. Illustration du concept de fond couvert et découvert.

En imagerie médicale de projection par rayons X, le problème est plus complexe lorsque les structures 3D sont animées de mouvements varies, leur projection fournit une superposition de champs de vitesse qui rend le problème encore plus sous-déterminé. Dans ce cas il sera nécessaire de faire appel à des méthodes robustes d'estimation du mouvement, avec prise en compte de discontinuités éventuellement combinées à des méthodes de segmentation, capable de dissocier chaque objet et d'en estimer le mouvement séparément.

II.3.2. Problème d'ouverture :

Le problème d'ouverture représente une reformulation du fait que la solution du problème d'estimation du mouvement n'est pas unique. Si on suppose les vecteurs de mouvement en chacun pixel comme des variables indépendantes, alors le nombre d'inconnues sera (dans le cas 2D) deux fois plus grand que le nombre d'équations disponibles. Ceci est dû au fait que le nombre d'équations est égal au nombre de pixels de l'image, mais que le vecteur de mouvement en chaque pixel a deux composantes.

On peut montrer que, en l'absence de contrainte supplémentaire, on ne peut déterminer en chaque point, que la composante normale du déplacement, qui est orientée dans la direction du gradient spatial de l'intensité, au point considéré.

Le problème d'ouverture est illustré sur la figure II.5. On suppose que l'un des coins d'un objet est en mouvement dans la direction verticale haut. Si, pour estimer le mouvement on utilise un bloc (noté B_1, sur la figure II.5), il n'est pas possible de déterminer si l'objet est en mouvement dans la direction verticale supérieure (comme en réalité) ou dans la direction normale au bord de

l'objet.

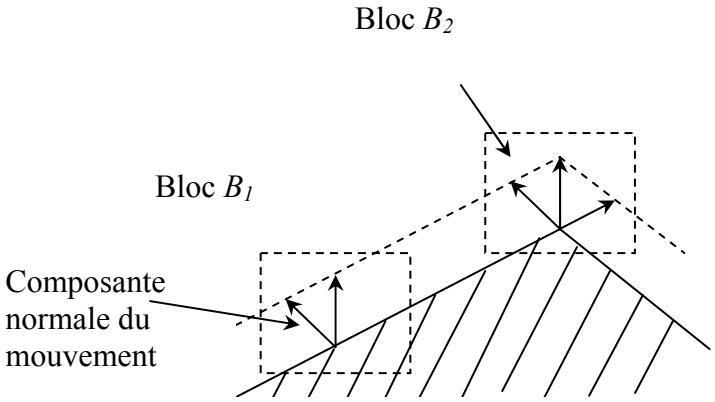


Figure II.5. Illustration du problème d'ouverture.

Si pour estimer le mouvement, on utilise le bloc B_2, on peut déterminer correctement le mouvement, parce que le gradient d'intensité image à dans ce bloc deux composantes, normales entre elles. Par conséquent, pour éviter le problème d'ouverture, en estimation du mouvement on doit utiliser un bloc qui contienne suffisamment d'information traduite par des variations locales de niveaux de gris.

II.3.3. Modèles du champ de vecteurs mouvement :

A cause du caractère sous-determiné de l'estimation de mouvement (problème d'ouverture), il est nécessaire d'introduire des modèles des contraintes supplémentaires concernant le champ de mouvement ou directement des modèles (paramétriques ou non paramétriques) de ce champ.

II.3.3.1. Modèle de translation ;

Le modèle de mouvement le plus utilisé dans les méthodes d'estimation du mouvement est le modèle de mouvement constant ou modèle de translation, qui suppose que tous les pixels du bloc effectuent le même déplacement ***d*** (figure II.6) [27].

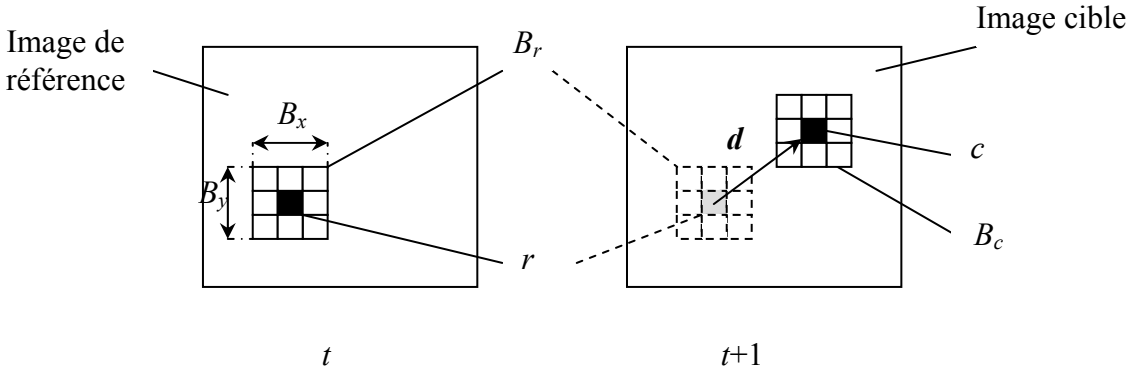


Figure II.6. Modèle de translation du mouvement.

Ainsi, un bloc B_r de pixels de l'image de référence, de dimensions $B_x \times B_y$, centre sur le pixel (***r***)

de coordonnées (*x,y*) dans l'image de référence à l'instant *t*, sera mis en correspondance avec le bloc B_c centre en pixel *c*, dans l'image cible i_t l'instant *t*+ 1, ce qui s' écrit:

$$B_r(x,y,t) = B_c(x + d_x, y + d_y, t + 1) \quad \text{(II.9)}$$

où $d=(d_x,d_y)$ est le vecteur déplacement. Dans ce modèle, les blocs peuvent être disjoints (le cas le plus fréquent en pratique) ou chevauchants (figure II.7). Dans le cas du modèle de translation avec des blocs disjoints, on peut attribuer un seul vecteur au bloc entier. Ainsi, la compensation de mouvement peut être réalisée en copiant l'information de gris ou de couleur, pixel par pixel, de l'image de référence dans l'image cible. Dans le cas du modèle de translation avec blocs chevauchants, on doit estimer le vecteur moyen de mouvement dans la région de superposition.

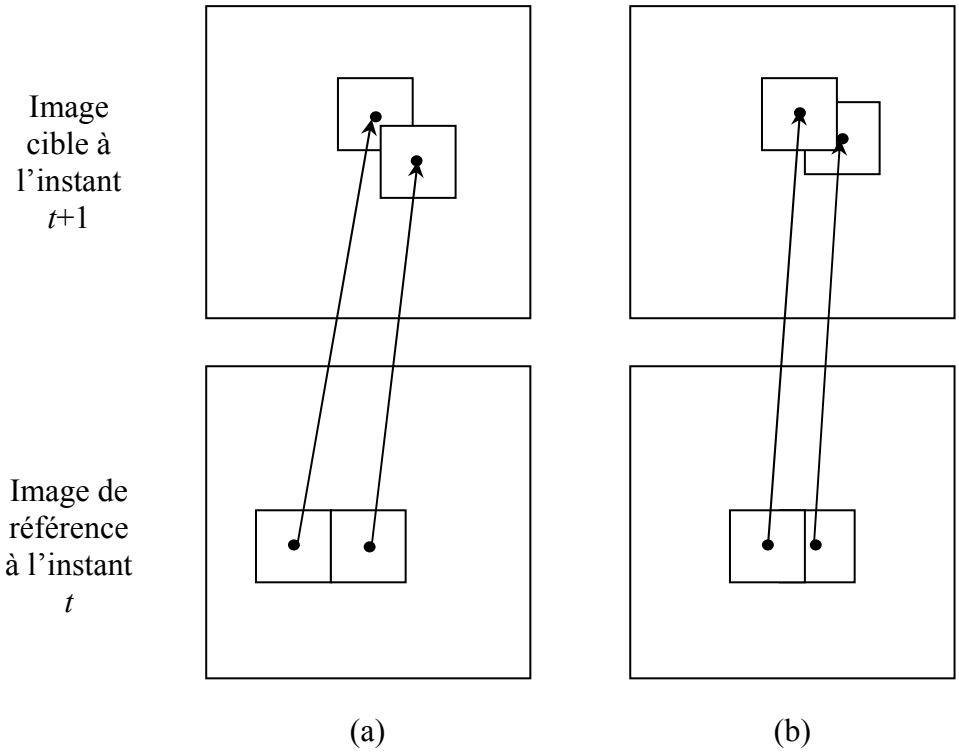


Figure II.7. Modèle de translation, avec des blocs: (a) disjoints; (b) chevauchants.

Les avantages de l'estimation de mouvement utilisant un modèle de translation du mouvement sont:

- la compression obtenue en attribuant un seul vecteur de mouvement par bloc;
- la facilite d'implantation hardware.

Un des inconvénients de l'estimation de mouvement utilisant un modèle de translation du mouvement est constitué par les artefacts de type bloc, dus à un sous-échantillonnage du champ de mouvement.

II.3.3.2. Méthodes non paramétriques :

L'avantage des méthodes non paramétriques est qu' e1les peuvent être utilisées pour régulariser des mouvements complexes.

Les principales approches non paramétriques sont:

a) Les méthodes différentielles : Les méthodes différentielles d'estimation du mouvement [05] s'appuient sur l'estimation des gradients spatio-temporels de l'intensité en chacun pixel. Dans le cas des images monochromatiques, les méthodes différentielles avec une contrainte de lissage spatio-temporel font l'hypothèse que le vecteur déplacement varie lentement au voisinage du pixel [02]. Dans le cas des images couleurs, cette contrainte peut être imposée séparément pour chaque couleur, ce qui peut contraindre le vecteur déplacement dans 3 directions différentes. En général, une contrainte supplémentaire est nécessaire pour obtenir des résultats satisfaisants. Dans le cas de superposition des objets (occultations), une contrainte globale appliquée à l'image conduit à une estimation inexacte du mouvement.

b) Les méthodes de mise en correspondance : Dans ces méthodes on suppose que l'image est divisée en plusieurs régions de mouvement. Il existe des méthodes de mise en correspondance dans le plan image ou dans le plan transforme [27].

c) Les méthodes récursives : Les méthodes récursives sont basées sur la correction d'une prédiction (ou d'un estimé) du vecteur déplacement [12]. Ainsi, l'estimée ou la prédiction peut être considérée comme étant le vecteur déplacement d'un pixel voisin ou une combinaison linéaire des déplacements des pixels dans un voisinage du pixel courant [04]. La correction de la prédiction est faite en minimisant le gradient de l'image de différence entre les images déplacées DFD. Le pas de la prédiction est considère, en général, comme une contrainte de lissage. Une extension de cette méthode dans le cas des méthodes de mise en correspondance de blocs, a comme résultat les méthodes d'estimation de type Wiener.

d) Les méthodes statistiques : Parmi les méthodes statistiques, les méthodes Markoviennes [19] ou Bayesiennes [30],[31] sont les plus répandues. Elles utilisent pour estimer le champ de déplacement des contraintes probabilistes de lissage, en général, sous la forme d'un champ aléatoire (éventuellement de Gibbs). Le principal désavantage de ces méthodes consiste en la grande quantité de calculs nécessaires pour estimer le champ de mouvement.

II.3.3.3. Modèles paramétriques du mouvement :

Le modèle de translation du mouvement, décrit par la relation (II.9) peut être réécrit sous la forme d'une transformation linéaire de coordonnées spatiales:

$$\begin{cases} x_c = x_r + d_x \\ y_c = y_r + d_y \end{cases} \qquad \text{(II.10)}$$

où (x_c,y_c) représentent les coordonnées du point de l'image cible, à l'instant $t+1$ et (x_r,y_r) représentent les coordonnées du point de l'image de référence, à l'instant t.

La transformation spatiale (II.10) peut être généralisée pour inc1ure des transformations affines des coordonnées:

$$\begin{cases} x_c = a_1.x_r + a_2.y_r + a_3.d_x \\ y_c = a_4.xr + a_5.yr + a_6.d_y \end{cases} \quad \text{(II.11)}$$

Où $a_i \in \boldsymbol{R}$, $i \in \{1,...,6\}$.

La transformation affine (II.11) peut traiter des mouvements de rotation du blocs ou des déformations relatives rigides (par exemple, la transformation d'un carré en parallélogramme). Parmi les transformations spatiales utilisées dans l'estimation du mouvement basées sur la mise en correspondance de blocs, on peut rappeler les transformations bilinéaires et projectives (figure II.8).

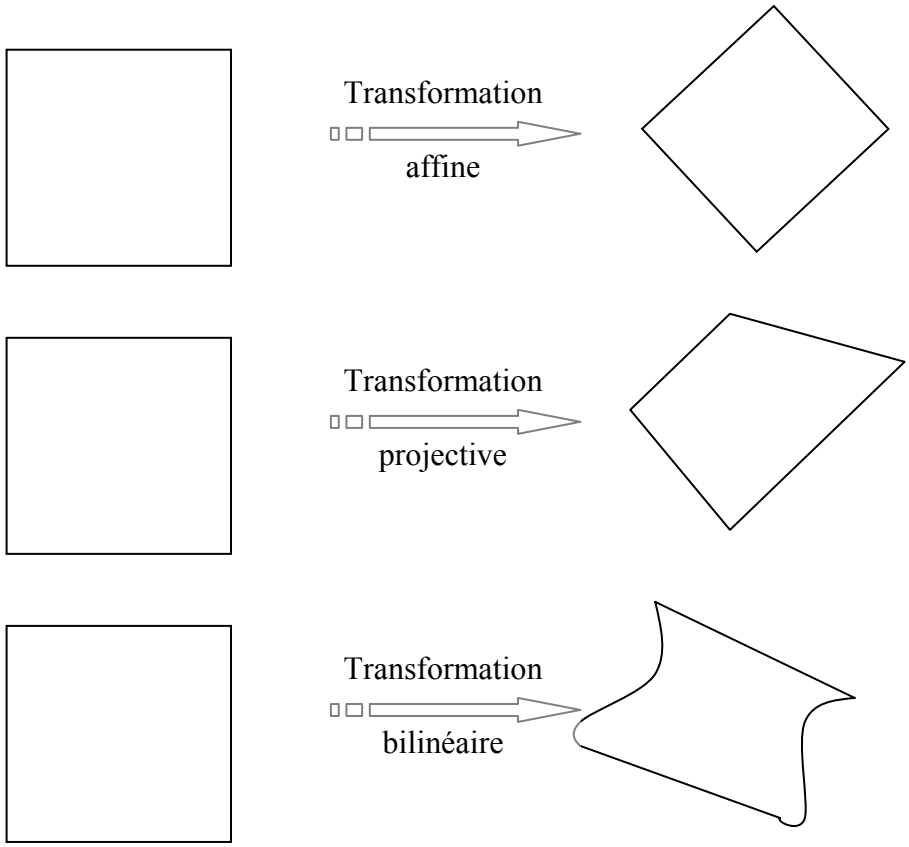


Figure II.8. Exemples de transformations des coordonnées spatiales.

Une transformation projective peut être décrite par la relation:

$$\begin{cases} x_c = \dfrac{b_1.x_r + b_2.y_r + b_3}{b_7.x_r + b_8.y_r + 1} \\ y_c = \dfrac{b_4.x_r + b_5.y_r + b_6}{b_7.x_r + b_8.y_r + 1} \end{cases} \quad \text{(II.12)}$$

où $b_i \in \boldsymbol{R}$, $i \in \{1,...,8\}$.

Une transformation bilinéaire peut être décrite par la relation:

$$\begin{cases} x_c = c_1.x_r + c_2.y_r + c_3.x_r.y_r + c_4. \\ y_c = c_5.x_r + c_6.y_r + c_7.x_r.y_r + c_8. \end{cases} \quad \text{(II.13)}$$

où $c_i \in \boldsymbol{R}$, $i \in \{1,...,8\}$.

Une transformation affine (resp. projective) correspond à la projection orthographique (resp. projective) du mouvement rigide 3D, dans le plan image 2D. La transformation bilinéaire n'a aucune correspondance physique particulière, mais elle présente des propriétés algébriques et géométriques intéressantes.

Une sous-classe de modèles paramétriques est représentée par les modèles appelés quasi-paramétriques, qui traitent la profondeur de chaque point comme une inconnue indépendante.

Le principal inconvénient des méthodes paramétriques est qu'elles sont applicables seulement dans le cas de mouvements rigides. Les différents modèles de mouvement présentés dans ce paragraphe, peuvent être vus comme des contraintes possibles de régularisation locale du champ de déplacement.

II.4. Méthodes différentielles :

Les méthodes différentielles sont les méthodes basées sur les gradients spatiaux et temporaux d'intensité lumineuse [33]. Ces gradients sont approches, dans le cas discret, par des différences finies.

II.4.1. Équation de contrainte du mouvement :

L'hypothèse d'invariance en temps de l'intensité lumineuse le long de la trajectoire du mouvement est exprimée par l'équation DFD (II.2). En réécrivant cette équation [05], on obtient:

$$\frac{dI(x,y,t)}{dt} = 0 \quad \text{(II.14)}$$

où x et y varient en temps, le long de la trajectoire du mouvement. Sous l'hypothèse de différentiabilité spatio-temporelle de l'intensité lumineuse et en utilisant les règles de différentiation connues, on obtient:

$$I(x+dx,\ y+dy,\ t+\Delta t)=I(x,y,t)+\frac{\partial I(x,y,t)}{\partial x}dx+\frac{\partial I(x,y,t)}{\partial y}dy+\frac{\partial I(x,y,t)}{\partial t}\Delta t+tos \quad \text{(II.15)}$$

où *tos* représente les termes d'ordre supérieur. En remplaçant ce développement dans l'équation de DFD en négligeant les termes d'ordre supérieur, on obtient:

$$DFD(x,y,t)=\frac{\partial I(x,y,t)}{\partial x}d_x+\frac{\partial I(x,y,t)}{\partial y}d_y+\frac{\partial I(x,y,t)}{\partial t}\Delta t=0 \quad \text{(II.16)}$$

Si on divise l'équation (II.16) par rapport à la distance en temps inter-images Δt, on obtient l'équation de contrainte du mouvement(ECM) nommée aussi l'équation du flux optique(EFO)[05] :

$$\frac{\partial I(x,y,t)}{\partial x}v_x(x,y,t)+\frac{\partial I(x,y,t)}{\partial y}v_y(x,y,t)+\frac{\partial I(x,y,t)}{\partial t}=0 \quad \text{(II.17)}$$

où: $v_x(x,y,t) = dx/\Delta t$, $vy(x,y,t) = d_y/\Delta t$ sont les deux composantes de la vitesse. L'équation ECM peut être réécrite, en chaque point (x,y,t):

$$I_x . v_x + I_y . v_y + I_t = 0 \qquad \text{où:} \quad < \nabla I, v > + \frac{dI}{dt} = 0 \tag{II.18}$$

Où: $\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) = \left(I^x, I^y \right)$ est Le gradient spatial de l'image avec les deux composantes $I^x = \frac{\partial I}{\partial x}$ et $I^x = \frac{\partial I}{\partial x}$, représentent respectivement $I^t = \frac{\partial I}{\partial t}$ est le gradient temporel de l'intensité, et $< . , . >$ représente le produit scalaire entre deux variables. Dans le cas discret, les gradients spatiaux et temporaux de l'image sont approches par des différences finies.

Comme on peut l'observer, l'équation de contrainte du mouvement n'est pas suffisante pour déterminer d'une manière unique le champ de vitesses, parce que l'ECM représente une équation à deux inconnues (v_x et v_y), en chacun pixel. La composante normale, $V\perp(x,y,t)$ du vecteur vitesse, orientée dans la direction du gradient spatial d' image $\frac{\nabla I}{\|\nabla I\|}$, est accessible. En effet, la composante tangentielle, tangente aux frontières photométriques des objets, disparaît suite au produit scalaire. La résolution du problème d'estimation du mouvement nécessite d'introduire une contrainte supplémentaire.

L'équation de contrainte du mouvement (ECM) décrit une droite dans l'espace de vitesses (v_x, v_y), comme sur la figure II.9:

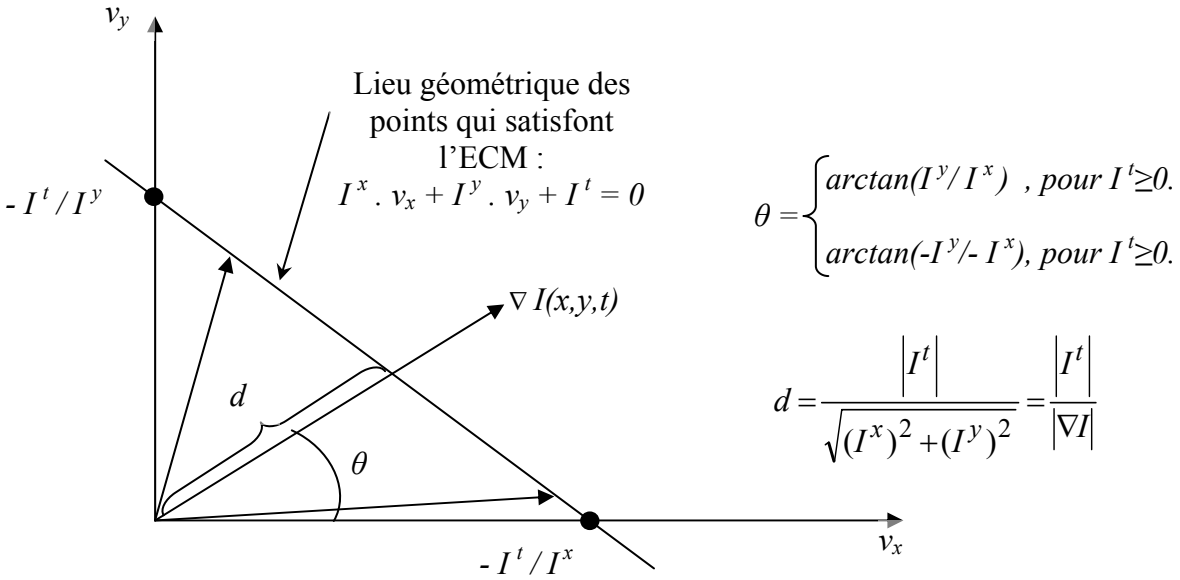


Figure II.9. Illustration géométrique de l'équation et de la droite de contrainte du mouvement.

La distance qui sépare la droite de contrainte du mouvement d'origine dépend du rapport des modules des gradients spatiaux et temporaux, sachant que l'orientation de cette droite est déterminée par l'orientation du gradient spatial au point d'intérêt.

II.4.2. Méthode de Horn et Schunck :

La méthode de Horn et Schunck [05] n'est pas limitée aux translations. C'est une méthode différentielle itérative adaptée à l'estimation des petits déplacements et basée sur le développement en série de Taylor du DFD, des gradients spatiaux et temporaux [06]. Le but est de trouver le champ de vecteurs qui satisfait l'équation de contrainte du mouvement en chacun pixel. Soit:

$$\xi_{flux}=I^x.v_x+I^y.v_y+I^t. \qquad \text{(II.19)}$$

L'erreur par rapport à l'équation de contrainte du mouvement (ECM), en chaque point (x,y,t). On observe que l'ECM est satisfaite quand l'erreur ξ_{flux} est nulle. Dans la méthode de Horn et Schunck on minimise le carre de l'erreur ξ_{flux}. Une contrainte supplémentaire de lissage qui suppose que tous les points voisins ont un mouvement semblable est utilisée:

$$\begin{aligned}\xi^2_{uniformité} &= \|\nabla v_x\|^2+\|\nabla v_y\|^2=\left(\frac{\partial v_x}{\partial x}\right)^2+\left(\frac{\partial v_x}{\partial y}\right)^2+\left(\frac{\partial v_y}{\partial x}\right)^2+\left(\frac{\partial v_y}{\partial y}\right)^2 \\ &= \left(v_x^x\right)^2+\left(v_x^y\right)^2+\left(v_y^x\right)^2+\left(v_y^y\right)^2\end{aligned} \qquad \text{(II.20)}$$

On Peut montrer [05] que l'erreur $\xi^2_{uniformité}$ est faible si le champ de vecteurs vitesse est lisse. La méthode de Horn et Schunck consiste donc à minimiser une énergie de la forme:

$$\xi^2=(1-\gamma).\xi^2_{flux}+\gamma.\xi^2_{uniformité} \qquad \text{(II.21)}$$

où γ est un coefficient de pondération entre les termes d'attache aux données et de lissage.

Pour minimiser ξ^2 (II.21) on utilise les équations d'Euler-Lagrange qui conduisent aux relations suivantes:

$$\begin{cases}\left(I^x\right)^2.v_x+I^x.I^y.v_y=\gamma.\left(\bar{v}_x-v_x\right)-I^x.I^t \\ \left(I^y\right)^2.v_y+I^x.I^y.v_x=\gamma.\left(\bar{v}_y-v_y\right)-I^y.I^t\end{cases} \qquad \text{(II.22)}$$

où $\bar{v}_x$ (resp. $\bar{v}_y$) est la valeur moyenne locale de v_x (resp. de v_y).

A partir de ce système on obtient v_x et v_y:

$$V_x=\Delta v_x/\Delta \qquad \text{et} \qquad V_y=\Delta v_y/\Delta,$$

avec : $$\Delta v_x=\begin{vmatrix}\gamma.\bar{v}_x-I^x.I^t & I^x.I^y \\ \gamma.\bar{v}_y-I^y.I^t & \gamma+\left(I^y\right)^2\end{vmatrix}.$$

$$\Leftrightarrow \Delta v_x=\gamma.(\gamma.\bar{v}_x+(I^y)^2.\bar{v}_x-I^x.I^t-\bar{v}_y.I^x.I^y).$$

et $\Delta = \gamma.(\gamma + (I^x)^2 + (I^x)^2)$.

$$\Rightarrow \begin{cases} \bar{v}_x = \dfrac{\Delta \bar{v}_x}{\Delta} = \dfrac{\bar{v}_x.(\gamma + (I^y)^2) - \bar{v}_y.I^x.I^y - I^x.I^t.}{\gamma + (I^x)^2 + (I^y)^2} \\ \bar{v}_y = \dfrac{\Delta \bar{v}_y}{\Delta} = \dfrac{\bar{v}_y.(\gamma + (I^x)^2) - \bar{v}_x.I^x.I^y - I^y.I^t.}{\gamma + (I^x)^2 + (I^y)^2} \end{cases} \quad \text{(II.23)}$$

Ainsi, en parcourant l'image entière, on obtient pour chaque pixel le couple de valeurs (v_x, v_y). Si on effectue plusieurs itérations (le nombre des itérations étant Le deuxième paramètre de la méthode de Horn et Schunck), ces valeurs vont diminuer, $\bar{v}_x$ et $\bar{v}_y$ étant des fractions des v_x et v_y. La solution est obtenue en utilisant la formule récurrente [05]:

$$\begin{cases} v_{x,n+1} = \bar{v}_{x,n} - \dfrac{I^x.\lfloor I^x.\bar{v}_{x,n} + I^y.\bar{v}_{y,n} + I^t \rfloor}{\gamma + (I^x)^2 + (I^y)^2} \\ v_{y,n+1} = \bar{v}_{x,n} - \dfrac{I^y.\lfloor I^x.\bar{v}_{x,n} + I^y.\bar{v}_{y,n} + I^t \rfloor}{\gamma + (I^x)^2 + (I^y)^2} \end{cases} \quad \text{(II.24)}$$

La convergence est obtenue quand l'erreur (c'est-à-dire la différence entre deux itérations successives de v_x et v_y) est inférieure à un seuil choisi ou pour un nombre (n) d'itérations choisi.

II.4.3. Méthodes différentielles avancées :

La méthode de Horn et Schunck impose l'équation de contrainte du mouvement (II.18) et la contrainte de lissage du champ de vitesse au niveau global (II.20), dans l'image entière ou dans la fenêtre d'estimation du mouvement. Cette globalité spécifique aux méthodes non adaptatives a au moins deux effets non souhaités:

- l'effet de "flou" sur les frontières du mouvement estimé, dû à la contrainte de lissage (II.20) dans la direction perpendiculaire à frontière d'un objet en mouvement. Les frontières du mouvement peuvent être préservées en imposant la contrainte de lissage seulement dans la direction dans laquelle le champ de vitesse varie peu;
- les méthodes non adaptatives imposent l'équation de contrainte du mouvement dans les zones d'occultation, où l'effet de cette équation devrait être annulé. Ce résultat peut être obtenu en modifiant localement le coefficient de pondération γ. Par exemple, dans les zones d'occultation on doit conserver seulement la contrainte de lissage.

En général, les méthodes différentielles adaptatives supposent une contrainte de lissage directionnel, en particulier le long des contours du mouvement et pas dans la direction perpendiculaire. Ainsi, dans on suppose la minimisation du même critère (II.21) seulement le long des contours.

II.4.3.1. Méthodes différentielles directionnelles :

La contrainte de lissage directionnel peut être exprimée par la relation:

$$\xi_{ud}^2(v) = (\nabla v_x)^T .W.(\nabla v_x) + (\nabla v_y)^T .W.(\nabla v_y) \qquad \text{(II.25)}$$

où *W*est une matrice de pondération qui pénalise les variations du champ de mouvement en fonction de variations spatiales de niveaux de gris par la séquence d'images . Un exemple de définition de la matrice *W*est :

$$W = \frac{F + \delta I}{trace(F + \delta I)} \qquad \text{(II.26)}$$

où *I* est la matrice identité qui représente le terme de lissage global nécessaire pour assurer une matrice de pondération non nulle dans les zones d'uniformité spatiale, δ est un scalaire et:

$$F = \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 + b^2\left(\left(\frac{\partial^2 I}{\partial x^2}\right)^2 + \left(\frac{\partial^2 I}{\partial x.\partial y}\right)^2\right) & \frac{\partial I}{\partial x}.\frac{\partial I}{\partial y} + b^2\frac{\partial^2 I}{\partial x.\partial y}\left(\left(\frac{\partial^2 I}{\partial x^2}\right)^2 + \left(\frac{\partial^2 I}{\partial y^2}\right)^2\right) \\ \frac{\partial I}{\partial x}.\frac{\partial I}{\partial y} + b^2\frac{\partial^2 I}{\partial x.\partial y}\left(\left(\frac{\partial^2 I}{\partial x^2}\right)^2 + \left(\frac{\partial^2 I}{\partial y^2}\right)^2\right) & \left(\frac{\partial I}{\partial y}\right)^2 + b^2\left(\left(\frac{\partial^2 I}{\partial y^2}\right)^2 + \left(\frac{\partial^2 I}{\partial x.\partial y}\right)^2\right) \end{bmatrix}^{-1} \qquad \text{(II.27)}$$

où b^2 est une constante.

Ainsi, dans le cadre de méthodes d'estimation du mouvement avec un lissage directionnel on minimise la fonction suivante, qui est similaire à la fonction utilisée dans le cas de la méthode de Horn et Schunck (II.21):

$$\xi^2 = (1-\gamma).\xi_{flux}^2 + \gamma.\xi_{ud}^2 \qquad \text{(II.28)}$$

où γ est un coefficient de lissage.

On observe que la méthode de Horn et Schunck (II.21) représente un cas particulier de cette formulation et correspond à l'équation (II.28), pour: δ=1 et ***F***=O.

Pour minimiser la fonction (II.28) on peut utiliser une minimisation de type Gauss-Seidel, dans laquelle le terme actualisé à chaque itération peut être calculé par un algorithme linéaire. Les performances de la méthode avec lissage directionnel dépendent de la précision avec laquelle on peut estimer les dérivées partielles d'ordre 1 et surtout celles d'ordre 2 de l'intensité de l'image, qui interviennent dans l'équation (II.28).

II.4.3.2. Méthodes différentielles hiérarchiques :

Le concept de lissage directionnel et d'adaptabilité pondérée est décrit au travers une formulation hiérarchique [26]. Une représentation multirésolution d'une séquence d'images peut être définie de la manière suivante:

$$I_\alpha(x,y,t) = \int_S I(x,y,t).h\left(\frac{m-x}{\alpha},\frac{n-y}{\alpha}\right)dmdn \qquad \text{(II.29)}$$

où α est Le paramètre de résolution, S représente le support d'image et h est un noyau de filtrage. Un exemple de filtre passe-bas de paramètres A, B, C et D est donne par:

$$h(x,y) = \begin{cases} A.\exp\left[\dfrac{-\left(C^2.x^2 + D^2.y^2\right)}{B^2 - \left(C^2.x^2 + D^2.y^2\right)}\right], & \text{pour}\left(C^2.x^2 + D^2.y^2\right) < B^2 \\ 0 & \text{sinon} \end{cases} \qquad \text{(II.30)}$$

On peut observer que la résolution spatiale diminue quand α augmente. On peut observer aussi que les dérivées spatiales d'intensité $I_\alpha(x,y,t)$ peuvent être calculées par la convolution de $I(x,y,t)$ avec les dérivées partielles de $h(x,y)$, qui peuvent être calculées analytiquement. A chaque niveau de résolution on peut utiliser une méthode d'optimisation non-linéaire, comme par exemple la méthode de Quasi-Newton. L'estimation obtenue à un niveau de résolution est utilisée comme estimation initiale au niveau supérieur de résolution.

Pour augmenter l'efficacité des méthodes différentielles, on peut combiner différentes approches: l'approche hiérarchique avec l'approche directionnelle ou la prise en compte de discontinuités dans le champ de mouvement.

II.5. Méthodes de mise en correspondance :

Les méthodes de mise en correspondance sont parmi les plus utilisées en pratique grâce à la simplicité de l'implantation physique du modèle de translation utilise le plus souvent dans ces méthodes. Elles sont utilisées dans les standards de compression vidéo, comme H.261, MPEG-1, MPEG-2, MPEG-4 et MPEG-7.

Parmi les méthodes de mise en correspondance, on distingue deux classes:

- les méthodes de mise en correspondance dans le plan transformé;
- les méthodes de mise en correspondance dans le plan image.

II.5.1. Méthodes de mise en correspondance dans le plan transformée :

Dans les méthodes de mise en correspondance dans le plan transforme, le champ de déplacement d est estimé par la mise en correspondance de blocs définis dans un plan transforme. La méthode la plus répandue est la méthode de corrélation de phase obtenue par la transformée de Fourier. Dans cette méthode, on calcule la corrélation de phase entre les transformées de Fourier de deux blocs correspondants dans les deux images successives.

On va noter: $\Im\{I(x,y,t)\} = F_t(f_x,f_y) = F_t(f)$ la transformée de Fourier de l'image $I(x,y,t)$ à l'instant t, où $f = (f_x,f_y)$ représente les fréquences spatiales. La transformée de Fourier de l'image à l'instant $t+\Delta t$: $J(x,y,t+\Delta t)=I(x+d_x,y+d_y,t)$ est:

$$\Im_{t+\Delta t}\{J(x,y,t+\Delta t)\} = F_{t+\Delta t}(f) = F_t(f).\exp\{-j.2\pi.(f_x.d_x + f_y.d_y)\} \qquad \text{(II.31)}$$

où $I(p,t)$ est l'intensité d'image au point $p=(x,y)$, à l'instant t, x et y représentent les coordonnées spatiales. $\Delta t = \pm 1$ représente l'intervalle de temps entre les deux images et $d(p)=(d_x(p),d_y(p))$ est le

vecteur déplacement correspondant au point *p=(x,y)*.

Dans le cas d'un modèle de translation du mouvement, équivalent à l'équation (II.9) ou (II.10), la différence de phase entre les transformées de Fourier des deux blocs est donnée par:

$$\arg\{F_t(f)/F_{t+\Delta t}(f)\} = \Delta\varphi(f_x, f_y) = -2\pi.(f_x.d_x + f_y.d_y) \quad \text{(II.32)}$$

L'équation (II.32) détermine un plan dans l'espace des variables f_x et f_y .Le mouvement entre les deux images consécutives peut être exprime par la différence de phase entre deux paires de fréquences spatiales. Il est donne par un système linéaire des deux équations, correspondant ainsi à l'orientation du plan (équation (II.32)).

Les méthodes de corrélation de phase estiment le déplacement relatif entre deux images consécutives en utilisant une fonction normalisée d'inter-corrélation calculée dans l'espace de Fourier. Le résultat est un Dirac dont la position correspond au vecteur déplacement.

Un exemple d'une telle fonction ou coefficient d'inter-corrélation entre les images *t* et *t+ Δt* est:

$$c_{t,t+\Delta t}(x,y) = I_{t,t+\Delta t}(x+d_x, y+d_y) * I_t(-x,-y) \quad \text{(II.33)}$$

où * représente l'opération de convolution. La transformée de Fourier de l'équation (II.33) fournit l'expression complexe du spectre inter-images, *F** étant la conjuguée de *F*:

$$C_{t,t+\Delta t}(f_x, f_y) = F_{t+\Delta t}(f_x, f_y).F_t(f_x, f_y) \quad \text{(II.34)}$$

En normalisant l'amplitude du spectre inter-images (II.34), on obtient la phase du spectre:

$$\arg\{C_{t,t+\Delta t}(f_x, f_y)\} = \frac{F_{t+\Delta t}(f_x, f_y).F_t^*(f_x, f_y)}{\left|F_{t+\Delta t}(f_x, f_y).F_t^*(f_x, f_y)\right|} \quad \text{(II.35)}$$

En supposant le modèle de translation (II.4) et en remplaçant (II.32) en (II.35), on obtient:

$$\arg\{C_{t,t+\Delta t}(f_x, f_y)\} = \exp\{-j.2\pi.(f_x.d_x + f_y.d_y)\} \quad \text{(II.36)}$$

La transformée de Fourier inverse (II.36), donne la fonction de corrélation de phase:

$$\arg\{c_{t,t+\Delta t}(f_x, f_y)\} = \delta(x-d_x, y-d_y) \quad \text{(II.37)}$$

$$\delta(q) = \begin{cases} 1, & pour \quad q=0 \\ 0, & ailleurs \end{cases}$$

En pratique, la transformée de Fourier est remplacée par la Transformée de Fourier (TF) discrète, pour laquelle il y a des algorithmes rapides de calcul. Un exemple d'un tel algorithme utilisant la corrélation de phase est décrit comme suit :

1. Calcul de *TF* discrète 2D des blocs correspondants dans les images *t* et *t+ Δt*;
2. Calcul de la phase du spectre inter-images (II.36);

3. Calcul de la *TF* inverse de la relation (II.36), afin d'obtenir la fonction de corrélation de phase (II.37).

4. Détection de la position du pic du maximum de corrélation de phase, donnant la valeur estimée du déplacement $\hat{d} = (\hat{d}_x, \hat{d}_y)$.

Dans le cas idéal, la fonction de corrélation de phase doit présenter un seul maximum (pic) qui indique le déplacement relatif entre les deux blocs. En pratique, la fonction de corrélation de phase peut présenter plusieurs pics à cause de :

- L'utilisation de la transformée de Fourier discrète (TFD) au lieu de la transformée de Fourier continue ;
- la présence de bruit ;
- la présence à l'intérieur du bloc de référence de plusieurs objets en mouvement.

Les méthodes de corrélation de phase ont plusieurs avantages qui sont :

- Leur insensibilité aux variations d'illumination de la scène. Les méthodes de corrélation de phase sont relativement insensibles aux modifications de l'illumination parce qu'un déplacement ou une multiplication par une constante de la valeur moyenne de l'intensité n'affecte pas la phase de la transformée de Fourier. Les méthodes de corrélation de la phase sont insensibles aux variations d'amplitude de la transformée de Fourier.
- Leur capacité à traiter plusieurs objets en mouvement. La possibilité de traiter plusieurs objets en mouvement ayant des vitesses différentes dans un même bloc de calcul est une propriété très importante. Celle-ci se traduit par la présence de plusieurs pics dans la fonction de corrélation de phase, chacun d'entre eux correspondant au déplacement d'un objet. Dans ce cas, une étape supplémentaire est nécessaire pour déterminer quel vecteur déplacement correspond à quel pixel du bloc de calcul. Cette étape peut être réalisée en comparant l'amplitude de la différence entre les images déplacées (DFD) avec chaque vecteur déplacement.

En pratique, les méthodes de corrélation de phase sont rarement utilisées à cause de plusieurs inconvénients, parmi lesquels on retiendra :

- Les effets de bord. Pour obtenir un impulsion parfaite, le déplacement doit être cyclique (à cause du calcul par transformée de Fourier), ce qui n'est pas le cas en pratique car les objets qui disparaissent à un bord du bloc de calcul ne réapparaissent pas sur l'autre bord. Ceci entraînera donc la présence de pics non souhaités, conduisant à une estimation erronée du mouvement.
- Les déformations du spectre, dues aux valeurs non entières correspondant aux vecteurs de mouvement. Pour obtenir un pic idéal, les composantes du vecteur déplacement doivent correspondre à un multiple de la fréquence fondamentale. Dans le cas contraire, de faux pics peuvent apparaître, déformant Le spectre.
- L'amplitude maximale du de placement qui peut être estimée. Parce que la TFD est périodique, de période égale à la dimension du bloc ($B_x \times B_y$), Le déplacement estimé doit être réécrit :

$$\hat{d}_i = \begin{cases} d_i & \text{,si} |d_i| \le d_{\max}/2, \text{quand } d_{\max} \text{est pair, ou si } |d_i| \le (d_{\max}-1)/2, \text{quand } d_{\max} \text{est impair} \\ d_i - d_{\max} & \text{,en reste} \end{cases}$$

- pour permettre des déplacements négatifs. Ainsi, quand le déplacement maximal (d_{max}) qui peut être estimé est pair, la gamme maximale du déplacement qui peut être estimé est [(-d_{max}/2)+ 1, d_{max}/2]. Par exemple, pour estimer un déplacement maximal de 8 pixels, c'est-à-dire une amplitude maximale du déplacement comprise entre [-7, +8] pixels, la dimension du bloc de calcul doit être de [16×I6] pixels.

La dimension du bloc est un paramètre important dans tous les algorithmes d'estimation du mouvement utilisant la mise en correspondance. Le choix de la dimension du bloc suppose un compromis entre deux contraintes contradictoires. Dans le cas des méthodes de corrélation de phase, le bloc doit être suffisamment grand pour estimer de grands déplacements. Mais, celui-ci doit rester suffisamment petit pour que le déplacement soit constant à l'intérieur du bloc. Une manière de résoudre le problème consiste à utiliser des méthodes hiérarchiques multirésolutions.

II.5.2. Méthodes de mise en correspondance dans le plan image :

La méthode de mise en correspondance par bloc dans le plan image Block Matching en Anglais est une méthode largement utilisée dans l'estimation de mouvement, pour les grand famille de codecs MPEG (MPEG, MPEG-1, MPEG-2, et MPEG-4) ainsi pour les codecs H.261 et H263.

Les différents techniques de recherches appartient à cette méthode sont discutées dans le chapitre .III.

Chapitre III :

Méthodes de Mise en Correspondance dans le Plan Image

Chapitre III :

Méthodes de Mise en Correspondance dans le Plan Image

III.1. Introduction :

Parmi les méthodes de mise en correspondance dans le plan image, les méthodes de mise en correspondance de blocs **"Block Matching"** sont les plus utilisées, grâce a leur simplicité d'implantation software et hardware. Ce type de méthode s'est impose dans les standards de compression d'images, comme H.261 ou MPEG-1, MPEG-2, MPEG-4, et MPEG-7 [07],[10],[25].

Les méthodes de mise en correspondance par blocs sont basées sur l'hypothèse selon laquelle l'intensité lumineuse des pixels est constante ou faiblement variable le long des trajectoires de mouvement. Pour chaque point de l'image courante à l'instant t on cherche le point d'intensité la plus proche dans l'image cible (instant t-1 ou t+ 1). Pour éviter les correspondances incohérentes dans l'image cible, on limite la recherche à une fenêtre centrée sur le point courant (x,y). Ainsi, le déplacement d'un pixel de coordonnées (x,y), centre du bloc B de dimension $B_x \times B_y$ de l'image courante, est déterminé par la position du bloc de même dimension, le plus proche au sens d'un critère de similarité, dans l'image cible (instants t-l ou t+ 1), figure III.1. La recherche est limitée à une fenêtre de dimensions: $F = (2.d_{x\_max}+1) \times (2.d_{y\_max}+1)$, où d_{x_max} (resp. d_{y_max}) est la valeur maximale du déplacement à estimer dans la direction x (resp. y).

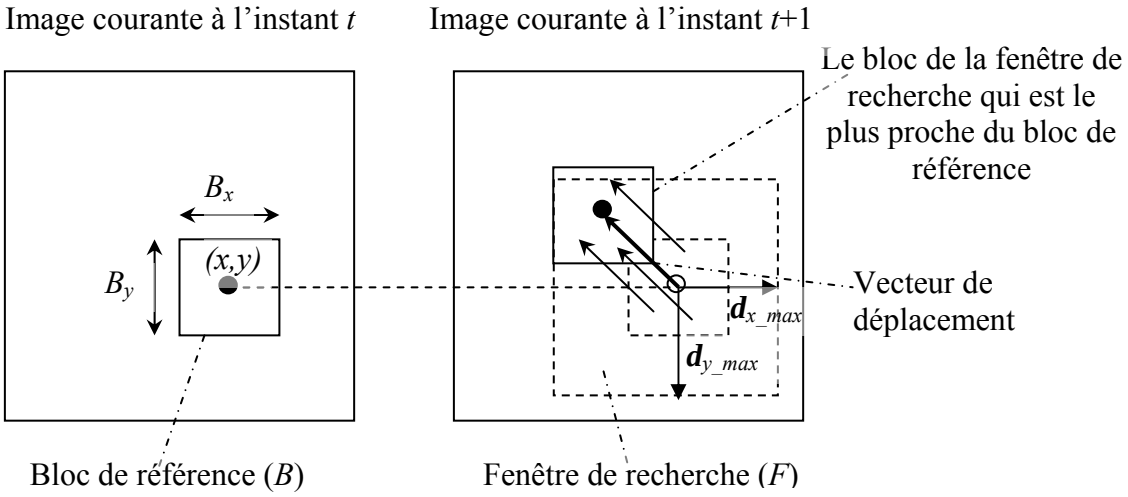


Figure III. 1. Principe de mise en correspondance par blocs.

La ressemblance entre deux blocs est calculée sur l'ensemble du bloc. Ceci sous entend une hypothèse supplémentaire selon laquelle tous les pixels d'un bloc effectuent le même mouvement. On considère en général, un modèle de translation (figure III.1) mais on peut considérer aussi des modèles plus complexes (voir le paragraphe II.3.3). Pour une translation, le vecteur déplacement (resp. vitesse) obtenu est celui qui lie les 2 blocs correspondants de centre à centre. Les méthodes de mise en correspondance de blocs peuvent être c1assees en fonction:

- du critère de mise en correspondance;
- de la mise en œuvre nécessitant d'optimiser 2 éléments: la dimension du bloc et celle

de la fenêtre de recherche;

- de la stratégie de balayage (ou de recherche) de la fenêtre de recherche et/ou de l'image cible.

III.2. Critères de mise en correspondance :

Parmi les critères de différences, les plus utilisés sont:

- la différence moyenne absolue;
- l'erreur moyenne quadratique.

III.2.1. Différence absolue :

Le critère le plus utilisé en pratique dans l'implantation hardware est la minimisation de l'erreur absolue MAD, définie comme [27]:

$$MAD(l,k) = \frac{1}{B_x.B_y}\sum_{i=1}^{B_x}\sum_{j=1}^{B_y}\left|B_r(i,j) - B_c(i,j)\right|$$

$$= \frac{1}{B_x.B_y}\sum_{i=1}^{B_x}\sum_{j=1}^{B_y}\left|I(i,j,t_k) - I(i+d_x, j+d_y, t_k+\delta t)\right| \qquad \text{(III.1)}$$

où (l,k) sont les coordonnées du centre du bloc et (i,j) sont les coordonnées des pixels du bloc.

III.2.2. Erreur quadratique moyenne :

L'erreur quadratique moyenne (MSE) peut être définie comme [14]:

$$MSE(l,k) = \frac{1}{B_x.B_y}\sum_{i=1}^{B_x}\sum_{j=1}^{B_y}\left[B_r(i,j) - B_c(i,j)\right]^2 \qquad \text{(III.2)}$$

L'estimé du vecteur déplacement est $\hat{d} = (\hat{d}_x, \hat{d}_y)$ qui minimise MSE:

$$MSE(l,k) = \frac{1}{B_x.B_y}\sum_{i=1}^{B_x}\sum_{j=1}^{B_y}\left[B_r(i,j) - B_c(i,j)\right]^2$$

$$= \frac{1}{B_x.B_y}\sum_{i=1}^{B_x}\sum_{j=1}^{B_y}\left[I(i,j,t_k) - I(i+d_x, j+d_y, t_k+\delta t)\right]^2 \qquad \text{(III.3)}$$

$$\Rightarrow \hat{d} = (\hat{d}_x, \hat{d}_y) = \arg_{(d_x,d_y)} \min MSE(d_x, d_y) \qquad \text{(III.4)}$$

La minimisation de MSE peut être vue comme l'équivalent de la contrainte du mouvement (ECM) pour tous les pixels du bloc de référence, *argmin* (ξ_{flux}) définit dans la méthode de Horn et Schunck (équations II.17-II.19). La minimisation de l'erreur quadratique MSE est très rarement utilisée dans la version hardware, à cause de sa difficulté d'implantation.

Coefficient d'intercorrélation :

Le critère de ressemblance statistique le plus utilise est le coefficient d'intercorrélation centré et normé à la variance, qui s'écrit dans le cas discret:

$$C(l,k) = \frac{\sum_{i=1}^{B_x}\sum_{j=1}^{B_y}[B_r(i,j)-\mu_r][B_c(i,j)-\mu_c]}{\sqrt{\sum_{i=1}^{B_x}\sum_{j=1}^{B_y}[B_r(i,j)-\mu_r]^2}\sqrt{\sum_{i=1}^{B_x}\sum_{j=1}^{B_y}[B_c(i,j)-\mu_c]^2}} \quad \text{(III.5)}$$

Où μ_r(resp. μ_c) représente la moyenne des intensités des pixels du bloc de référence (resp. cible). Ce critère donne des bons résultats dans la majorité des cas de mouvements estimés, mais il a le désavantage majeur d'un temps de calcul long, supérieur aux critères de différence, à cause du grand nombre d'opérations impliquées (additions, soustractions, multiplications, divisions, radicaux).

D'une façon générale, les performances de l'opération de mise en correspondance diminuent quand les dimensions de la fenêtre de recherche augmentent, à cause de la présence possible de plusieurs minima locaux.

III.3. Dimension optimale du bloc et de la fenêtre de recherche :

La stratégie de recherche et les dimensions de la fenêtre de recherche et du bloc influencent le coût de calcul. La détermination de leurs dimensions optimales respectives nécessite toujours un compromis. Une grande fenêtre de recherche implique des calculs longs et un risque élevé de confusion du bloc recherche avec un bloc semblable. Mais plus la taille de la fenêtre de recherche décroît, plus le déplacement maximal estimé diminue. De la même manière, quand les blocs sont grands, le coût de calcul est grand, la résolution spatiale est faible et un bloc peut contenir des pixels appartenant à des objets différents. Inversement, un bloc trop petit peut ne pas contenir suffisamment d'information discriminante. Ce denier problème apparaît, en particulier, dans les zones homogènes où la mise en correspondance n'est pas fiable, pouvant devenir aléatoire.

En plus des problèmes déjà mentionnés (temps de calcul, résolution spatiale et fiabilité de la mise en correspondance) on peut rappeler les limites de l'hypothèse selon laquelle tous les pixels d'un bloc ont le même vecteur déplacement. La méthode suppose des mouvements de translation ou localement assimilables (figure III.1), mais cette hypothèse n'est pas respectée, par exemple dans le cas d'une rotation ou d'une homothétie rapide par rapport à la fréquence temporelle d'échantillonnage. Elle n'est pas respectée non plus à la frontière aux bords des objets animés de mouvement différent. Pour résoudre ce type de problème il est nécessaire de choisir un modèle de mouvement plus complexe (voir le paragraphe II.3.3) dans lequel la mise en correspondance s'effectue sur des blocs de formes arbitraires ou sur des contours.

III.4. Stratégies de recherche :

La méthode de mise en correspondance de blocs suppose une stratégie de recherche exhaustive dans l'image et dans la fenêtre de recherche. Un balayage exhaustif peut s'envisager pour l'image et également pour la fenêtre de recherche, le bloc de comparaison étant alors déplace pixel par pixel, à l'intérieur de la fenêtre de recherche. Le balayage exhaustif pressente le désavantage d'un temps de calcul prohibitif, car il impose d'estimer le critère de ressemblance en chacun pixel et pour toutes les positions possibles du bloc dans la fenêtre de recherche.

Le balayage non exhaustif nécessite le choix d'une stratégie. On envisagera les 2 cas classiques suivants (figure III.2):

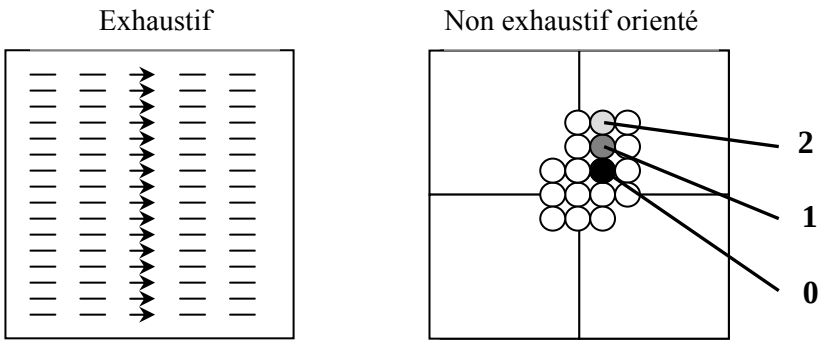


Figure III. 2. Stratégies de balayage de la fenêtre de recherche.

Le balayage non exhaustif orienté consiste en une exploration initiale du voisinage direct de la fenêtre de recherche puis sélective dans la direction de la ressemblance croissante (points noirs sur la figure III.2). Les stratégies de recherche orientée les plus utilisées sont:

- la recherche en *n* pas;
- la recherche 2D-Iogarithmique (2DLog);
- la recherche orthogonale.

Ces stratégies de balayage sont plus rapides, mais peuvent échouer dans le cas d'existence de plusieurs optima (maxima ou minima) locaux.

III.4.1. Recherche en n pas :

L'algorithme de recherche en 3 pas est base sur un raffinement successif, avec décroissance logarithmique du pas (figure III.3) [09]. La flèche indique la direction et le sens de la recherche pour le pas suivant.

Le principe de recherche en trois pas, comme les principes des autres méthodes de balayage qui seront présentées par la suite, sera illustré en considérant un déplacement maximal de 7 pixels. Le pixel note "0" sur la figure III.3 représente le pixel courant. A la première itération le critère de ressemblance est évalue, en 9 points: le point noté "0" et les points notés" 1 ".

Si le critère de ressemblance optimal correspondant est le point "0", le déplacement estimé est nul. Le pas initial (1) est choisi égal à la moitié du déplacement maximal admis d_{max} (on va considérer $d_{x\_max}= d_{y\_max}=d_{max}$), c'est-à-dire $\lceil d_{\max}/2 \rceil$ où $\lceil . \rceil$ est la fonction d'arrondi à l'entier supérieur. A chaque itération, on évalue le critère de ressemblance et l'optimum devient le centre de recherche pour l'itération suivante. L'amplitude du pas, à chaque itération, décroît selon une loi logarithmique.

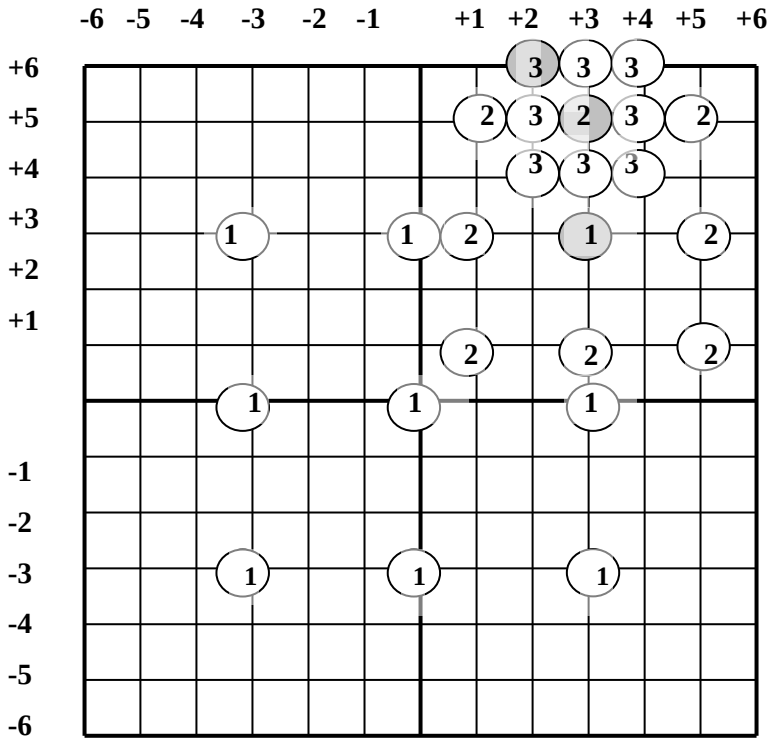


Figure III.3. Le principe de recherche en trois pas.

L'estimation du déplacement sera obtenue à la troisième itération, le pas vaut 1 pixel. Pour d_{max}=7, le nombre total de points de comparaison nécessaire est: (9+8+8)=25. L'algorithme peut être étendu à des fenêtres de recherche plus larges. En utilisant la même stratégie, Le nombre n de points de comparaison nécessaires sera n=1+8. $\lceil \log 2(d_{\max} + 1) \rceil$. L'algorithme est dit "recherche en n-pas" ou "recherche en log-n".

III.4.2. Recherche 2D-logaritmique :

La recherche 2D-Iogarithmique met en œuvre une stratégie de recherche en croix (+) à chaque itération. Le pas initial est égal à $\lceil d_{\max}/4 \rceil$. Il est réduit de moitié quand le point correspondant à l'optimum trouve est le centre ou situé au bord de la fenêtre de recherche. Dans le cas contraire, le pas ne change pas. Quand le pas atteint la valeur 1, les 8 points immédiatement voisins du pixel central sont testés. Sur la figure III.4 on présente deux situations différentes: la situation en haut nécessite: n=5+3+3+8=19 points de calcul et la situation en bas et à droite nécessite: n=5+3+2+3+2+8=23 points de calcul.

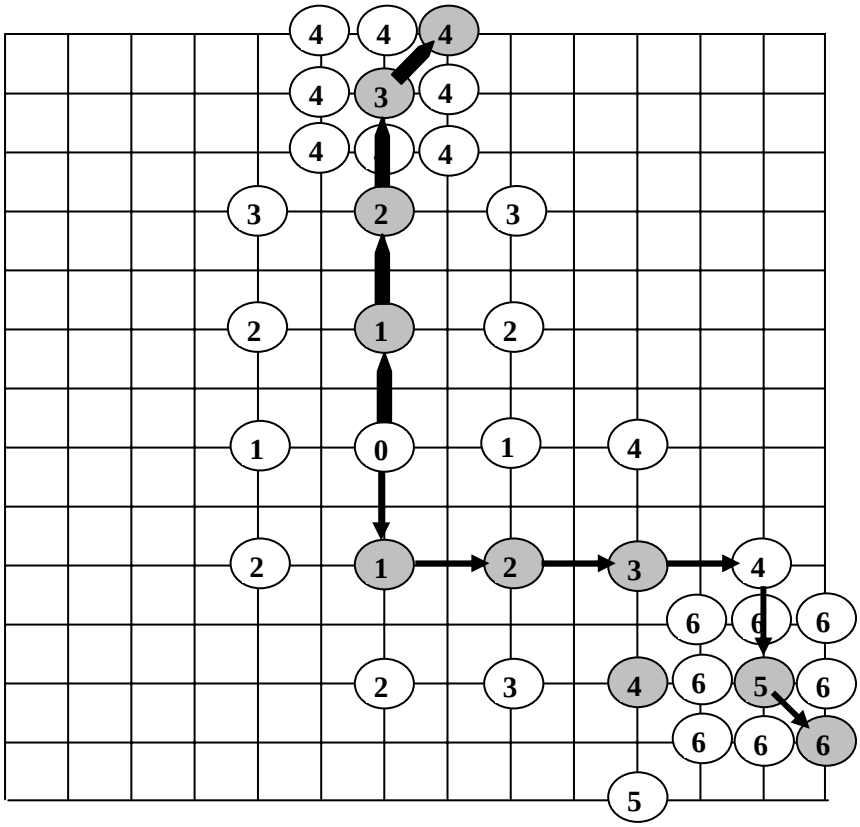


Figure III. 4. Le principe de recherche 2D-logarithmique.

III.4.3. Recherche orthogonale :

L'algorithme de recherche orthogonale consiste en paires de comparaisons horizontales et verticales avec une décroissance logarithmique du pas [24]. La dimension initiale du pas est $\lfloor d_{\max}/2 \rfloor$, où $\lfloor . \rfloor$ est la fonction d'arrondi à l'entier inférieur. Le principe de recherche orthogonale est montré sur la figure III.5.

Une itération à deux étapes. Dans la première étape, trois calculs du critère de ressemblance sont faits en trois pixels horizontaux, 0 et 1. L'optimum dans la direction horizontale devient le centre de la deuxième étape (recherche dans la direction verticale), le pas restant inchangé. Le pas est alors diminué de moitie en répétant la même stratégie de recherche à l'itération suivante. L'algorithme s'arrête quand le pas devient égal à 1. Pour d_{max}=7, dans le cas présente sur la figure III.5 en haut à droite, l'algorithme de recherche orthogonale nécessite d'estimer le critère en: n=3+2+2+2+2+2=13 points de calcul Dans le cas général: n=1+4. $\lfloor \log2(d_{\max}+1) \rfloor$ points de calcul sont nécessaires.

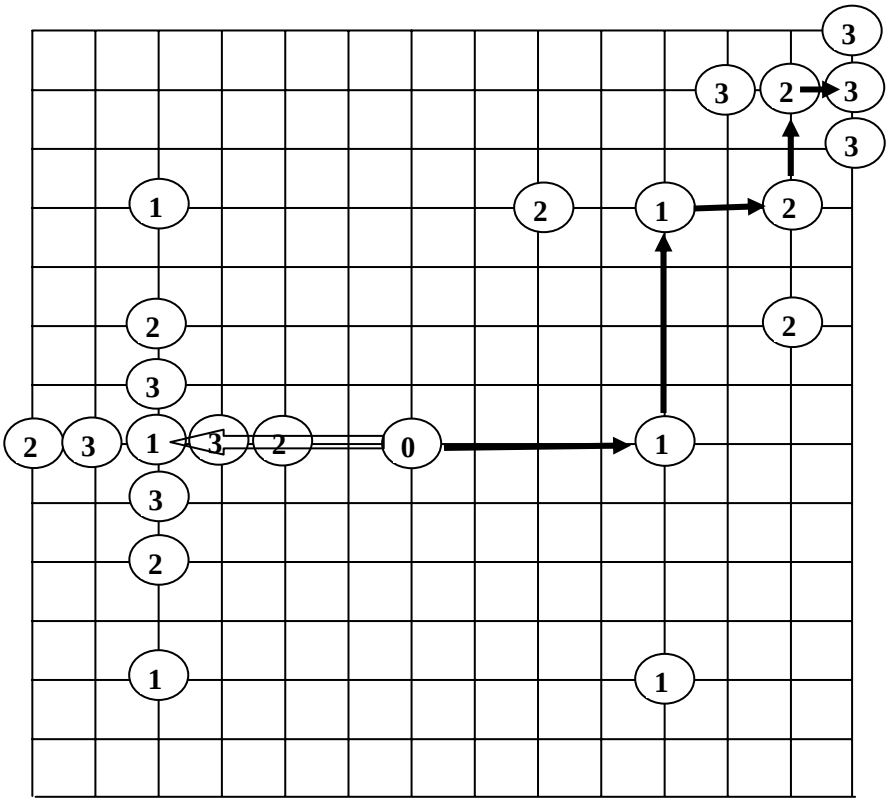


Figure III. 5. Principe de recherche orthogonale.

III.5. Méthodes avancées de mise en correspondance :

Une des limitations des méthodes c1assiques de mise en correspondance est le temps de calcul, qui augmente avec le carré de l'amplitude maximale du mouvement. En effet la fenêtre de recherche peut être assimilée à un cerc1e centre sur le pixel central et de rayon égal à l'amplitude maximale du mouvement (d_{max}). De plus, les méthodes alternatives à la recherche exhaustive, peuvent échouer dans le cas d'existence de plusieurs optima (maxima ou minima) locaux. Pour diminuer le temps de calcul, on peut utiliser des méthodes avancées qui garantissent l'optimum global et un temps de calcul plus faible que dans le cas de la recherche exhaustive. Parmi les méthodes avancées qui existent, on a sélectionne: l'algorithme d'élimination successive, la mise en correspondance hiérarchique multirésolution et la mise en correspondance de blocs déformables.

III.5.1. Algorithme d'élimination successive :

L'algorithme d'élimination successive (SEA) établit une contrainte dans le processus de recherche de l'algorithme classique de mise en correspondance de blocs. On suppose que la dimension du bloc est $B \times B$ (B_x=B_y=B) pixels, que la dimension de la fenêtre de recherche est $(2.d_{max} + 1) \times (2.d_{max} + 1)$ pixels et $I(i,j,t)$ est l'intensité du pixel de coordonnées (i,j) de l'image courante, à l'instant t. En utilisant l'inégalité triangulaire ($|a+b| \leq |a|+|b|$) et en considérant que l'estimation est faite par rapport à l'image précédente à l'instant t-l, on peut écrire l'inégalité suivante:

$$\left|I(i,j,t)\right| - \left|I(i-d_x, j-d_y, t-1)\right| \leq \left|I(i,j,t) - I(i-d_x, j-d_y, t-1)\right| \qquad \text{(III.6)}$$

La relation (1.5.14) est équivalente aux deux inégalités suivantes:

$$\left|I(i,j,t)\right| - \left|I(i-d_x, j-d_y, t-1)\right| \le \left|I(i,j,t) - I(i-d_x, j-d_y, t-1)\right| \quad \text{(III.7)}$$

$$\left|I(i-d_x, j-d_y, t-1)\right| - \left|I(i,j,t)\right| \le \left|I(i,j,t) - I(i-d_x, j-d_y, t-1)\right| \quad \text{(III.8)}$$

où d_x et d_y sont les deux composantes du vecteur déplacement $d=(d_x.\ d_y)$ du pixel de coordonnées (i,j), avec: $-d_{max} \le d_x,\ d_y \le d_{max}$. En sommant (III.7) et (III.8), on obtient les relations:

$$\sum_{i=1}^{B}\sum_{j=1}^{B}\left|I(i,j,t)\right| - \sum_{i=1}^{B}\sum_{j=1}^{B}\left|I(i-d_x, j-d_y, t-1)\right| \le \sum_{i=1}^{B}\sum_{j=1}^{B}\left|I(i,j,t) - I(i-d_x, j-d_y, t-1)\right| \quad \text{(III.9)}$$

$$\sum_{i=1}^{B}\sum_{j=1}^{B}\left|I(i-d_x, j-d_y, t-1)\right| - \sum_{i=1}^{B}\sum_{j=1}^{B}\left|I(i,j,t)\right| \le \sum_{i=1}^{B}\sum_{j=1}^{B}\left|I(i,j,t) - I(i-d_x, j-d_y, t-1)\right| \quad \text{(III.10)}$$

qui sont équivalentes aux relations:

$$R - M(d_x, d_y) \le MAD(d_x, d_y) \quad \text{(III.11)}$$

$$M(d_x, d_y) - R \le MAD(d_x, d_y) \quad \text{(III.12)}$$

où R est la somme des intensités des pixels du bloc de référence, $M(d_x,d_y)$ est la somme des intensités des pixels du bloc candidat, de vecteur déplacement $d=(d_x,d_y)$, et $MAD(d_x,d_y)$ est la mesure de ressemblance (dans ce cas une mesure de différence) entre le bloc de référence et le bloc candidat de déplacement (d_x,d_y):

$$R = \sum_{i=1}^{B}\sum_{j=1}^{B}\left|I(i,j,t), M(d_x, d_y)\right| = \sum_{i=1}^{B}\sum_{j=1}^{B}\left|I(i-d_x, j-d_y, t-1)\right| \quad \text{(III.13)}$$

$$MAD(d_x, d_y) = \sum_{i=1}^{B}\sum_{j=1}^{B}\left|I(i,j,t) - I(i-d_x, j-d_y, t-1)\right| \quad \text{(III.14)}$$

(*MAD* = différence moyenne absolue).

En supposant qu'on dispose déjà de $MAD(m_x,m_y)$ calculée pour un bloc candidat avec le vecteur déplacement (m_x,m_y), à la suite on sera intéressés seulement par les candidats qui sont les "plus semblables", qui ont les vecteurs déplacement (d_x,d_y) et qui satisfont la relation:

$$MAD\ (d_x, d_y) \le MAD\ (m_x, m_y) \quad \text{(III.15)}$$

Ainsi, les inégalités (III.11) et (III.12) deviennent:

$$R - M\ (d_x, d_y) \le MAD\ (m_x, m_y) \quad \text{(III.16)}$$

$$M\ (d_x, d_y) - R \le MAD\ (m_x, m_y) \quad \text{(III.17)}$$

C'est à dire:

$$R - MAD(m_x, m_y) \le M(d_x, d_y) \le R + MAD(m_x, m_y) \quad \text{(III.18)}$$

Par conséquent, si on dispose d'une estimation initiale $MAD(m_x,m_y)$, on cherche seulement les

blocs pour lesquels le vecteur déplacement associé (d_x,d_y) satisfait l'inégalité (III.15), respectivement (III.18). L'efficacité de cette méthode dépend de la rapidité de calcul de la somme des intensités des pixels du chaque bloc et de qualité de l'estimation initiale $M(d_x,d_y)$.

Pour calculer rapidement la somme des intensités des chaque bloc $(M(d_x,d_y))$ on utilise une technique rapide en deux étapes. On suppose que la taille de l'image est $K \times L$. L'image est divisée en $(L-B)$ bandes-lignes, chaque bande-ligne contenant B lignes (figure III.6).

Étape 1: Pour la première bande-ligne, on calcule la somme des intensités des pixels de chaque colonne: C11, C12,... ,C1K. Le calcul nécessite K.(B-l) opérations. Pour la deuxième bande-ligne:

$$\begin{aligned} C_{21} &= C_{11} - I(1,1) + I(B+1,1), \\ C_{22} &= C_{12} - I(1,2) + I(B+1,2), \\ &\cdots\cdots \\ C_{2K} &= C_{1K} - I(1,K) + I(B+1,K). \end{aligned} \qquad \text{(III.19)}$$

Ce calcul nécessite 2K opérations. En utilisant le même principe, on obtient la somme des intensités des pixels pour toutes les bandes-lignes. Le nombre total de calculs nécessaire pour l'étape 1 est:

$K.(B-1)+2K.(L-B-1)$.

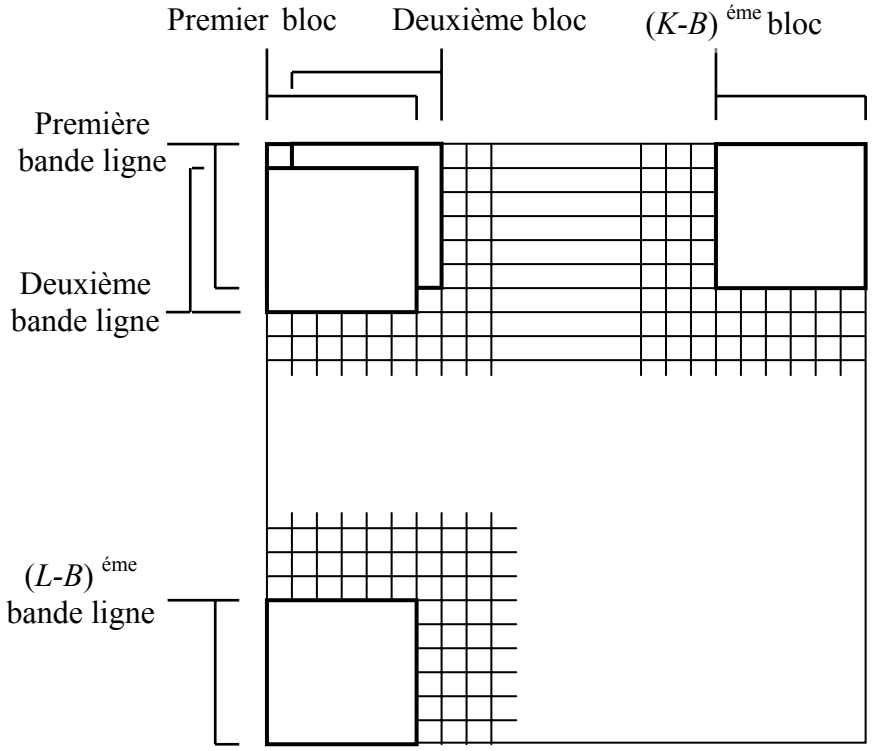


Figure III.6. Configuration de bandes-lignes pour l'algorithme d'élimination successive.

Étape 2 : Pour la première bande-ligne, la somme des intensités des pixels du premier bloc (SN_{11}) est obtenue en sommant C_{11}, C_{12}, ..., C_{1B}. Après cette opération, Le bloc est déplace

horizontalement d'un pixel et la norme de la somme du deuxième bloc (SN_{12}) est ($SN_{11} - C_{11} + C_{l(B+l)}$). De même manière on obtient SN_{13}, SN_{14}, ..., $SN_{l(K-B)}$. Le nombre de calculs pour une bande-ligne est (B-1) +2. (K-B-1). On peut calculer la somme des intensités des pixels du chaque bloc pour toutes les bandes-lignes. Le nombre total de calculs nécessaires pour l'étape 2 est:

(L-B).[(B-1)+2.(K-B-1)].

Le nombre total d'opérations nécessaires pour calculer la somme des intensités des pixels pour tous les blocs de l'image est:

$$\begin{aligned} NT &= K.(B-1) + 2K.(L-B-1) + (L-B).\left[(B-1) + 2.(K-B-1)\right] \\ &= 4.K.L - (L-B).(B+3) - 3.K.(B+1) \end{aligned} \qquad \text{(III.20)}$$

Le nombre total de blocs de référence est: NB=(K/B).(L/B), donc Le nombre total de calculs nécessaires pour chaque bloc de référence est:

$$G = NT / NB = [4 - (L-B).(B+3)/(K.L) - 3.(B+1)/L].B^2 \qquad \text{(III.21)}$$

En pratique, K et L sont beaucoup plus grands que B, et donc la partie droite de l'équation (III.21) est égale à $4B^2$. Par conséquent, dans le cas de l'algorithme d'élimination successive, les calculs sont nettement réduits (jusqu'à 15%) par rapport à la méthode classique de mise en correspondance de blocs, avec une recherche exhaustive .

Comme estimation initiale pour un bloc *($MAD(m_x,m_y)$)* on peut utiliser l'estimation du même bloc de l'image précédente. On doit remarquer aussi le fait que l'estimation initiale $MAD(m_x,m_y)$ associée à l'estimation initiale, ne reste pas constante pendant le processus de recherche. Cette estimation est actualisée si on trouve une meilleure estimation. Ainsi, à chaque pas de la recherche, si on trouve une $MAD(p_x,p_y)$ plus faible que $MAD(m_x,m_y)$, alors $MAD(m_x,m_y)$ est remplacé par $MAD(m_x,m_y)$. Ce processus va raffiner graduellement l'espace de recherche, d'où le nom d'élimination successive de l'algorithme.

Pour éviter les cas où *MAD* de la meilleure mise en correspondance est très grand (par exemple quand l'image change significativement) on peut remplacer l'estimation initiale $MAD(m_x,m_y)$ dans l'inéquation (III.18) avec un seuil prédéfini T. Ce seuil va spécifier que tous les blocs pour lesquels $M(d_x,d_y)$ est plus grand que ce seuil, sont considérés comme étant sans correspondant.

L'algorithme d'élimination successive à beaucoup d'applications, parmi lesquelles on peut mentionner la compression de séquences d'images dans le cadre du standard MPEG-4, l'algorithme pouvant être utilise dans sa version monorésolution mais aussi dans sa version multirésolution.

III.5.2. Estimation hiérarchique multirésolution du mouvement :

La représentation hiérarchique multirésolution d'images sous forme d'une pyramide Laplacienne ou en utilisant la transformation ondelette représente un des principes modernes utilisés en traitement et analyse d'images [17]. Ce principe peut être appliqué aussi dans l'estimation du mouvement et en particulier dans les méthodes d'estimation basées sur la mise en correspondance de blocs, pouvant conduire à une précision subpixelique.

Le principe de la représentation hiérarchique pyramidale (multirésolution) d'images est illustré sur la figure III.7. L'image initiale, de résolution maximale, est représentée à la base de la pyramide.

Les images des niveaux supérieurs sont des images d'une faible résolution, chaque image étant

obtenue à partir du niveau inférieur, par filtrage passe-bas, suivi d'un sous échantillonnage.

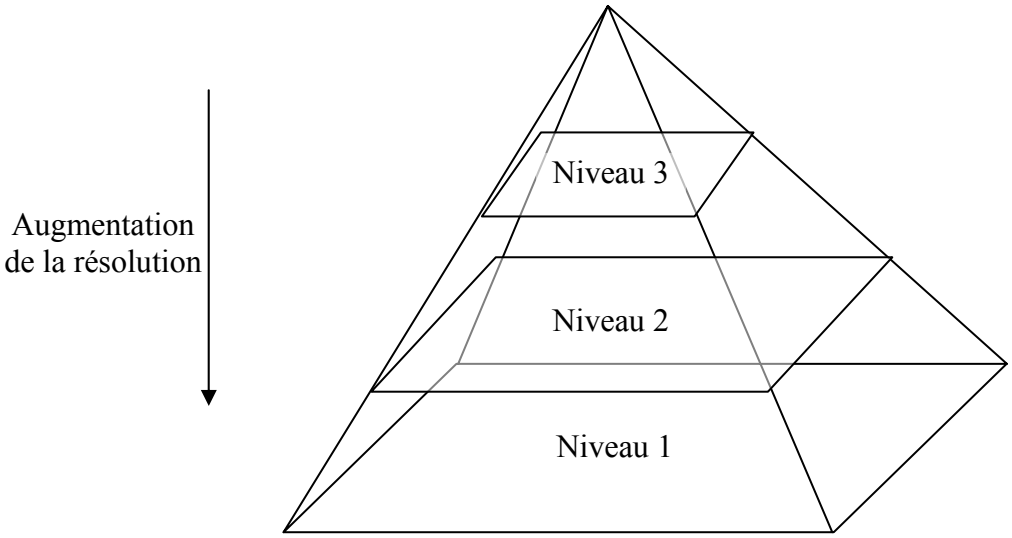


Figure III. 7. Représentation hiérarchique d'images.

L'idée de base de l'estimation hiérarchique du mouvement consiste à estimer successivement le mouvement à chaque niveau, en commençant par le niveau de résolution la plus faible, situé au sommet de la pyramide sur la figure III.7. L'estimation au niveau de résolution inférieure (niveau supérieur dans la représentation pyramidale) est transmise au niveau de résolution supérieure où est utilisée comme initialisation brute d'estimation du mouvement. Cette initialisation est alors raffinée par une méthode d'estimation de mouvement de paramètres correspondant au niveau respectif de résolution, le processus étant repris pour le niveau suivant (figure III.8).

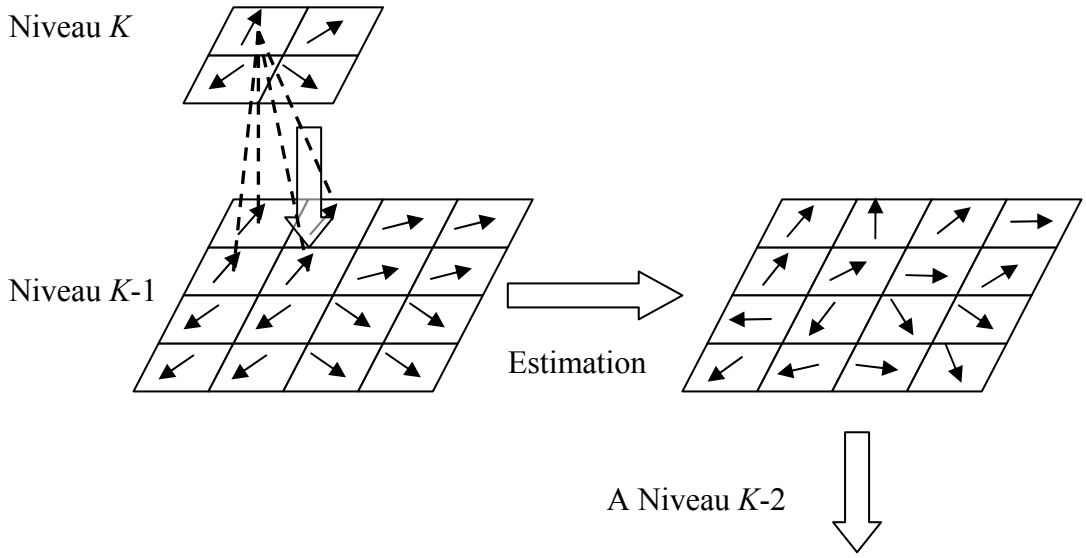


Figure III. 8. L'illustration du principe d'estimation hiérarchique (multirésolution) du mouvement.

La transmission de l'estimation finale d'un certain niveau au niveau inférieur (de résolution supérieure) peut être faite, par exemple, en utilisant le principe d'arbre quaternaire (figure III.8) éventuellement suivi d'une interpolation.

Le principe décrit est un principe général, pouvant être appliqué pour tous les types de traitement d'images. Dans le cadre de l'estimation du mouvement, ce principe peut s'appliquer pour toutes les méthodes. Ce principe a déjà été décrit sous une autre forme (paragraphe II.4.3.2).

Dans la suite, on illustre le principe d'estimation hiérarchique (multirésolution) du mouvement pour des méthodes de mise en correspondance de blocs, au niveau de résolution le plus faible, ainsi que pour raffiner l'estimation à chaque niveau. Dans ce cas, on peut utiliser des fenêtres de recherche relativement petites aux niveaux supérieurs, combinées à n'importe quelle stratégie de recherche (exhaustive, en n-pas, 2DLog, orthogonale, d'élimination successive). Le principe de mise en correspondance hiérarchique de blocs est illustré sur la figure III.9, dans le cas de la recherche en 3 pas.

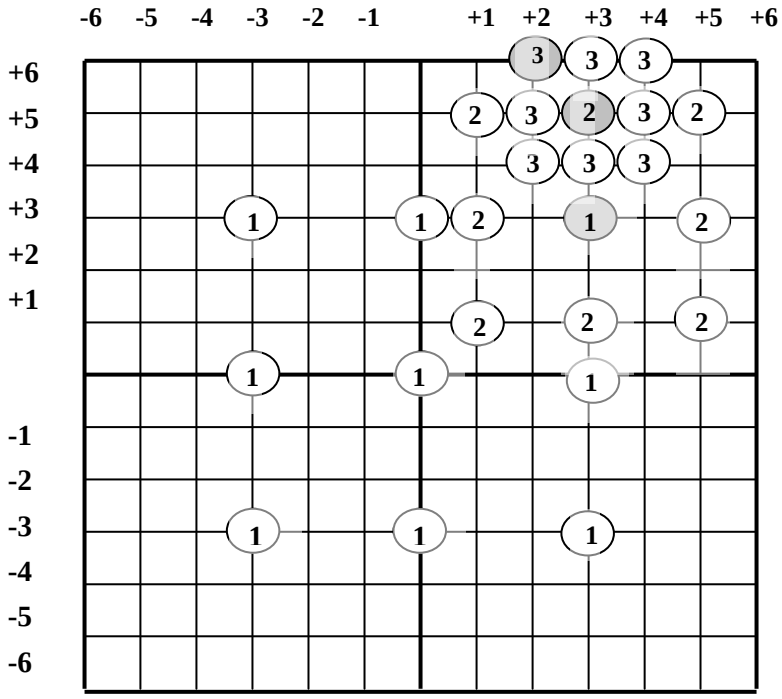


Figure III. 9. Illustration de la mise en correspondance hiérarchique de blocs, en utilisant la recherche en trois pas.

Dans cet exemple, le niveau de résolution inférieur (niveau 2) peut être obtenu par sous échantillonnage d'un facteur 2 du niveau précédent (niveau 1). Ainsi, un déplacement de 1 pixel au 2, correspond à un déplacement de 2 pixels au niveau 1. Cela veut dire que la fenêtre de recherche est 4 fois plus petite au niveau inférieur. L'estimation finale (la ressemblance optimale) obtenue au niveau de résolution courant notée "3" encerclé (figure III.9), est transmise au niveau de résolution supérieure (niveau 1), comme centre, noté "0", de la fenêtre de recherche à ce niveau.

En pratique, l'étape de sous échantillonnage peut ne pas être utilisée. Dans ce cas, la "pyramide" contient des images de mêmes dimensions, mais de plus en plus "floues", à cause de filtrage passe-bas. Des blocs de dimensions plus grandes (figure III.10) sont appliques aux images plus floues. Le balayage multirésolution peut être vu aussi comme une estimation du mouvement sur une grille grossière suivie d'un raffinement progressif de la grille vers la solution (figure III.10).

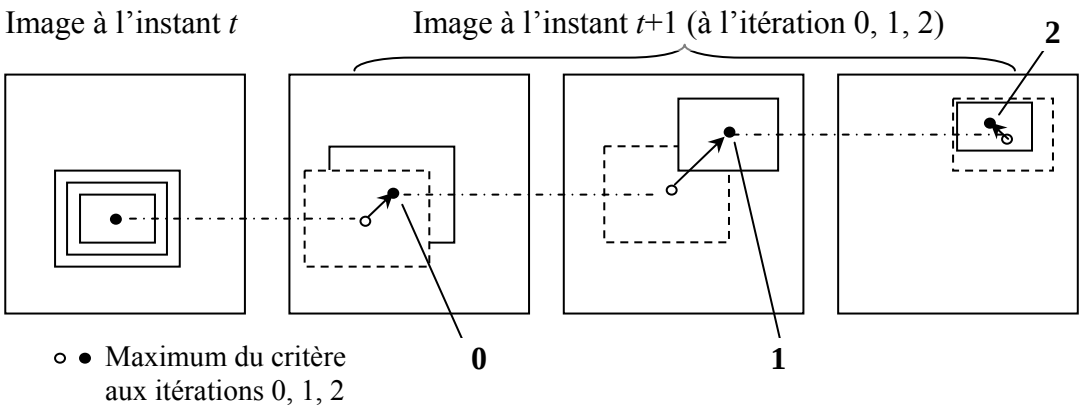


Figure III. 10. Illustration du principe de mise en correspondance hiérarchique multirésolution de blocs.

La mise en correspondance hiérarchique des blocs peut être appliquée dans le cas des vidéo-codeurs avec "scalabilité" spatiale, comme par exemple MPEG-2 ou H.263.

III.5.3. Mise en correspondance de blocs déformables :

Les méthodes de mise en correspondance de blocs de dimensions fixes donnent de bons résultats dans le cas d'un modèle de translation homogène du mouvement au sein de chaque bloc, mais donnent des résultats insatisfaisants pour des rotations, zoom ou des mouvements plus complexes. En présence de discontinuités dans le champ de mouvement, quand les contours des objets en mouvement ne coïncident pas avec les frontières photométriques de blocs, ces méthodes donnent une mauvaise prédiction et des "effets de bloc".

Cet effet diminue avec la dimension des blocs de même que la quantité d'information contenue dans un bloc, qui peut devenir insuffisante dans les zones homogènes de mouvement ou des niveaux de gris. Une solution consiste à choisir des blocs de grande dimension dans les zones stationnaires et des dimensions plus réduites lors de mouvements complexes.

Il en résulte une méthode d'estimation du mouvement basée sur la mise en correspondance de blocs de dimensions variables localement adaptative.

Dans les méthodes de mise en correspondance de blocs déformables, l'image courante est divisée en blocs de forme rectangulaire, triangulaire ou arbitraire et on cherche dans l'image cible le bloc qui ressemble le mieux au polygone de référence (figure III.11), pour une transformation spatiale donnée.

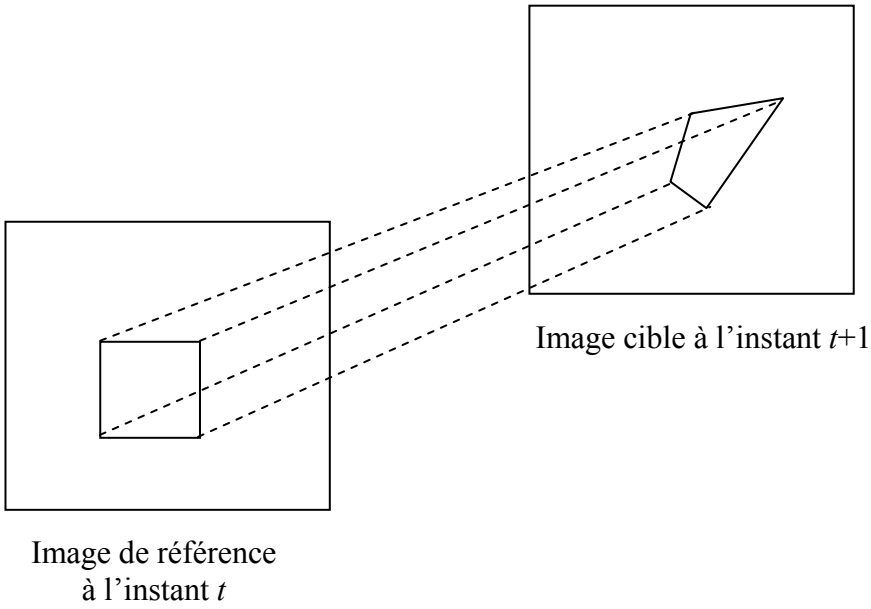


Figure III. 11. Illustration du principe de mise en correspondance de blocs déformables.

Le choix de la forme du bloc et de la transformation spatiale sont interdépendants. La forme triangulaire offre les degrés de liberté suffisants (2 équations par sommet) dans le cas d'une transformation affine (6 paramètres indépendants). Les transformations bilinéaires et projectives ont 8 paramètres indépendants et par conséquent, sont bien adaptées aux formes rectangulaires des blocs (figure III.6).

Par rapport aux deux paramètres dans le cas du modèle de translation, une transformation spatiale affine décrite par l'équation (II.11), bilinéaire (II.13) ou projective (II.12), permet une meilleure estimation de mouvement complexe (rotation, zoom etc.), mais augmente la complexité d'estimation: on doit déterminer 6 paramètres dans le cas d'une transformation affine, respectivement 8 paramètres dans le cas d'une transformation bilinéaire ou projective. La résolution du problème peut s'effectuer selon plusieurs stratégies de recherche, commençant avec la recherche exhaustive et jusqu'aux stratégies plus évoluées et plus rapides, comme par exemple la recherche hexagonale.

Dans le cas de la recherche exhaustive, l'algorithme peut être résumé ainsi

1. Diviser l'image de référence en blocs quadrilatéraux (ou triangulaires);

2. Modifier les coordonnées des sommets d'un quadrilatère (ou d'un triangle) de l'image cible, à partir d'une estimation initiale;

3. Pour chaque quadrilatère (ou triangle), trouver les paramètres qui mettent en correspondance le mieux chaque quadrilatère (ou triangle) de I 'image cible avec Le quadrilatère de l'image de référence, en utilisant une transformation spatiale donnée, des sommets;

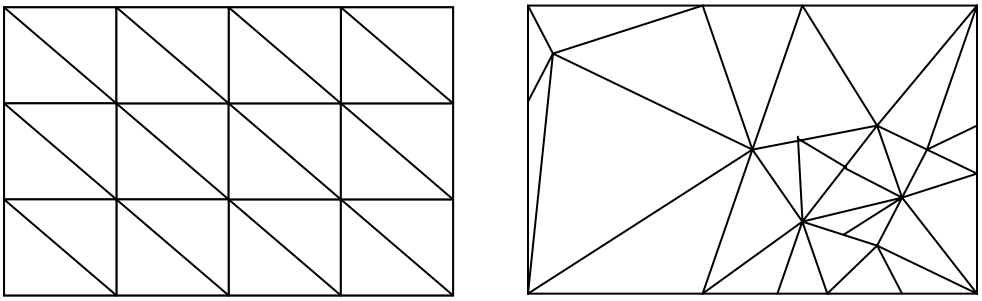
4. Trouver les paramètres de chaque pixel du quadrilatère (ou du triangle) en utilisant la transformation spatiale donnée et en calculant la ressemblance (critère a définir) entre le bloc de référence et le bloc de comparaison;

5. Optimiser les paramètres de la transformation qui conduisent à la meilleure ressemblance.

Pour réduire la complexité de l'estimation du mouvement, la mise en correspondance de blocs

déformables peut être réservée aux zones dans lesquelles la mise en correspondance standard n'offre pas de résultats satisfaisants. Ces zones peuvent être déterminées par exemple, en utilisant la différence entre les images déplacées (DFD) ou d'autres critères similaires, comme par exemple le critère de l'algorithme d'élimination successive (SEA), décrit dans le paragraphe III.5.1. Le temps de calcul peut être réduit aussi par l'utilisation de stratégies plus sophistiquées.

Le but des méthodes de mise en correspondance de blocs de dimensions variables est d'établir une correspondance entre tous les pixels des blocs (triangulaires ou quadrilatéraux) en s'appuyant sur les correspondances établies entre les sommets des blocs respectifs. Il est important que l'image de référence soit divisée en blocs contenant les pixels d'un seul objet en mouvement. Ce principe est celui des méthodes de segmentation d'images utilisant l'information de mouvement, dans des grilles adaptatives. Sur la figure III.12 est illustrée la différence entre les grilles régulières et les grilles adaptatives.



(a) (b)

Figure III. 12. Exemple de grille: (a) - régulière, (b) - adaptative.

Dans le cadre des méthodes de mise en correspondance de blocs de dimensions variables, les processus de segmentation d'image et d'estimation du mouvement sont des processus qui peuvent succéder ou être simultanés, parce qu'ils sont interdépendants. Dans ces méthodes on suivi la réalisation des grilles dont les arêtes se superposent le mieux possible sur les arêtes de niveaux de gris ou du mouvement. Dans les approches hiérarchiques on commence par une grille initiale brute (résolution faible) qui est alors raffinée (résolution plus élevée). L'estimation de mouvement par la mise en correspondance de grilles adaptatives est un des domaines ouverts en traitement d'images.

En plus de ces modèles, on peut utiliser des méthodes de mise en correspondance de formes arbitraires de mise en correspondance de contours.

III.6. Les algorithmes étudiées :

III.6.1. L'algorithme "Full search" FS:

La technique de recherche complète FS (***Full Search*** en anglais) consiste à faire la recherche du bloc le plus correspondant de l'image actuelle dans l'image précédente [28], [37]. Cet algorithme est le plus simple. Il est aussi le moins performant car il n'optimise pas la recherche : il n'y a pas de seuil pour la fenêtre de recherche. Il sert néanmoins de référence pour les autres algorithmes.

Pour une image de taille $P \times Q$ avec une fenêtre de recherche de taille $(2.W+1) \times (2.W+1)$, un taux d'images de T frames/seconde et des blocs de taille $N \times N$, le nombre d'instruction processeur par seconde est :

$$T \times (P \times Q/N^2) \times (2.W+1)^2 \times (2.N^2-1)$$

Cette méthode est optimal car elle teste tous les blocs possibles, mais nécessite énormément de calculs (c'est l'inconvénient majeur de cet algorithme) [41].

Il est possible de réduire les calculs de cette méthode en rejetant la plupart des blocs : l'inégalité triangulaire montre que si la différence entre les luminances moyenne des deux blocs est moins bonne que le minimum actuel, il n'est pas nécessaire de calculer le critère. L'erreur sera de toute façon supérieure. On peut donc ne pas traiter la plupart des blocs.

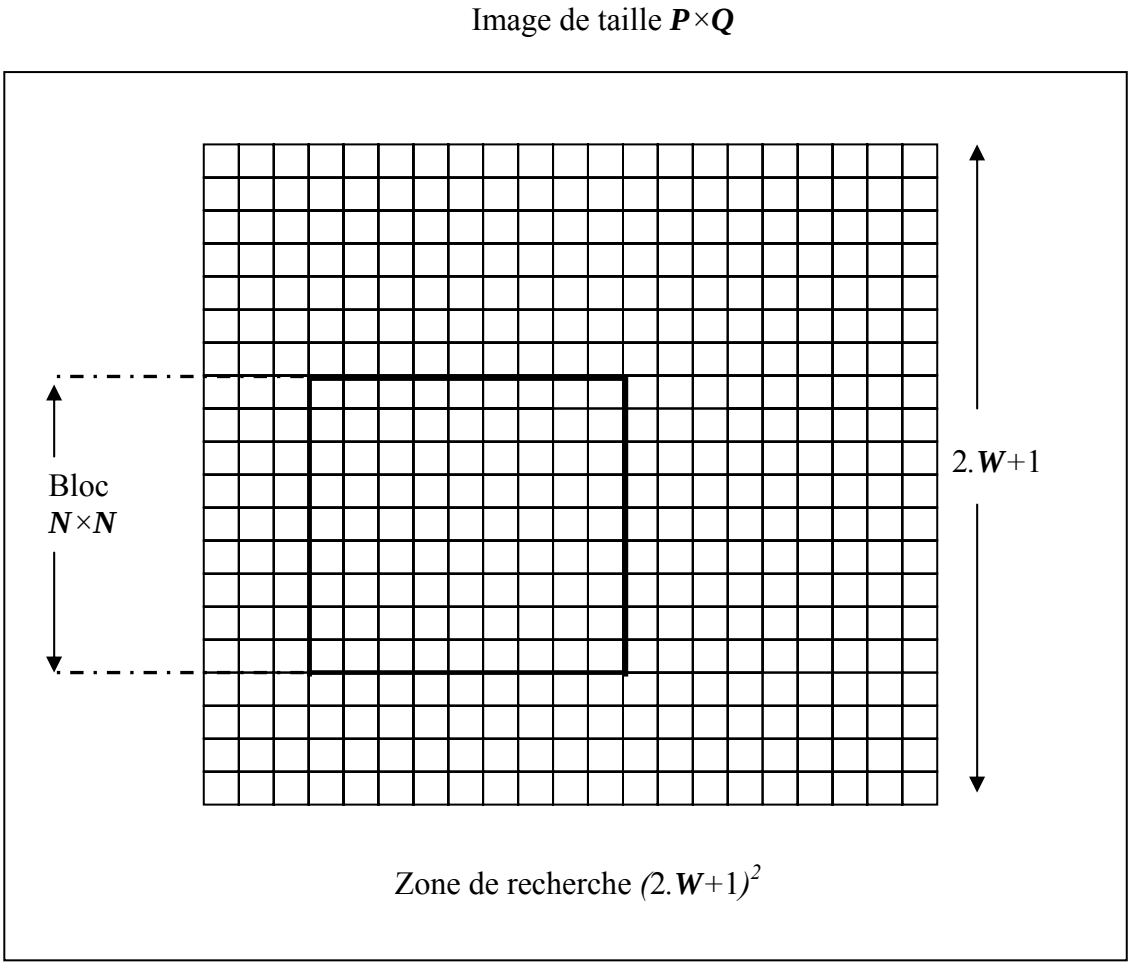


Figure III. 13. Modèle de recherche "Full Search" FS

III.6.2. L'algorithme à trois étapes de recherche TSS:

L'algorithme à trios étapes de recherche (***T**hree **S**tep **S**earch* en anglais) est l'algorithme le plus connu. C'est une technique de mise en correspondance de blocs dans le plan image et il utilise au maximum (03) étapes. Il est très utilisé pour l'estimation de mouvement dans les applications de la compression vidéo à bas débit binaire comme la téléphonie-visuelle et la vidéo-conférence. Le TSS est un modèle simple et effectif utilisé pour l'estimation de mouvement. Comparé avec EBMA il a un facteur d'économie plus grand que 100. Il à un nombre fixe de trois étapes de recherche et un nombre maximum et minimum de 25 points de recherche.

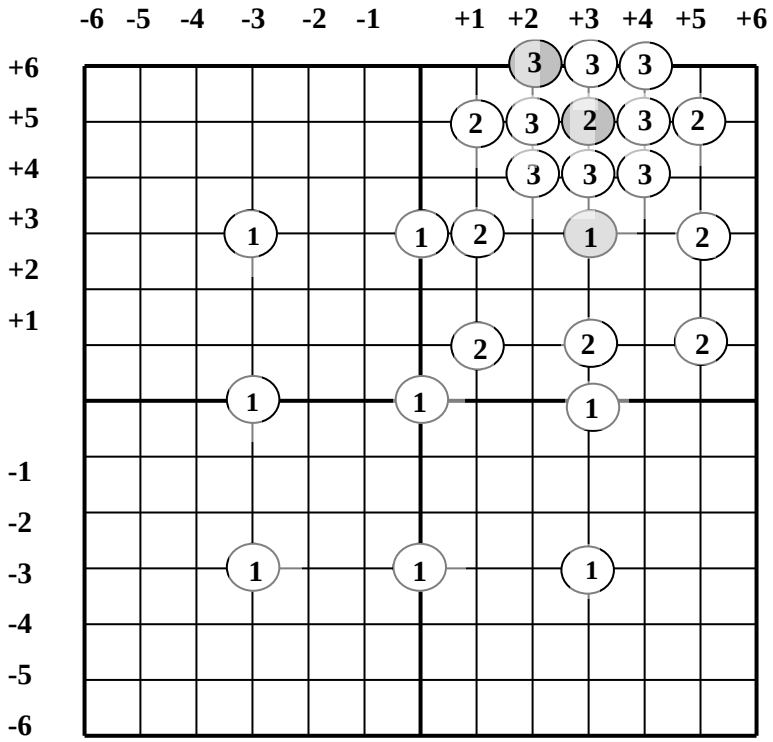


Figure III. 14. Modèle de recherche à trois étapes TSS.

La technique TSS commence avec une taille de l'étape plus large ou égale à la moitié de la gamme maximale. A la fin de recherche la taille de l'étape devient 1 pixel, comme suit [39]:

1. neuf points de recherche (le centre et huit points au frontière de la zone de recherche) sont comparés (son critère de minimum) dans la première étape, et après huit points seulement vont recherchés. (Les neuf points sont signés par le numéro (1) dans la figure III.14)

2. Au début de la nouvelle étape le point central de recherche va se déplacer vers le point le plus correspondant de l'étape précédente (le (1) en gris). Et la taille du carré de recherche vas se diminue en moitié après chaque étape.

3. A la fin de la recherche la taille de la recherche devient (01) un pixel.

Donc la technique NTSS est un simple algorithme pour l'estimation de mouvement spécialement utilisé pour l'implémentation dû à sa structure régulière. Mais il n'est pas efficace dans l'estimation du mouvement des objets stationnaires ou quasi-stationnaires qui est requis dans la compression vidéo.

III.6.3. Le nouvel algorithme à trois étapes de recherche NTSS :

L'algorithme à quatre étapes de recherche (***N**ew **T**hree **S**tep **S**earch* en anglais) est une modification de l'algorithme le bien connu TSS, qui est largement utilisé dans quelques algorithmes de compression vidéo a bas débit binaire. Cependant, comme mentionné dans le section précédente, l'algorithme TSS assignes des points de recherche uniformes dans la première étape. Par conséquent, il n'est efficace dans la détection des petits mouvements apparents dans les blocs stationnaires ou quasi stationnaires. Puisque la plupart des séquences sont lisses et doux, la détection des petits mouvements devient très importante. Une solution à ceci a pu être de rendre la recherche adaptable à l'incertitude et la variation du mouvement. Ceci est réalisé par l'emploi d'un modèle a point de contrôle à centre compensé dans la première étape. L'augmentation de la complexité due à l'addition

des points dans la première étape est compensée en ayant une technique d'arrêt à mi-chemin. Ceci fait la complexité de l'algorithme NTSS comparable à celle de l'algorithme TSS.

L'algorithme NTSS est comme suit. Comme mentionné dessus, NTSS diffère de TSS par l'utilisation d'un modèle a point de contrôle à centre compensé dans la première étape et par l'incorporation d'une technique d'arrêt à mi-chemin pour les blocs stationnaires ou quasi-stationnaires.

Les détails sont comme suit [29]:

1. en plus des points de contrôle utilisés en TSS, huit points supplémentaires sont ajoutés, qui sont les huit voisins supplémentaires de la fenêtre de recherche (centre-polarisé) (les cercles remplis en noir et les carrés dans la figure III.16).

2. une technique d'arrêt à mi-chemin est employée pour les blocs stationnaires et quasi-stationnaires afin d'identifier rapidement et aussi estimer les mouvements pour ces blocs.

a. Si le minimum *BDM* dans la première étape se trouve au centre de fenêtre de recherche, arrêter la recherche (arrêt à la première étape).

b. Si le point du minimum *BDM* dans la première étape est l'un des huit voisins du centre de fenêtre, la recherche dans la deuxième étape sera exécutée seulement pour les huit points voisins du minimum et arrêter alors la recherche (arrêt à la deuxième étape).(les triangles dans la Figure III.16).

Le diagramme bloc montré dans la figure III.15 montre les étapes impliquées dans l'algorithme NTSS.

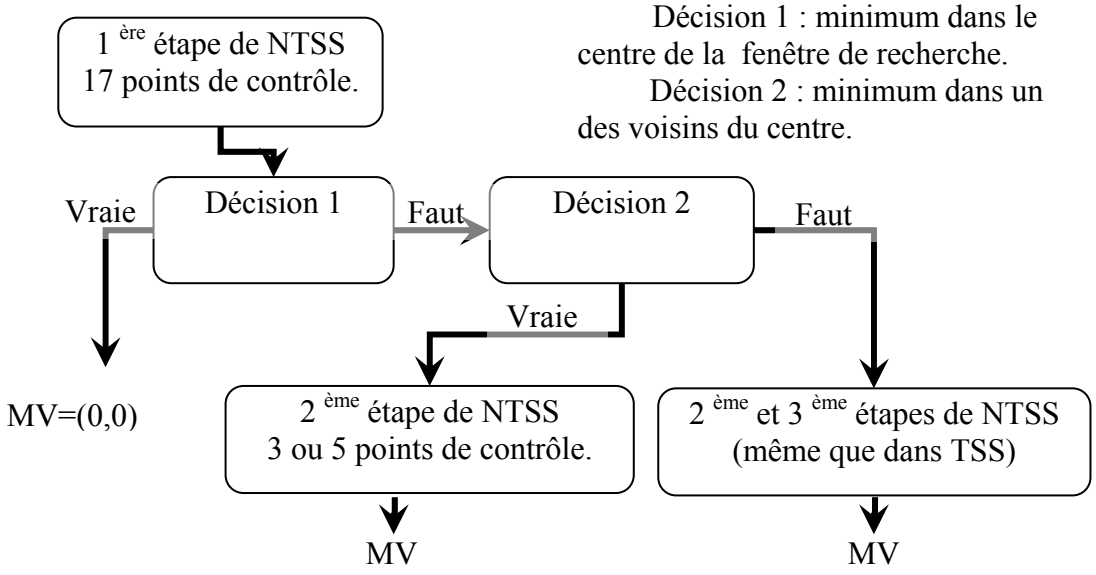


Figure III. 15. Diagramme bloc de l'algorithme NTSS.

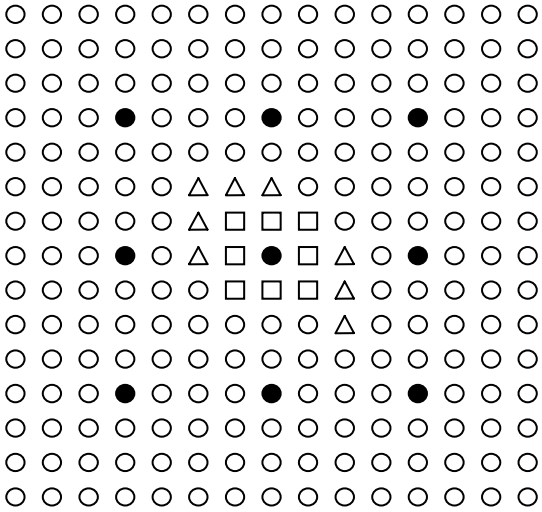


Figure III. 16. Région de recherche de l'algorithme NTSS.

III.6.4. L'algorithme à quatre étapes de recherche FSS:

L'algorithme à quatre étapes de recherche (***F**our **S**tep **S**earch* en anglais) à une meilleure performance par rapport à celle de TSS et une performance similaire par rapport à celle de NTSS. En plus, FSS réduit le nombre des points de recherche de 33 à 27 points [23].

Comme le NTSS, FSS a également un centre compensé et la technique d'arrêt à mi-chemin. Avec un modèle de recherche de +-7, l'algorithme *FSS* utilise un modèle de recherche à centre compensé avec neuf points de contrôle sur une fenêtre 5×5 dans la première étape, au lieu d'une fenêtre 9×9 comme dans le *TSS*. Le centre de la fenêtre est aussi décalé vers le point minimum BDM.

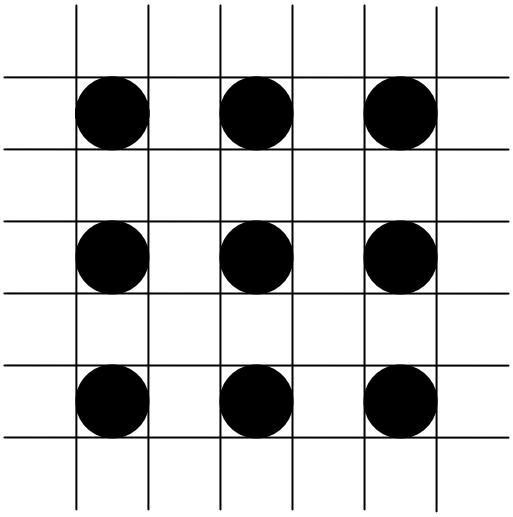


Figure III. 17.(a). Modèle de recherche 'première étape'.

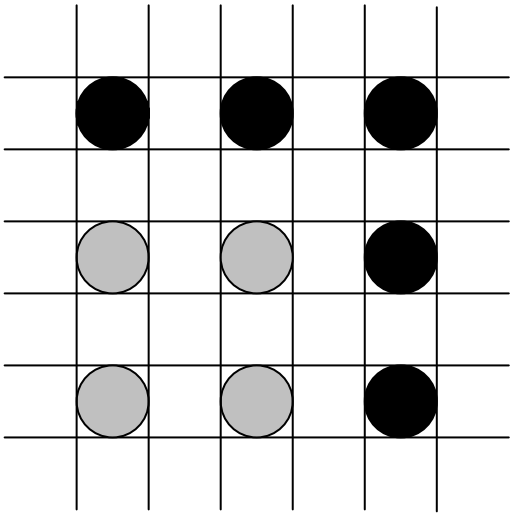


Figure III. 17.(b). Modèle de recherche 'deuxième/troisième étape'.

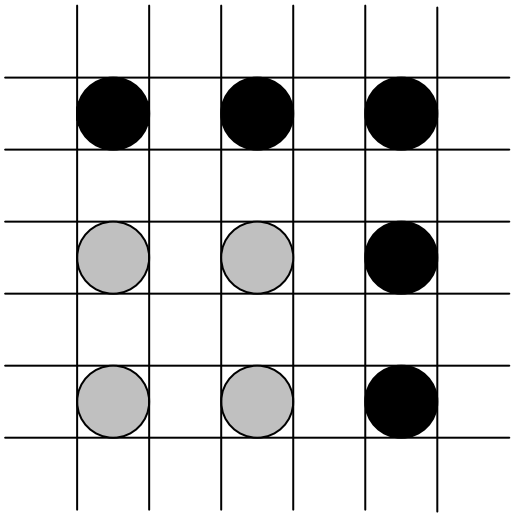


Figure III. 17.(c). Modèle de recherche ‘deuxième/troisième étape’.

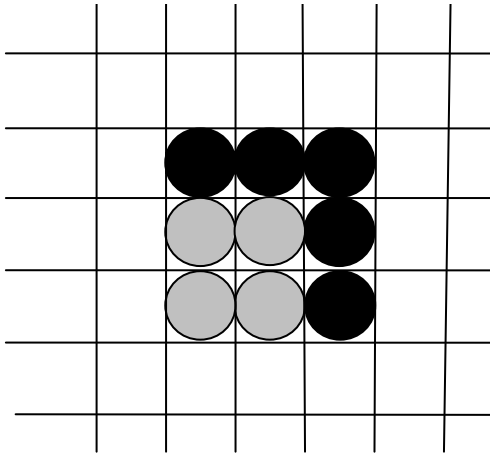


Figure III. 17.(d). Modèle de recherche ‘quatrième étape’.

L’algorithme FSS peut être résumé comme suit [20]:

1. Recherche de minimum BDM parmi les neuf points de contrôle sur une fenêtre de 5×5 situé au centre de la fenêtre de recherche, comme mentionné dans la figure III.17.(a). Si le minimum BDM est trouvé au centre de la fenêtre de recherche, alors aller a l’étape 4 sinon passer à l’étape 2.

2. La taille de fenêtre de recherche est maintenue à 5×5. Cependant le modèle de recherche dépendra de la position du minimum *BDM* précédent.

a. Si le point minimum *BDM* précédent est situé au coin de la fenêtre de recherche précédente, cinq points de contrôle additionnels montrés dans la figure III.17.(b) sont employés.

b. Si le point minimum *BDM* précédent est situé au milieu de l'axe horizontal ou vertical de la fenêtre de recherche précédente, trois points de contrôle additionnels comme montrés dans la figure III.17.(c) sont employés.

c. Si le minimum *BDM* est trouvé au centre de la fenêtre de recherche, passer à l'étape 4 autrement, passer à l'étape 3.

3. La stratégie du modèle de recherche est identique à celle de l’étape 2, mais passera finalement à l’étape 4.

4. La fenêtre de recherche est réduite à 3×3 comme montré dans FigureIII.17.(d) et la direction du vecteur global de mouvement est considéré comme le point du minimum *BDM* parmi ces neuf points de recherche.

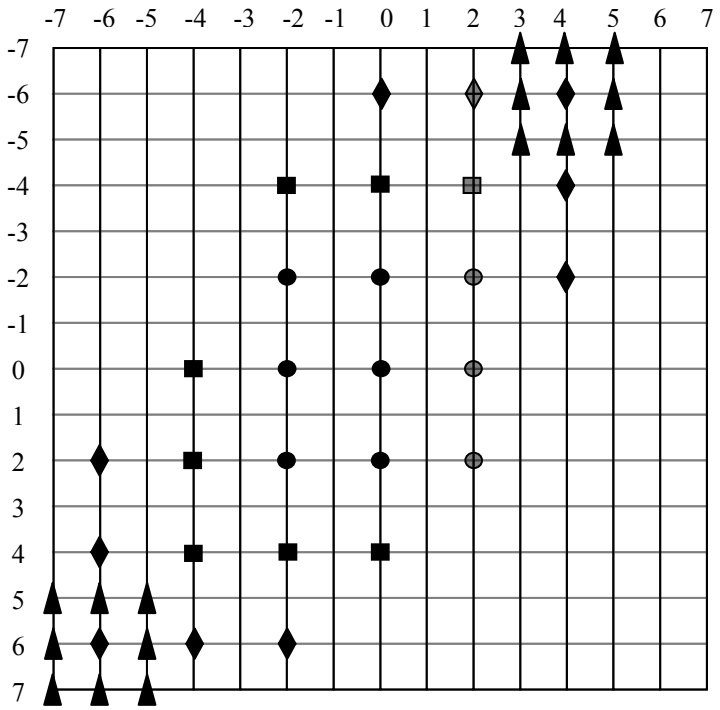


Figure III. 18. Modèle de recherche de l'algorithme FSS.

L'algorithme *FSS* est plus robuste par rapport aux algorithmes *TSS* et *NTSS*. C'est parce que l'utilisation de *FSS* est maintenue pour des séquences d'image qui contiennent le mouvement complexe de la caméra telle que le zoom et le mouvement rapide. En même temps, la régularité et la simplicité des dispositifs orientés par matériel sont également maintenues.

III.6.5. L'algorithme de recherche en diamant DS:

L'algorithme de recherche en diamant (***D****iamond* ***S****earch* en Anglais) est un algorithme rapide d'estimation de mouvement de correspondance de blocs, basé sur l'étude de la distribution des vecteurs de mouvement de plusieurs séquences vidéo généralement utilisées.

Les résultats de simulation prouvent que l'algorithme *DS* surpasse l'algorithme de *TSS*. Comparé avec l'algorithme *NTSS* , le *DS* exige 22% moins de calcul par rapport *NTSS*. La performance de cet algorithme est mieux que *FSS* en termes de nombre de points de recherche exigés.

Les résultats expérimentaux prouvent que 53% à 98% des vecteurs de mouvement sont joints dans un appui circulaire avec un radium de 2 *pixels* et portés sur la position du mouvement nul. En outre, le déplacement de bloc des séquences vidéo peut être dans n'importe quelle direction, mais principalement dans des directions horizontales et verticales. Basé sur ces deux observations, les points de recherche encourus dans le cercle avec du radium de 2 pixels sont les plus appropriés à choisir de composer le modèle de recherche. L'algorithme *DS* est fondé sur l'hypothèse que les vecteurs de mouvement sont en général de centre compensé.

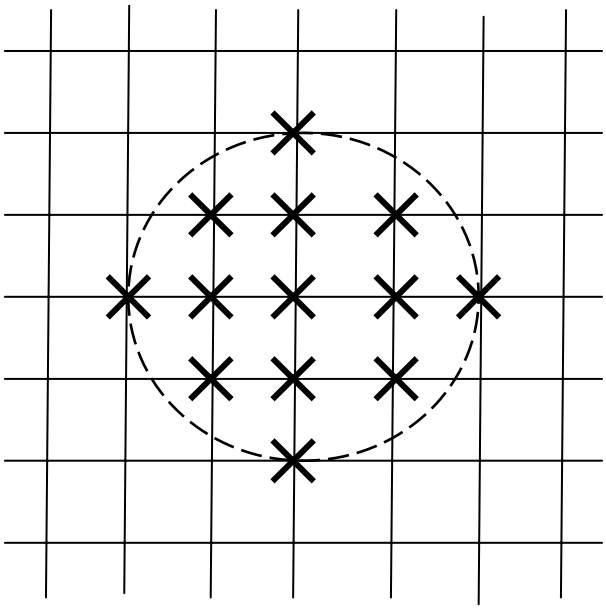
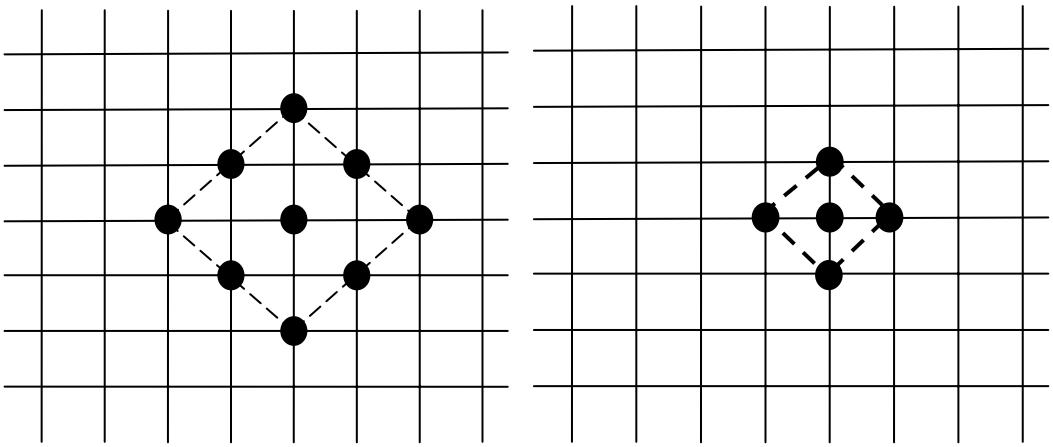


Figure III. 19. le modèle approprié de recherche, un secteur circulaire et radium de 2 pixels. Les 13 crois indiquent tous les points de contrôle possibles dans le cercle.

L'algorithme *DS* utilise deux modèles de recherche. Un modèle de recherche en grand diamant (***L**arge **D**iamond **S**earch **P**attern*) de 9 points de recherche comme montré sur la figureIII.20.(a), et un modèle de recherche en petit diamant (***S**mall **D**iamond **S**earch **P**attern*) de 5 points comme montré sur la Figure III.20.(b)



(a) **(b)**

FigureIII. 20. (a) Modèle de recherche en grand diamant **LDSP**. **(b)** Modèle de recherche en petit diamant **SDSP**.

Le *LDSP* est toujours employé dans la procédure de recherche jusqu'à l'étape dans laquelle le point minimum de la déformation de bloc (*MBD*) se produit au centre du diamant. Le modèle de recherche est alors passer à *SDSP*. Le point rapportant le *MBD* parmi les 5 points de contrôle dans le *SDSP* fournit le vecteur de mouvement du bloc le mieux correspondant.

L'algorithme DS peut se récapitulé comme suit [32]:

1. Le *LDSP* initial est centré à l'origine de la fenêtre de recherche, et les 9 points de contrôle de *LDSP* sont examinés. Si le point minimum *MBD* calculé est situé à la position centrale, passer à l'étape 3; autrement, passer à l'étape 2.

2. Le point minimum *MBD* trouvé dans l'étape de recherche précédente devient le point central du nouveau *LDSP*. Si le nouveau point minimum *MBD* obtenu est situé à la position centrale, passer à l'étape 3; autrement, répéter périodiquement cette étape.

3. Changer le modèle de recherche de *LDSP* à *SDSP*. Le point minimum *MBD* trouvé dans cette étape est la solution finale du vecteur de mouvement, donc c'est le bloc le plus correspondant.

III.6.6. L'algorithme de recherche minimum MIN :

L'algorithme de recherche minimum est un algorithme efficace pour la détection des petits mouvements, car il fait toujours un pas de (1) un pixels, donc pour les petits déplacements, il converge rapidement.

La stratégie de recherche est comme suit [18]:

1. Il y a cinq points de recherche dans la première étape, un point du centre de la fenêtre de recherche et quatre points voisins (avec une distance de 01 pixel entre chacun des quatre points et le point central) comme indiqué sur la figure III.21. Le minimum BDM sera le centre de la fenêtre de recherche suivante.
2. Dans la deuxième étape, il y a quatre points de recherche, le point central (minimum de l'étape précédente) et trois points voisins, avec l'élimination de point minimum de l'étape précédente.
3. Les autres étapes sont identiques à l'étape 2, est l'algorithme s'arrête et donne la position de bloc le plus correspondant lorsque la position des minimum des deux recherches successives sont les mêmes ou lorsqu'il touche la frontière de la zone de recherche.

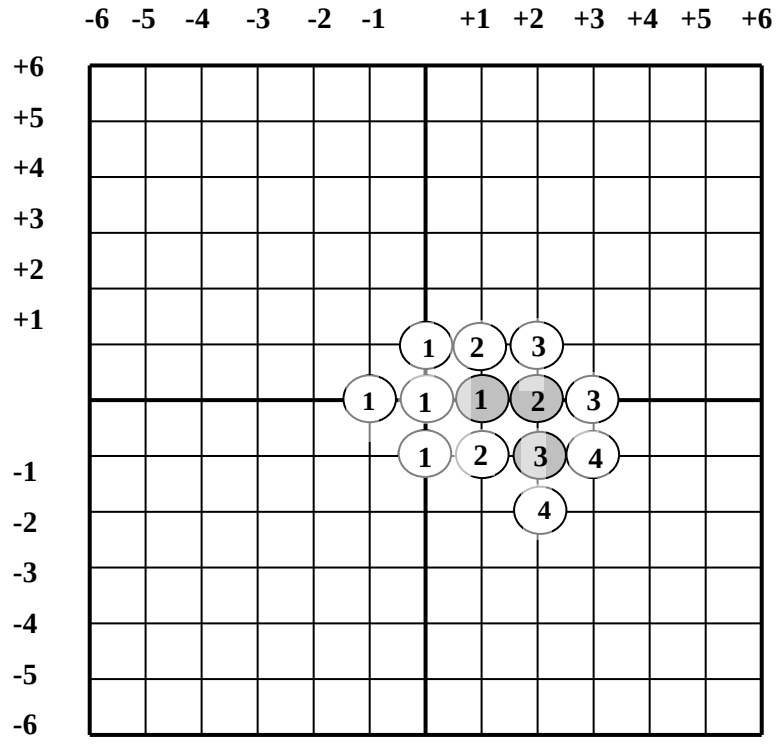


Figure III. 21. Méthode de recherche de l'algorithme minimum.

Chapitre IV :

Applications des Algorithmes et Résultats Pratiques

Chapitre IV :

Applications des Algorithmes et Résultats Pratiques

Dans ce chapitre final, on a choisi comme séquence vidéo (couleur) pour le test des six algorithmes ; trois séquences connues dans ce domaine de recherche, de format AVI, la première est 'Akiyo', la deuxième est 'Flower Garden', la troisième est 'Foreman', les dimensions de ces séquences sont 528 pixels × 352 pixels.

On vas discuter les résultats obtenues par les six algorithmes, sur les trois séquences vidéos, pour la séquence 'Akiyo' les frames entre n°20 et n°25, pour la séquence 'Flower Garden' les frames entre n°1 et n°6, et finalement pour la séquence 'Foreman' les frames entre n°30 et n° 35.

Par le langage de programmation MATLAB, les blocs utilisés dans les six algorithmes sont de taille 16 × 16 pixels, sur un PC Pentium III. 933 MHz et 256 Moctets comme RAM, on a traité six (06) frames successives pour chaque séquence, par la stratégie suivante :

- la frame n° 2 reconstruite, est reconstruites à partir de la frame n° 1 originale.
- la frame n° 3 reconstruite, est reconstruites à partir de la frame n° 2 reconstruite.
- la frame n° 4 reconstruite, est reconstruites à partir de la frame n° 3 reconstruite.
- la frame n° 5 reconstruite, est reconstruites à partir de la frame n° 4 reconstruite.
- la frame n° 6 reconstruite, est reconstruites à partir de la frame n° 5 reconstruite.

Les tableaux suivants nous montre les frames originaux de ces trois séquences dans la première colonne, les frames résultantes reconstruites (en utilisant les algorithmes étudiées) dans la deuxième colonne, et dans la toisième colonne les schémas qui nous donne les vecteurs de déplacements selon l'axe horizontal et selon l'axe vertical entre deux frames successives :

IV.1. Résultats de l'algorithme FS :

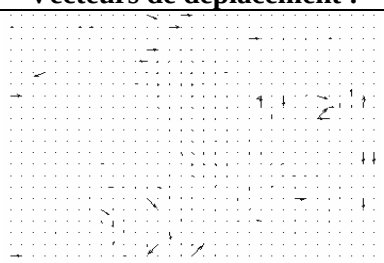
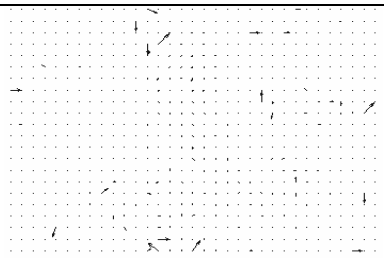
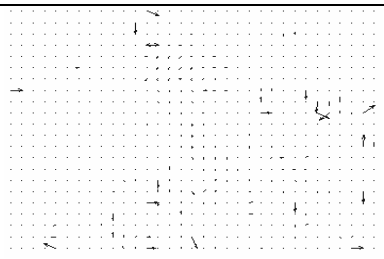
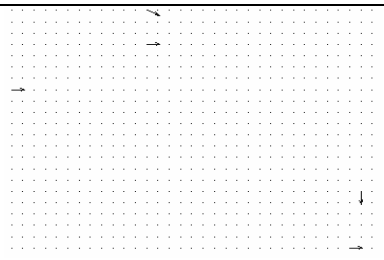
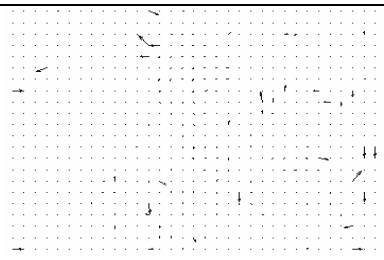
Frames originaux :	Frames reconstruites :	Vecteurs de déplacement :
 Frame n° 21	 Frame n° 21 reconstruite à partir de la frame n° 20 originale	 Entre les frames n° 20 et 21
 Frame n° 22	 Frame n° 22 reconstruite à partir de la frame n° 21 reconstruite	 Entre les frames n° 21 et 22
 Frame n° 23	 Frame n° 23 reconstruite à partir de la frame n° 22 reconstruite	 Entre les frames n° 22 et 23
 Frame n° 24	 Frame n° 24 reconstruite à partir de la frame n° 23 reconstruite	 Entre les frames n° 23 et 24
 Frame n° 25	 Frame n° 25 reconstruite à partir de la frame n° 24 reconstruite	 Entre les frames n° 24 et 25

Tableau 3 : résultats de l'algorithme FS sur la séquence 'Akiyo' (frames 20-25)

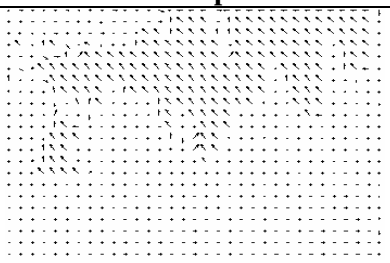
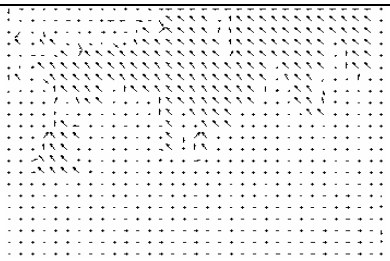
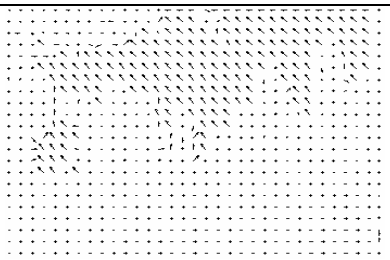
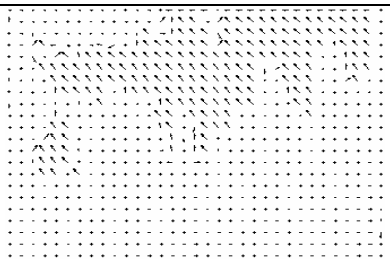
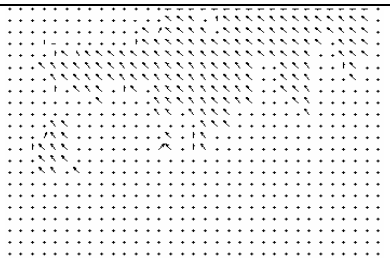
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 2	 Frame n° 2 reconstruite à partir de la frame n° 1 originale	 Entre les frames n° 1 et n° 2
 Frame n° 3	 Frame n° 3 reconstruite à partir de la frame n° 2 reconstruite	 Entre les frames n° 2 et n° 3
 Frame n° 4	 Frame n° 4 reconstruite à partir de la frame n° 3 reconstruite	 Entre les frames n° 3 et n° 4
 Frame n° 5	 Frame n° 5 reconstruite à partir de la frame n° 4 reconstruite	 Entre les frames n° 4 et n° 5
 Frame n° 6	 Frame n° 6 reconstruite à partir de la frame n° 5 reconstruite	 Entre les frames n° 5 et n° 6

Tableau 4: résultats de l'algorithme FS sur la séquence 'Flower Garden' (frames 1-6)

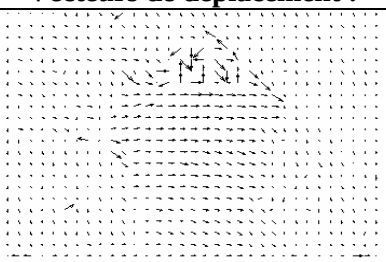
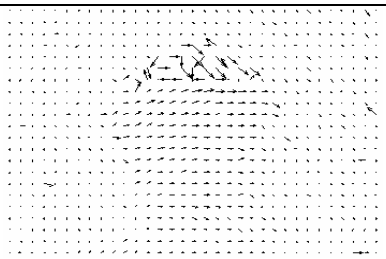
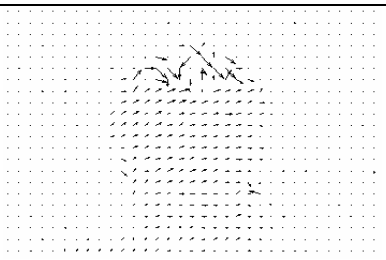
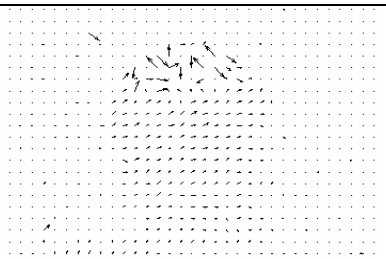
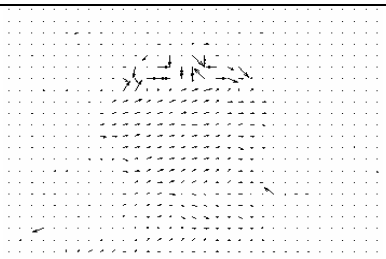
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 31	 Frame n° 31 reconstruite à partir de la frame n° 30 originale	 Entre les frames n° 30 et 31
 Frame n° 32	 Frame n° 32 reconstruite à partir de la frame n° 31 reconstruite	 Entre les frames n° 31 et 32
 Frame n° 33	 Frame n° 33 reconstruite à partir de la frame n° 32 reconstruite	 Entre les frames n° 32 et 33
 Frame n° 34	 Frame n° 34 reconstruite à partir de la frame n° 33 reconstruite	 Entre les frames n° 33 et 34
 Frame n° 35	 Frame n° 35 reconstruite à partir de la frame n° 34 reconstruite	 Entre les frames n° 34 et 35

Tableau 5: résultats de l'algorithme FS sur la séquence 'Foreman' (frames 30-35)

IV.2. Résultats de l'algorithme TSS :

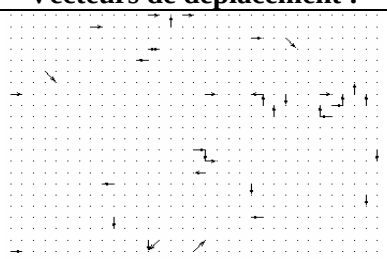
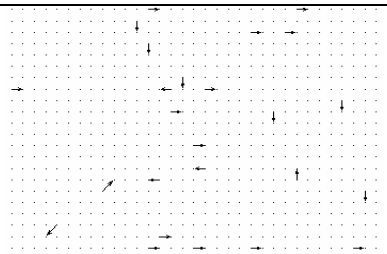
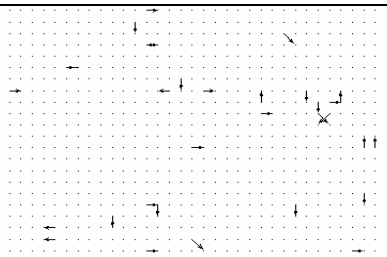
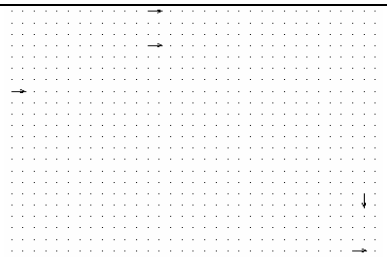
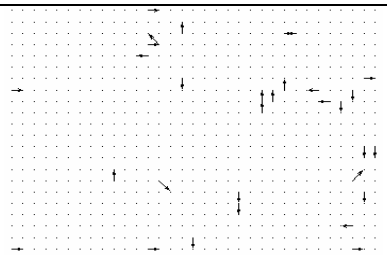
Frames originaux :	Frames reconstruites :	Vecteurs de déplacement :
 Frame n° 21	 Frame n° 21 reconstruite à partir de la frame n° 20 originale	 Entre les frames n° 20 et 21
 Frame n° 22	 Frame n° 22 reconstruite à partir de la frame n° 21 reconstruite	 Entre les frames n° 21 et 22
 Frame n° 23	 Frame n° 23 reconstruite à partir de la frame n° 22 reconstruite	 Entre les frames n° 22 et 23
 Frame n° 24	 Frame n° 24 reconstruite à partir de la frame n° 23 reconstruite	 Entre les frames n° 23 et 24
 Frame n° 25	 Frame n° 25 reconstruite à partir de la frame n° 24 reconstruite	 Entre les frames n° 24 et 25

Tableau 6: résultats de l'algorithme TSS sur la séquence 'Akiyo' (frames 20-25)

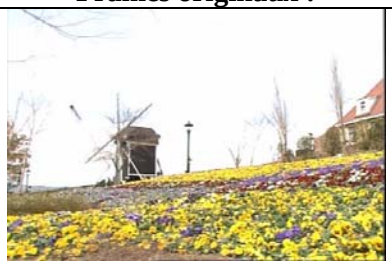
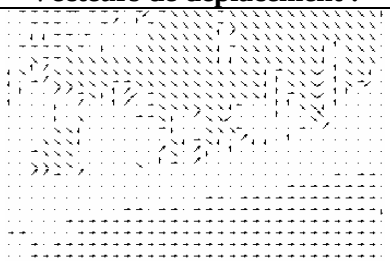
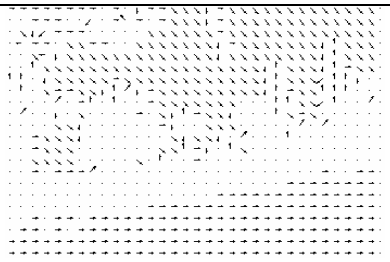
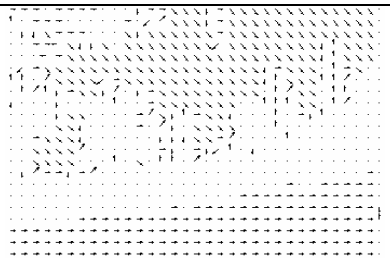
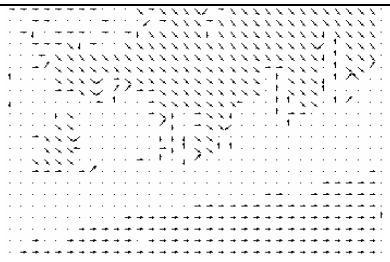
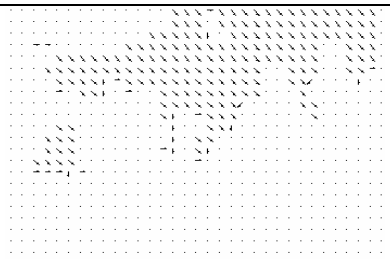
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 2	 Frame n° 2 reconstruite à partir de la frame n° 1 originale	 Entre les frames n° 1 et n° 2
 Frame n° 3	 Frame n° 3 reconstruite à partir de la frame n° 2 reconstruite	 Entre les frames n° 2 et n° 3
 Frame n° 4	 Frame n° 4 reconstruite à partir de la frame n° 3 reconstruite	 Entre les frames n°3 et n° 4
 Frame n° 5	 Frame n° 5 reconstruite à partir de la frame n° 4 reconstruite	 Entre les frames n° 4 et n° 5
 Frame n° 6	 Frame n° 6 reconstruite à partir de la frame n° 5 reconstruite	 Entre les frames n° 5 et n° 6

Tableau 7: résultats de l'algorithme TSS sur la séquence 'Flower Garden' (frames 1-6)

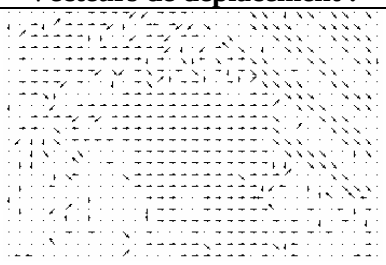
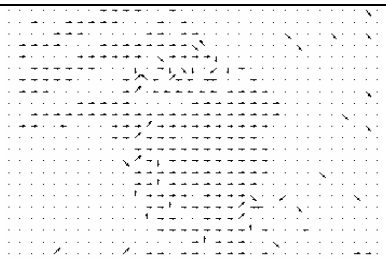
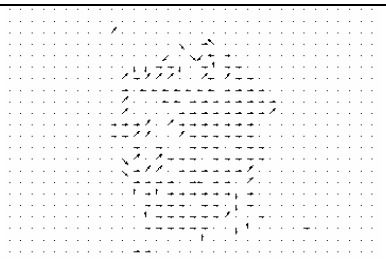
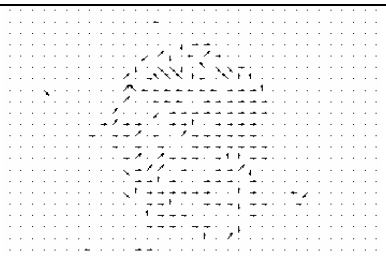
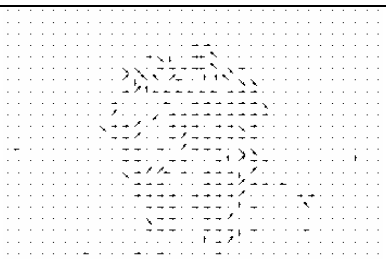
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 31	 Frame n° 31 reconstruite à partir de la frame n° 30 originale	 Entre les frames n° 30 et 31
 Frame n° 32	 Frame n° 32 reconstruite à partir de la frame n° 31 reconstruite	 Entre les frames n° 31 et 32
 Frame n° 33	 Frame n° 33 reconstruite à partir de la frame n° 32 reconstruite	 Entre les frames n° 32 et 33
 Frame n° 34	 Frame n° 34 reconstruite à partir de la frame n° 33 reconstruite	 Entre les frames n° 33 et 34
 Frame n° 35	 Frame n° 35 reconstruite à partir de la frame n° 34 reconstruite	 Entre les frames n° 34 et 35

Tableau 8: résultats de l'algorithme TSS sur la séquence 'Foreman' (frames 30-35)

IV.3. Résultats de l'algorithme NTSS :

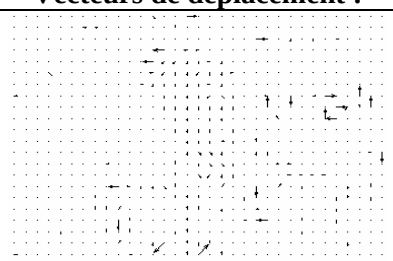
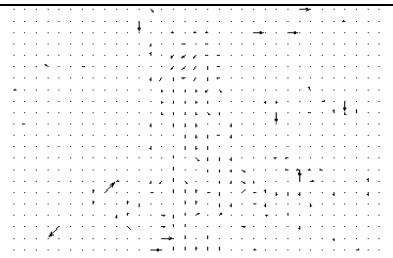
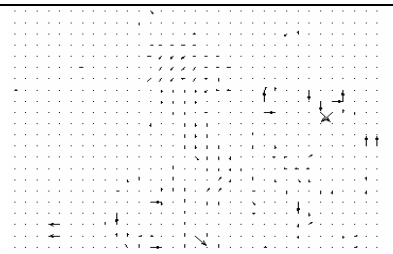
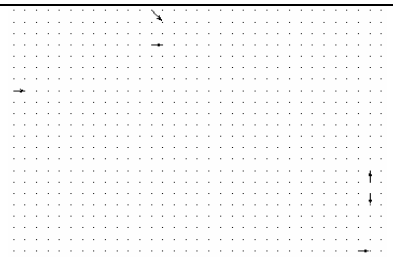
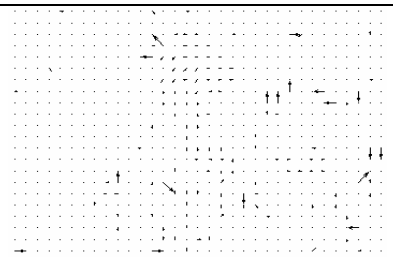
Frames originaux :	Frames reconstruites :	Vecteurs de déplacement :
 Frame n° 21	 Frame n° 21 reconstruite à partir de la frame n° 20 originale	 Entre les frames n° 20 et 21
 Frame n° 22	 Frame n° 22 reconstruite à partir de la frame n° 21 reconstruite	 Entre les frames n° 21 et 22
 Frame n° 23	 Frame n° 23 reconstruite à partir de la frame n° 22 reconstruite	 Entre les frames n° 22 et 23
 Frame n° 24	 Frame n° 24 reconstruite à partir de la frame n° 23 reconstruite	 Entre les frames n° 23 et 24
 Frame n° 25	 Frame n° 25 reconstruite à partir de la frame n° 24 reconstruite	 Entre les frames n° 24 et 25

Tableau 9: résultats de l'algorithme NTSS sur la séquence 'Akiyo' (frames 20-25)

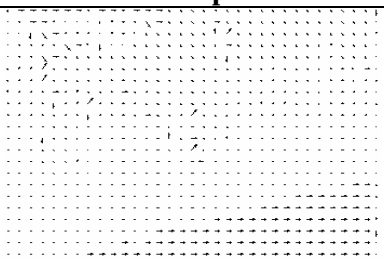
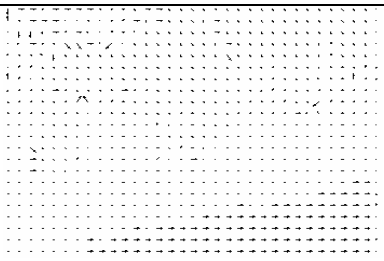
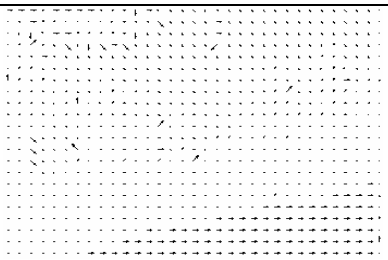
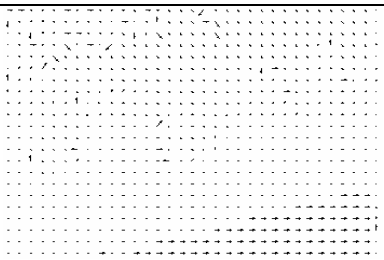
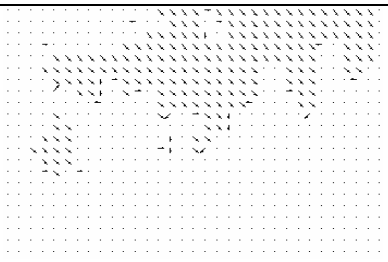
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 2	 Frame n° 2 reconstruite à partir de la frame n° 1 originale	 Entre les frames n° 1 et n° 2
 Frame n° 3	 Frame n° 3 reconstruite à partir de la frame n° 2 reconstruite	 Entre les frames n° 2 et n° 3
 Frame n° 4	 Frame n° 4 reconstruite à partir de la frame n° 3 reconstruite	 Entre les frames n°3 et n° 4
 Frame n° 5	 Frame n° 5 reconstruite à partir de la frame n° 4 reconstruite	 Entre les frames n° 4 et n° 5
 Frame n° 6	 Frame n° 6 reconstruite à partir de la frame n° 5 reconstruite	 Entre les frames n° 5 et n° 6

Tableau 10: résultats de l'algorithme NTSS sur la séquence 'Flower Garden' (frames 1-6)

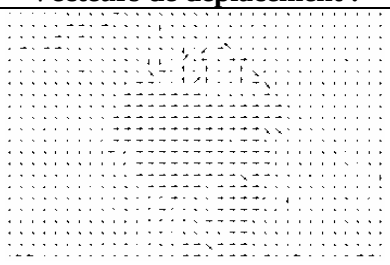
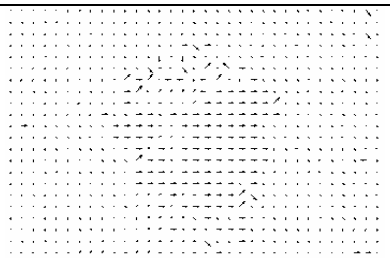
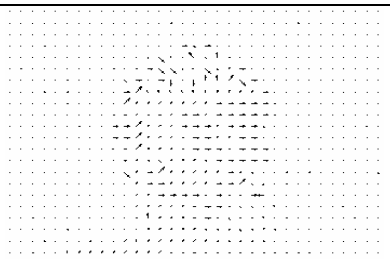
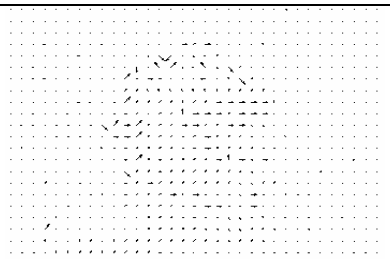
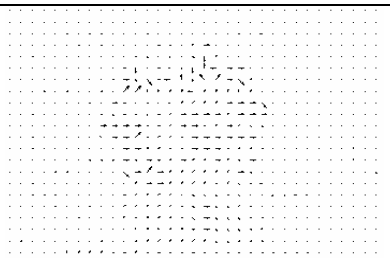
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 31	 Frame n° 31 reconstruite à partir de la frame n° 30 originale	 Entre les frames n° 30 et 31
 Frame n° 32	 Frame n° 32 reconstruite à partir de la frame n° 31 reconstruite	 Entre les frames n° 31 et 32
 Frame n° 33	 Frame n° 33 reconstruite à partir de la frame n° 32 reconstruite	 Entre les frames n° 32 et 33
 Frame n° 34	 Frame n° 34 reconstruite à partir de la frame n° 33 reconstruite	 Entre les frames n° 33 et 34
 Frame n° 35	 Frame n° 35 reconstruite à partir de la frame n° 34 reconstruite	 Entre les frames n° 34 et 35

Tableau 11: résultats de l'algorithme NTSS sur la séquence 'Foreman' (frames 30-35)

IV.4. Résultats de l'algorithme FSS :

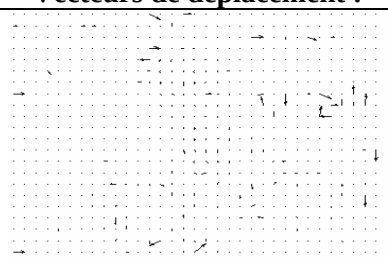
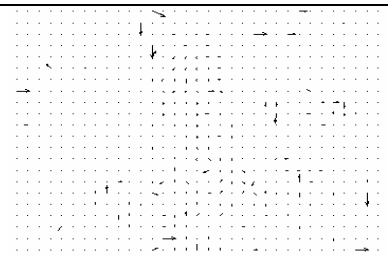
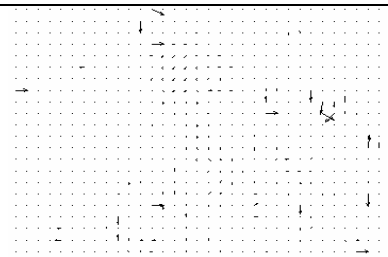
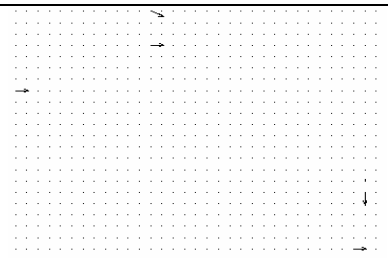
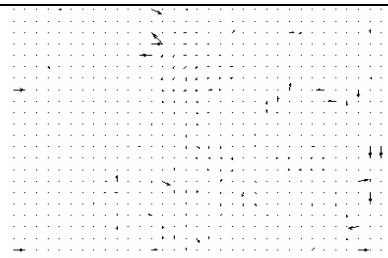
Frames originaux :	Frames reconstruites :	Vecteurs de déplacement :
 Frame n° 21	 Frame n° 21 reconstruite à partir de la frame n° 20 originale	 Entre les frames n° 20 et 21
 Frame n° 22	 Frame n° 22 reconstruite à partir de la frame n° 21 reconstruite	 Entre les frames n° 21 et 22
 Frame n° 23	 Frame n° 23 reconstruite à partir de la frame n° 22 reconstruite	 Entre les frames n° 22 et 23
 Frame n° 24	 Frame n° 24 reconstruite à partir de la frame n° 23 reconstruite	 Entre les frames n° 23 et 24
 Frame n° 25	 Frame n° 25 reconstruite à partir de la frame n° 24 reconstruite	 Entre les frames n° 24 et 25

Tableau 12: résultats de l'algorithme FSS sur la séquence 'Akiyo' (frames 20-25)

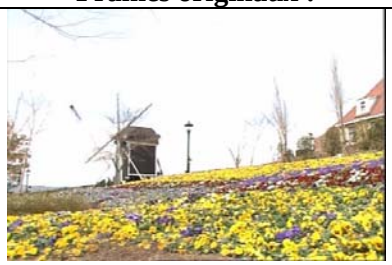
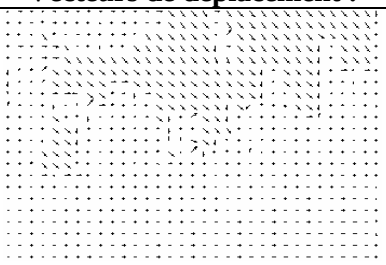
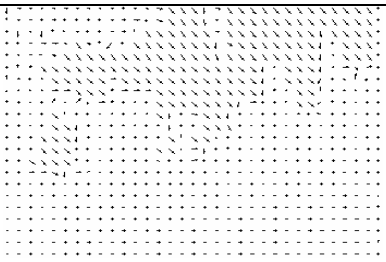
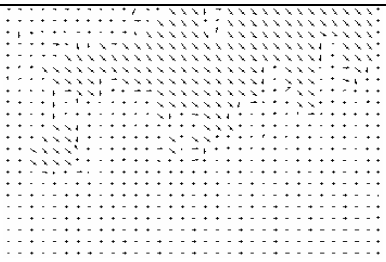
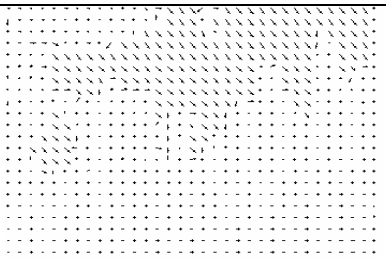
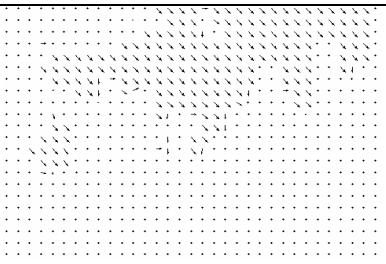
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 2	 Frame n° 2 reconstruite à partir de la frame n° 1 originale	 Entre les frames n° 1 et n° 2
 Frame n° 3	 Frame n° 3 reconstruite à partir de la frame n° 2 reconstruite	 Entre les frames n° 2 et n° 3
 Frame n° 4	 Frame n° 4 reconstruite à partir de la frame n° 3 reconstruite	 Entre les frames n° 3 et n° 4
 Frame n° 5	 Frame n° 5 reconstruite à partir de la frame n° 4 reconstruite	 Entre les frames n° 4 et n° 5
 Frame n° 6	 Frame n° 6 reconstruite à partir de la frame n° 5 reconstruite	 Entre les frames n° 5 et n° 6

Tableau 13: résultats de l'algorithme FSS sur la séquence 'Flower Garden' (frames 1-6)

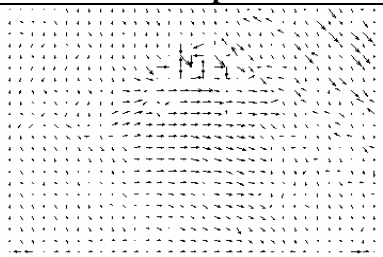
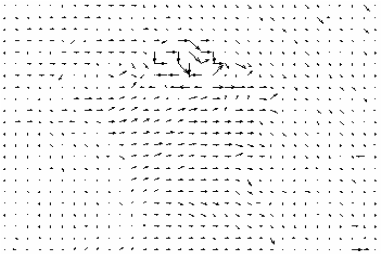
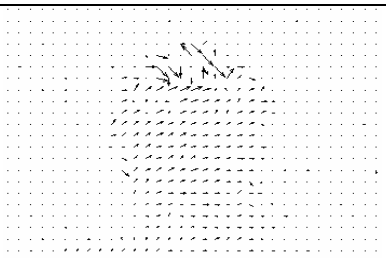
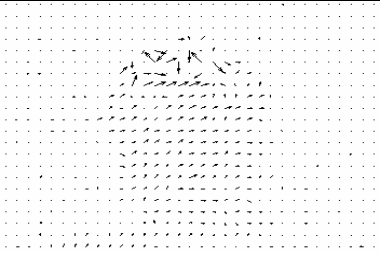
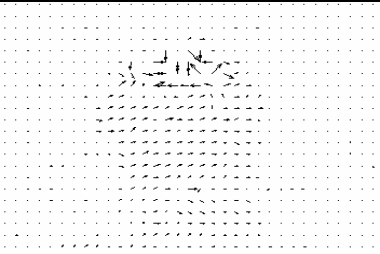
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 31	 Frame n° 31 reconstruite à partir de la frame n° 30 originale	 Entre les frames n° 30 et 31
 Frame n° 32	 Frame n° 32 reconstruite à partir de la frame n° 31 reconstruite	 Entre les frames n° 31 et 32
 Frame n° 33	 Frame n° 33 reconstruite à partir de la frame n° 32 reconstruite	 Entre les frames n° 32 et 33
 Frame n° 34	 Frame n° 34 reconstruite à partir de la frame n° 33 reconstruite	 Entre les frames n° 33 et 34
 Frame n° 35	 Frame n° 35 reconstruite à partir de la frame n° 34 reconstruite	 Entre les frames n° 34 et 35

Tableau 14: résultats de l'algorithme FSS sur la séquence 'Foreman' (frames 30-35)

IV.5. Résultats de l'algorithme DS :

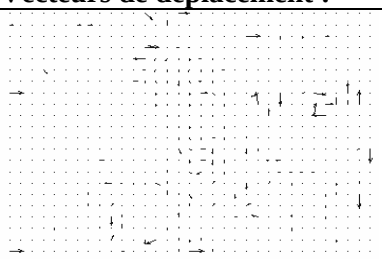
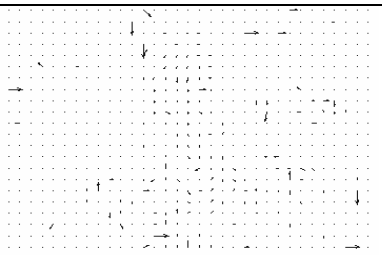
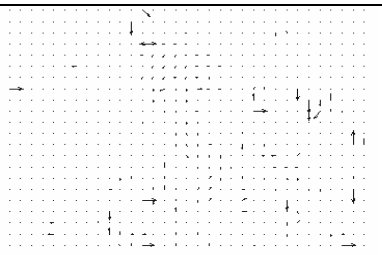
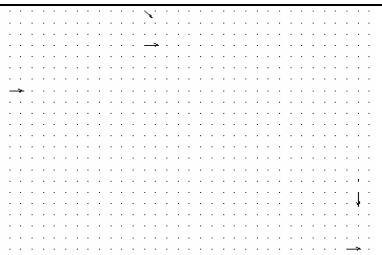
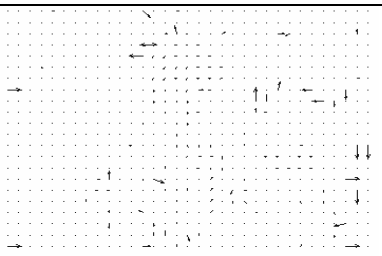
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 21	 Frame n° 21 reconstruite à partir de la frame n° 20 originale	 Entre les frames n° 20 et 21
 Frame n° 22	 Frame n° 22 reconstruite à partir de la frame n° 21 reconstruite	 Entre les frames n° 21 et 22
 Frame n° 23	 Frame n° 23 reconstruite à partir de la frame n° 22 reconstruite	 Entre les frames n° 22 et 23
 Frame n° 24	 Frame n° 24 reconstruite à partir de la frame n° 23 reconstruite	 Entre les frames n° 23 et 24
 Frame n° 25	 Frame n° 25 reconstruite à partir de la frame n° 24 reconstruite	 Entre les frames n° 24 et 25

Tableau 15: résultats de l'algorithme DS sur la séquence 'Akiyo' (frames 20-25)

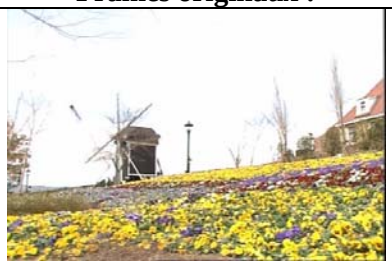
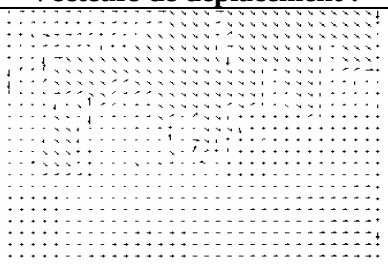
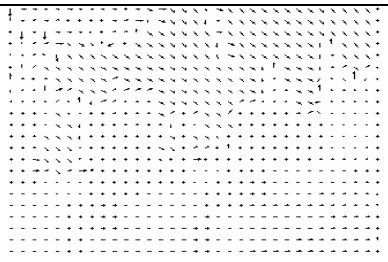
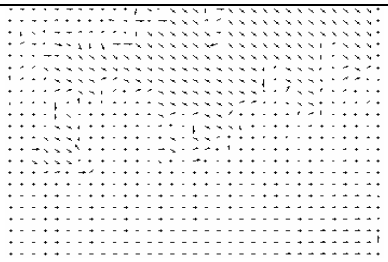
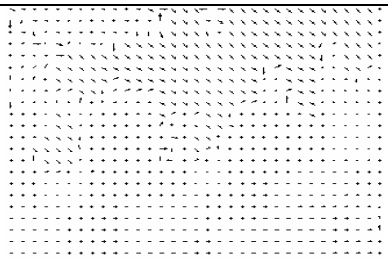
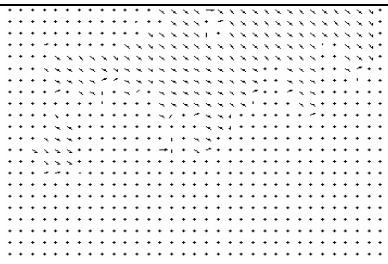
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 2	 Frame n° 2 reconstruite à partir de la frame n°1 originale	 Entre les frames n° 1 et n° 2
 Frame n° 3	 Frame n° 3 reconstruite à partir de la frame n° 2 reconstruite	 Entre les frames n° 2 et n° 3
 Frame n° 4	 Frame n° 4 reconstruite à partir de la frame n° 3 reconstruite	 Entre les frames n° 3 et n° 4
 Frame n° 5	 Frame n° 5 reconstruite à partir de la frame n° 4 reconstruite	 Entre les frames n° 4 et n° 5
 Frame n° 6	 Frame n° 6 reconstruite à partir de la frame n° 5 reconstruite	 Entre les frames n° 5 et n° 6

Tableau 16: résultats de l'algorithme DS sur la séquence 'Flower Garden' (frames 1-6)

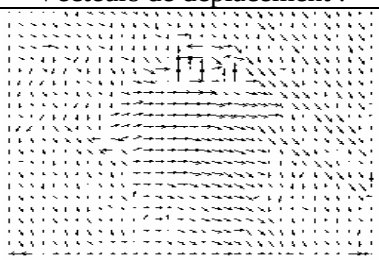
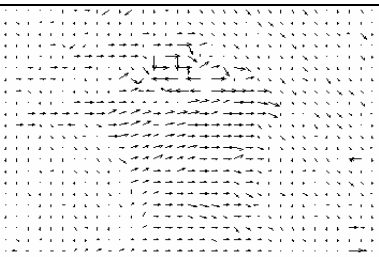
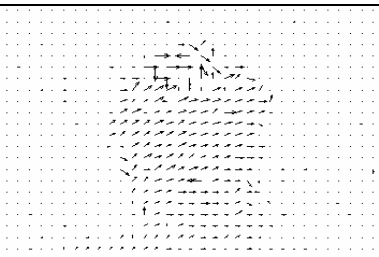
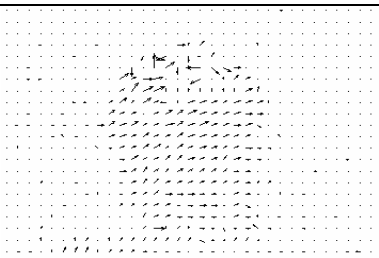
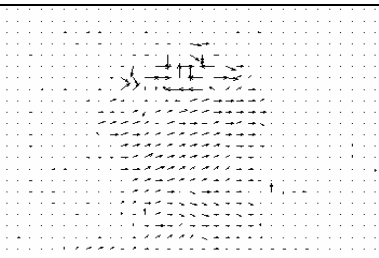
Frames originaux :	Frames reconstruites :	Vecteurs de déplacement :
 Frame n° 31	 Frame n° 31 reconstruite à partir de la frame n° 30 originale	 Entre les frames n° 30 et 31
 Frame n° 32	 Frame n° 32 reconstruite à partir de la frame n° 31 reconstruite	 Entre les frames n° 31 et 32
 Frame n° 33	 Frame n° 33 reconstruite à partir de la frame n° 32 reconstruite	 Entre les frames n° 32 et 33
 Frame n° 34	 Frame n° 34 reconstruite à partir de la frame n° 33 reconstruite	 Entre les frames n° 33 et 34
 Frame n° 35	 Frame n° 35 reconstruite à partir de la frame n° 34 reconstruite	 Entre les frames n° 34 et 35

Tableau 17: résultats de l'algorithme DS sur la séquence 'Foreman' (frames 30-35)

IV.6. Résultats de l'algorithme MIN :

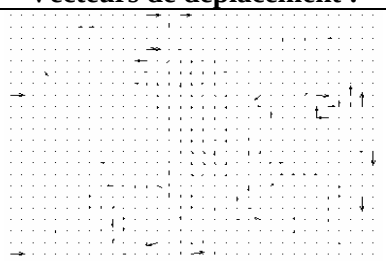
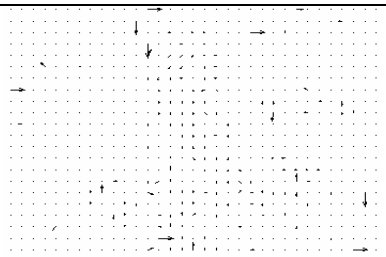
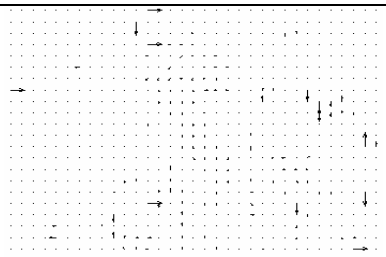
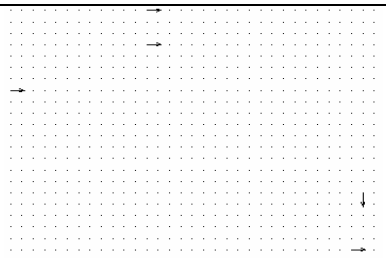
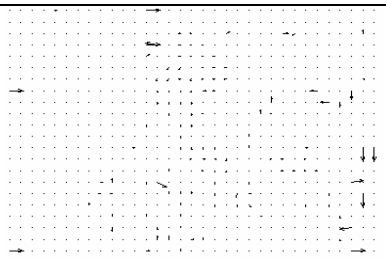
Frames originaux :	Frames reconstruites :	Vecteurs de déplacement :
 Frame n° 21	 Frame n° 21 reconstruite à partir de la frame n° 20 originale	 Entre les frames n° 20 et 21
 Frame n° 22	 Frame n° 22 reconstruite à partir de la frame n° 21 reconstruite	 Entre les frames n° 21 et 22
 Frame n° 23	 Frame n° 23 reconstruite à partir de la frame n° 22 reconstruite	 Entre les frames n° 22 et 23
 Frame n° 24	 Frame n° 24 reconstruite à partir de la frame n° 23 reconstruite	 Entre les frames n° 23 et 24
 Frame n° 25	 Frame n° 25 reconstruite à partir de la frame n° 24 reconstruite	 Entre les frames n° 24 et 25

Tableau 18: résultats de l'algorithme MIN sur la séquence 'Akiyo' (frames 20-25)

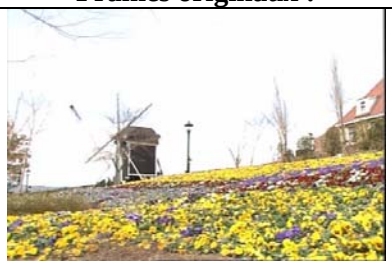
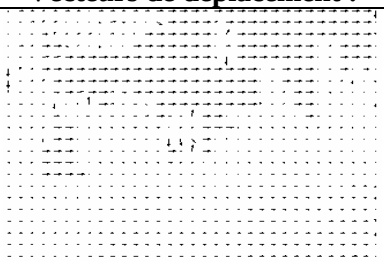
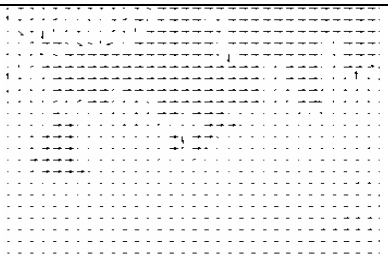
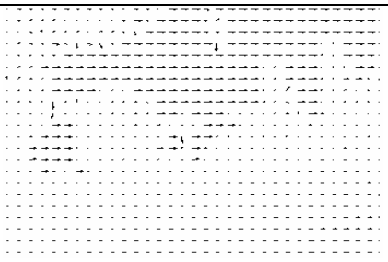
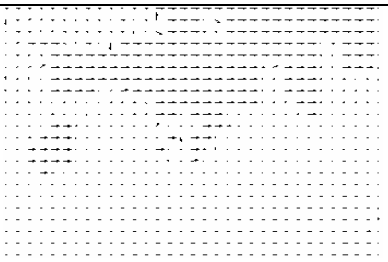
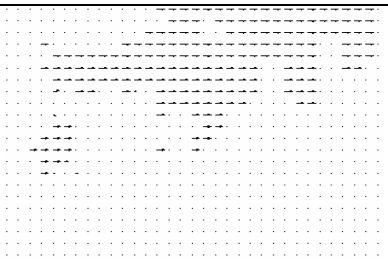
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 2	 Frame n° 2 reconstruite à partir de la frame n° 1 originale	 Entre les frames n° 1 et n° 2
 Frame n° 3	 Frame n° 3 reconstruite à partir de la frame n° 2 reconstruite	 Entre les frames n° 2 et n° 3
 Frame n° 4	 Frame n° 4 reconstruite à partir de la frame n° 3 reconstruite	 Entre les frames n°3 et n° 4
 Frame n° 5	 Frame n° 5 reconstruite à partir de la frame n° 4 reconstruite	 Entre les frames n° 4 et n° 5
 Frame n° 6	 Frame n° 6 reconstruite à partir de la frame n° 5 reconstruite	 Entre les frames n° 5 et n° 6

Tableau 19: résultats de l'algorithme MIN sur la séquence 'Flower Garden' (frames 1-6)

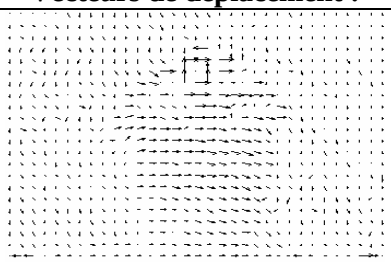
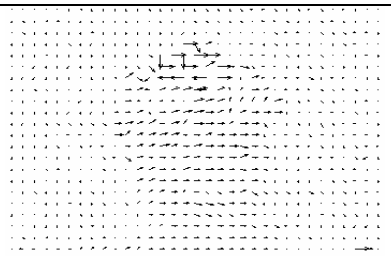
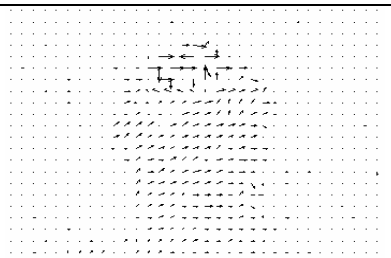
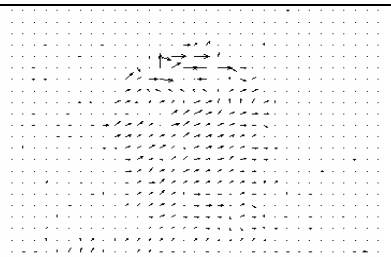
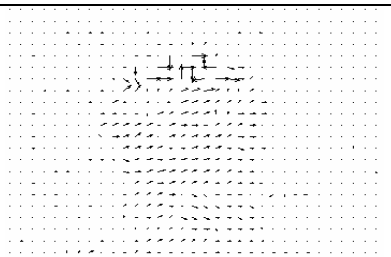
Frames originaux :	**Frames reconstruites :**	**Vecteurs de déplacement :**
 Frame n° 31	 Frame n° 31 reconstruite à partir de la frame n° 30 originale	 Entre les frames n° 30 et 31
 Frame n° 32	 Frame n° 32 reconstruite à partir de la frame n° 31 reconstruite	 Entre les frames n° 31 et 32
 Frame n° 33	 Frame n° 33 reconstruite à partir de la frame n° 32 reconstruite	 Entre les frames n° 32 et 33
 Frame n° 34	 Frame n° 34 reconstruite à partir de la frame n° 33 reconstruite	 Entre les frames n° 33 et 34
 Frame n° 35	 Frame n° 35 reconstruite à partir de la frame n° 34 reconstruite	 Entre les frames n° 34 et 35

Tableau 20: résultats de l'algorithme MIN sur la séquence 'Foreman' (frames 30-35)

IV.7. Analyse des résultats :

Afin d'améliorer les observations et de clarifier les résultats obtenues; en plus des frames reconstruites on a fait des tableaux qui nous indique le temps de calculs (en secondes) et le rapport signal sur bruit **PSNR** (*Pic Signal Noise Rate*) pour chaque algorithme et pour les trois séquences. Le rapport Signal/Bruit pour chaque image est obtenu par les relations suivantes [14]:

$$PSNR = 20 \log_{10} (P^2 / \sqrt{MSE}).$$
$$MSE = (1 / N) \times \sum_N X_i - X_i^{'})^2 .$$

Où :

MSE : est l'erreur quadratique moyenne.

$\mathbf{X_i}$ **:** les éléments de la frame originale.

$\mathbf{X_i^{'}}$ **:** les éléments de la frame reconstruite.

N : la hauteur de la frame × la largeur de la frame (en pixels),
dans notre cas 352 × 528=185 856.

P : valeur maximale de chaque élément de la frame**,**
dans notre cas 255, car on a des valeurs sur 8 bits.

Algorithmes de recherche		**Frame n° 21**	**Frame n° 22**	**Frame n° 23**	**Frame n° 24**	**Frame n° 25**
Full **S**earch	**Temps de calcul (en secondes)**	487,21	487,14	487,43	487,95	487,03
	PSNR (en dB)	38,95	35,92	33,68	33,67	32,22
Four **S**tep **S**earch	**Temps de calcul (en secondes)**	29,30	29,29	29,25	28,64	29,15
	PSNR (en dB)	39,30	35,96	33,65	33,64	32,20
Diamond **S**earch	**Temps de calcul (en secondes)**	36,34	36,37	36,47	36,37	36,37
	PSNR (en dB)	39,37	36,00	33,69	33,68	32,33
Three **S**tep **S**earch	**Temps de calcul (en secondes)**	45,46	45,61	45,44	45,70	45,45
	PSNR (en dB)	36,91	32,67	30,13	30,13	28,88
New **T**hree **S**tep **S**earch	**Temps de calcul (en secondes)**	32,15	31,86	31,69	28,23	31,43
	PSNR (en dB)	39,45	36,08	33,73	33,72	32,34
Minimum **S**earch	**Temps de calcul (en secondes)**	14,23	12,61	12,40	9,44	12,30
	PSNR (en dB)	39,43	36,07	33,71	33,70	32,30

Tableau 21 : résultats des algorithmes sur la séquence 'Akiyo'.

Algorithmes de recherche		**Frame n° 02**	**Frame n° 03**	**Frame n° 04**	**Frame n° 05**	**Frame n° 06**
Full **S**earch	**Temps de calcul (en secondes)**	500,41	500,82	501,22	500,25	500,47
	PSNR (en dB)	28,72	25,50	23,09	22,20	22,19
Four **S**tep **S**earch	**Temps de calcul (en secondes)**	37,74	37,81	37,68	37,46	33,52
	PSNR (en dB)	28,63	25,42	23,04	22,16	22,15
Diamond **S**earch	**Temps de calcul (en secondes)**	36,61	36,64	36,78	36,62	36,88
	PSNR (en dB)	28,69	25,46	23,02	22,14	22,12
Three **S**tep **S**earch	**Temps de calcul (en secondes)**	46,30	46,25	46,44	46,24	46,46
	PSNR (en dB)	22,58	19,29	18,05	17,43	17,43
New **T**hree **S**tep **S**earch	**Temps de calcul (en secondes)**	46,54	46,48	46,16	45,89	33,23
	PSNR (en dB)	25,41	21,59	19,69	18,86	18,85
Minimum **S**earch	**Temps de calcul (en secondes)**	41,44	41,62	40,67	39,57	27,93
	PSNR (en dB)	28,66	25,45	23,06	22,18	22,17

Tableau 22 : résultats des algorithmes sur la séquence ‘Flower Garden’.

Algorithmes de recherche		**Frame n° 31**	**Frame n° 32**	**Frame n° 33**	**Frame n° 34**	**Frame n° 35**
Full **S**earch	**Temps de calcul (en secondes)**	498,47	498,71	500,24	499,01	499,41
	PSNR (en dB)	33,71	31,93	31,29	30,91	30,34
Four **S**tep **S**earch	**Temps de calcul (en secondes)**	36,58	33,34	31,34	31,03	30,83
	PSNR (en dB)	33,00	31,39	30,78	30,36	29,79
Diamond **S**earch	**Temps de calcul (en secondes)**	36,53	36,42	36,42	36,54	36,38
	PSNR (en dB)	33,05	31,35	30,66	30,26	29,66
Three **S**tep **S**earch	**Temps de calcul (en secondes)**	45,73	45,55	45,55	45,54	45,56
	PSNR (en dB)	28,88	27,07	26,47	25,98	25,27
New **T**hree **S**tep **S**earch	**Temps de calcul (en secondes)**	47,47	45,69	35,88	35,62	35,74
	PSNR (en dB)	31,63	30,00	28,91	28,54	27,70
Minimum **S**earch	**Temps de calcul (en secondes)**	32,10	26,11	18,58	17,72	18,07
	PSNR (en dB)	33,23	31,28	30,52	29,97	29,34

Tableau 23 : résultats des algorithmes sur la séquence ‘Foreman’.

Les algorithmes sont classés dans le tableau suivant selon leurs PSNR, où le meilleur algorithme est celui qui a le PSNR le plus élevé :

Les algorithmes	FS	TSS	NTSS	FSS	DS	MIN
Classement selon PSNR décroissant	n° 01	n° 04	n° 03	n° 03	n° 02	n° 01

Tableau 24 : Classement des algorithmes selon leurs PSNR décroissants.

Dans le tableau suivant les algorithmes sont classés selon leurs temps de calcul, donc le meilleur algorithme est celui qui à le temps de calcul le plus court :

Les algorithmes	FS	TSS	NTSS	FSS	DS	MIN
Classement selon le temps de calcul croissant	n° 04	n° 03	n° 03	n° 02	n° 02	n° 01

Tableau 25 : Classement des algorithmes selon leurs temps de calcul croissants.

En plus de ces deux critères (PSNR et temps de calcul) ; on a préférer d'améliorer la comparaison par un troisième critère qui est l'existence des déformation au niveau des frames reconstruites (images) des trois séquence, qui est le contenu du tableau suivant :

Les algorithmes	FS	TSS	NTSS	FSS	DS	MIN
Existence des déformations sur la séquence 'Akiyo'	non	trop	peu	peu	non	peu
Existence des déformations sur la séquence 'Flower Garden'	peu	trop	peu	peu	non	peu
Existence des déformations sur la séquence 'Foreman'	non	trop	trop	non	non	peu
La performance	n° 02	n° 06	n° 05	n° 03	n° 01	n° 04

Tableau 26 : l'existence des déformations au niveau des frames reconstruites.

Finalement, voila un tableau comparatif final :

Les algorithmes	Temps de calcul	PSNR	Déformations au niveau des frames reconstruites
FS	n° 04	n° 01	n° 02
TSS	n° 03	n° 04	n° 06
NTSS	n° 03	n° 03	n° 05
FSS	n° 02	n° 03	n° 03
DS	n° 02	n° 02	n° 01
MIN	n° 01	n° 01	n° 04

Tableau 27 : performance des algorithmes.

IV.8. Conclusion :

D'après le tableau précédent il est clair et net que l'algorithme DS de recherche en diamant est le plus performant car il présente le meilleur compromis entre le temps de calcul et le PSNR, ainsi que l'absence des déformations au niveau des frames reconstruites.
Donc, on peut classer les algorithmes selon ses performances :

- L'algorithme DS est le plus performant, PSNR élevé et temps de calcul presque le plus court, et pas de déformation sur les trois séquence.
- L'algorithme MIN est meilleur selon ses PSNR élevé et son temps de calcul le plus court, mais son inconvénient est les déformations des frames.
- L'algorithme FS est meilleur selon son PSNR élevé mais son temps de calcul est très élevé.
- L'algorithme FSS a un PSNR élevé et un temps de calcul moyen, et peu de déformations des frames.
- L'algorithme NTSS a un PSNR moyen et un temps de calcul moyen aussi, et des déformations aux niveaux des frames.
- L'algorithme TSS a un PSNR le moins important et un temps de calcul le plus long, et il y a trop de déformations aux niveaux de ces des frames résultante.

Conclusion Générale:

L'estimation de mouvement est une étape très nécessaire pour la compression des images vidéos, elle permet de réduire énormément l'espace mémoire nécessaire au stockage d'une image. L'estimation de mouvement par la méthode de mise en correspondance est l'une des méthode les plus utilisées dans ce domaine vue sa simplicité et son efficacité. Chaque bloc dans la frame courante est le résultat d'un bloc de la frame précédente et un vecteur de déplacement, donc c'est une estimation locale de mouvement. La matrice des vecteurs de déplacement s'appelle ' flot optique'.

Donc au lieu de transmettre 6 frames successives dans une séquence, il faut transmettre une seule frame et cinq matrices de (33 × 22) éléments qui représentent les vecteurs de déplacements, c'est-à-dire 726 × 5 = 3630 éléments, sachant qu'une frame contient 528 × 352 × 3 = 557 568 éléments on a un taux de compression de

(557 568 + 3630) / (557 568 × 6) = 16,77 %.

Dans ce cadre plusieurs algorithmes de recherche pour la mise en correspondance ont été appliqués à des séquences vidéo qui sont largement utilisés pour le test (Foreman, Akiyo, Flower Garden). Les résultats obtenus montrent que la performance d'un algorithme est un compromis entre le rapport du signal sur bruit PSNR et le temps de calcul, prennent en compte les déformations des frames résultantes. De point de vue temps, on remarque que l'algorithme FS nécessite un temps de calcul énorme, c'est logique car la recherche est exhaustive, mais il donne de bons résultats en rapport sur bruit. L'algorithme TSS et NTSS nécessitent moins de temps de calcul par contre donnent un rapport signal sur bruit faible par rapport au FS. Pour l'algorithme MIN le temps de calcul est vraiment aléatoire et faible par rapport à celui du FS avec un rapport signal sur bruit faible par rapport au FS. L'algorithme DS peut être jugé comme le meilleur algorithme car il donne un rapport signal sur bruit équivalent à celui du FS et nécessite moins de temps de calcul ainsi que l'absence des déformations aux niveaux des frames reconstruites.

Ces résultats restent une application directe des algorithmes qui peuvent être améliorés par l'association avec d'autres techniques d'optimisation et/ou une implémentation hardware pour une exécution en temps réelle.

BIBLLIOGRAPHIE :

[01] Agate, C.S.; Iltis, R.A.; "An image tracking algorithm using reduced sufficient statistics". IEEE signals, systems and computers. Volume 1, Page(s):444 - 448 vol.1. Nov. 1995.

[02] Baumela, L.; Agapito, L.; Bustos, P.; Reid, I.; "Motion estimation using the differential epipolar equation". 15 th international conference on pattern recognition. Volume 3, Page(s):840 – 843. Sept. 2000.

[03] Bellaiche Philipe, "les secrets de l'image video: colorimétrie, éclairage optique, signal vidéo, compression numérique, et formats d'enregistrement", 5éme édition, Paris, Eyrolles, 2004.

[04] Beric, A.; de Haan, G.; Sethuraman, R.; van Meerbergen, J.; "A technique for reducing complexity of recursive motion estimation algorithms". IEEE workshop on signal processing systems. Page(s):195 - 200 Aug. 2003.

[05] B. K. P. Horn and B. G. Schunck, "Determining optical flow," Artificial Intelligence, vol. 17, pp. 185-203, 1981.

[06] Chich-Ling Huang; Yuh-Ren Choo; Pau-Choo Chung; "Combining region-based differential and matching algorithms to obtain accurate motion vectors for moving object in a video sequence". Proceedings. 22 nd international conference on distributed computing systems workshops. Page(s):202 – 207. July 2002

[07] Christoph Stiller and Janusz Konrad ,"Estimating Motion in Image Sequences: A tutorial on modelling and computation of 2D motion", IEEE Signal Processing Magazine , July 1999 .

[08] Chung-Neng Wang, Shin-Wei Yang, Chi-Min Liu and Tihao Chiang, "A hierarchical decimation lattice based on N-queen with an application for motion estimation" , IEEE signal processing letters, Vol. 10, N°. 8, Aout 2003.

[09] Coban, M.Z.; Mersereau, R.M.; "Fast rate-constrained N-step search algorithm for motion estimation". Proceedings of the acoustics, speech, and signal processing, Volume 5, Page(s):2613 – 2616. May 1998

[10] Frédéric Dufaux and Fabrice Moscheni, "Motion Estimation Tehcniques for Digital TV: A Review and a New Contribution", Proceedings of the IEEE, Vol 83, N° 6, Juin 1995.

[11] Fulvio Moschetti Murat Kunt and Eric Debes, "A statical adaptive block matching motion estimation", IEEE transactions on circuits and systems for video technology, Vol 13 N° 4 Avril 2003 .

[12] Gharavi, H.; Reza-Alikhani, H.; "Pel-recursive motion estimation algorithm" IEEE Electronics letters. Volume 37, Issue 21, Page(s):1285 – 1286. Oct 2001.

[13] Hui Gu and Yan-Shan Li, "The research of block motion estimation algorithm in video compression", IEEE transactions on circuits and systems for video technology, 2004.

[14] Atsuro Ichigaya, Masaai Kurozumi, Naohiro Hara, Yukihiro Nishida and Eisuke Nakasu, ”A Method of Estimaing Coding PSNR Using Quantized DCT Coefficients”, IEEE transactions on circuits and systems for video technology, Vol. 16, N°. 2, February 2006.

[15] Jau-Ling Chen and Pei-Yin Chen, “A new search algorithm for block motion estimation”, IEEE transactions on circuits and systems for video technology.pp 979-982, 2000

[16] Jie-Bin Xu, Lai-Man Po, and Chok-Kwan Cheung, “Adaptive Motion Tracking Block Matching Algorithms for Video Coding”, IEEE transactions on circuits and systems for video technology, Vol.9, N°. 7, October 1999.

[17] Jinwen Zan, M. Omair Ahmed, M. N. S. Swamy, “A Multiresolution motion estimation technique with indexing”, IEEE transactions on circuits and systems for video technology, Vol 16, N°. 2, February 2006.

[18] Jong-Nam Kim and Tae-Sun Choi, “A Fast Three-Step Search Algorithm with Minimum Checking Points Using Unimodal Error Surface Assumption”, IEEE transactions on consumer electronics, Vol 44 N° 3 Aout 1998.

[19] Jodoin, P.-M.; Mignotte, M.; “Unsupervised motion detection using a Markovian temporal model with global spatial constraints”. IEEE international conference on image processing. Volume 4, Page(s):2591 – 2594. Oct. 2004.

[20] Kuan-Tsang Wang , Oscal T.-C. Chen, “Motion estimation using an efficient four-step search method”, IEEE transaction on circuits and systems for video technology, vol. IV page 217-220

[21] Kuo-Liang Chung and Lung-Chun Chang, “A new predictive search area approach for fast block motion estimation”, IEEE transactions on image processing, Vol 12. N° 6, Juin 2003.

[22] Ko, S.-J.; Forest, T.M.; “Image sequence enhancement based on adaptive symmetric order statistics”. IEEE transactions on circuits and systems II: analog and digital signal processing. Volume 40, Issue 8, Page(s):504 – 509. Aug. 1993.

[23] Lai-Man Po , Wing-Chung Ma, “A novel four- step search Algorithm for fast Block Motion Estimation” , IEEE transactions on circuits and systems for video technology 1996 , vol 6 n° 3 page 313-317.

[24] Lai-Man Po; Chok-Kwan Cheung; “A new center-biased orthogonal search algorithm for fast block motion estimation”. Proceedings IEEE TENCON, digital signal processing applications. Volume 2, Page(s):874 – 877. Nov. 1996.

[25] Lijun Luo, Cairong Zou, Xiqi Gao, Zhenya He,“A new prediction search algorithm for block estimation in video coding”, IEEE transactions on consumer electronics, Vol. 43, N°. 1, Février 1997.

[26] Lopes, F.; Ghanbari, M.; “Hierarchical motion estimation with spatial transforms”. Proceedings. International conference on image processing. Volume 2, Page(s):558 – 561. Sept. 2000.

[27] Padmanabhan, A.; ShaoHua Tan; Kwong Huang Goh; "A study of fast block matching algorithms for H.263". Proceeding of information, communications and signal processing. Volume 1, Page(s):301 – 305. Sept. 1997.

[28] Mei-Juan Chen, Liang-Gee Chen , Tzi-Dar Chiueh, "One dimentional Full Search motion estimation algorithm for video coding", IEEE transactions on circuits and systems for video technology. Vol 4, N° 5 , Octobre 1994.

[29] Renxiang Li, Bing Zeng, and Ming L. Liou, "A New Three-Step Search Algorithm For Block Motion Estimation", IEEE transactions on circuits and systems for video technology. Vol 4 N° 4 Aout 1994.

[30] Schultz, R.R.; Stevenson, R.L.; "Bayesian motion estimation without spatial and temporal gradients". IEEE 39 th Midwest symposium on Circuits and systems.Volume 3, Page(s):1385 – 1388. Aug. 1996.

[31] Schultz, R.R.; Stevenson, R.L.; "Bayesian estimation of subpixel-resolution motion fields and high-resolution video stills". Proceedings, International conference on image processing. Volume 3, Page(s):62 – 65. Oct. 1997.

[32] Shan Zhu , Kai-Kuang Ma, "A new diamond search algorithm for fast block matching motion estimation", IEEE transactions on image processing, Vol 9, N° 2, fevrier 2000, pp 287-290.

[33] Shijun Sun; Haynor, D.; Yongmin Kim; "Motion estimation based on optical flow with adaptive gradients". International conference on image processing, proceedings.Volume 1, Page(s):852 – 855. Sept. 2000.

[34] Sohail Zafar, Ya-Qin Zhang, and John S. Baras, "predictive block matching motion estimation for TV coding ,Part I: InterBlock prediction", IEEE transactions on broadcasting, Vol 37, N°. 3, pp 97-101, Sebtembre 1991.

[35] Sohail Zafar, Ya-Qin Zhang, and John S. Baras, "Predictive block matching motion estimation for TV coding ,Part II: InterFrame prediction", IEEE transactions on broadcasting, Vol 37, N°. 3, pp 97-101, Sebtembre 1991.

[36] Viet-Anh Nguyen and Yap-peng Tan, "Efficient Block Matching Motion Estimation Based on Integral Frame Attributes", IEEE on circuits and systems for video technology, Vol. 16, N°. 3, March 2006.

[37] VSK Reddy and Somnath Sengupta, "A new predictive Full-Search Block Motion Estimation", Proceeding of the 17th international Conference on pattern Recognition 2004.

[38] Watkinson John, " la réduction de débit en audio et vidéo : JPEG, MPAG-1, MPEG-2, redondance ", Paris , Eyrolles, 1998.

[39] Xuan Jing and Lap-Pui Chau, "An Efficient Three-Step Search Algorithm For Block Motion Estimation", IEEE transactions on multimedia, Vol 6, No 3, Juin 2004.

[40] Yasuyuki Nakajima, Akiyo Yoneyama, Masaru Sugano, and Hiromasa Yanagihara, "A fast motion estimation algorithm for MPEG2 video using ripple-shaped search", IEEE transactions on circuits and systems for video technology,pp 207-210, 1999.

[41] Yih-Chuan and Shen-Chuan Tai, “Fast Full Search Block Matching Algorithm for Motion Compensated Video Compression”, IEEE transactions on communications, Vol 45, N°5, Mai 1997.

[42] Yilong Liu and Soontorn Oraintara, “Complexity comparison of block matching motion estimation algorithms”, IEEE transactions on circuits and systems for video technology, pp 341-344, 2004.

[43] Yu Yue, Zhou Jion Wang Yiliang, Li Fengting and Ge Ghenghui, “A fast effective block motion estimation algorithm”, IEEE transactions on circuits and systems for video technology,1998.

[44] Zhongli He and Ming L. Liou, “A high performance fast search algorithm for block matching motion estimation”, IEEE transactions on circuits and systems for video technology, Vol. 7 N° 5,pp 826-828, Octobre 1997.

Liste des figures :

Liste des tableaux :

E-mail : bailiche_amine@yahoo.fr