REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

UNIVERSITE MOHAMED KHIDER
BISKRA
Faculté des Sciences et des Sciences de l'ingénieur
Département d'Informatique

Mémoire de Magister

Option: Intelligence Artificielle et Images

Thème:

Compression des images animées par estimation de mouvement

Présenté par :
Mr: DJEFFAL Abdelhamid

Composition du jury:

Président	-Pr. M.C. Baatouche
Examinateurs	-Dr. M.K. Kholladi
	-Dr. A. Khelifa
	-Dr. A. Boukerram
Rapporteur	-Pr N.Djedi

2003-2004

A ma mère,

à mon père,

à la mémoire de ma grand-mère,

à la mémoire de mon grand-père,

à mes frères et sœurs,

et spécialement à celui qui a refusé que j'écrive son nom.

Je dédie ce mémoire.

Remerciements

Je remercie infiniment Dr. N.Djeddi pour le suivi de mon travail, pour ses lectures, pour sa grande patience et pour l'aide administratif qu'il nous a apporté durant la préparation de ce mémoire.

Je voudrais remercier Dr. Z.E.Baarir pour les lectures enrichissantes de mon mémoire. Merci encore de m'avoir accueilli au sein de son équipe de recherche "Traitement d'images" du laboratoire de recherche LESIA.

Mes remerciements vont également au membres de jury pour m'avoir honorer par leur participation à l'évaluation de mon travail.

Je doit remercier aussi Mr M.C.Babahenini le chef du département de l'informatique pour son soutient sans limites aux post-graduant, et pour son administration de qualité.

Je remercie également les membres de l'équipe d'image: Dr Khelifa, Mr Ouafi, Mr Zitouni, M^{elle} Terki pour leur aide.

Je tiens à remercier spécialement Mr A.Kheirallah pour son aide que je n'oublierai jamais, ainsi que l'association ElBadr pour l'aide que ses membres m'ont apportée.

Merci à tous mes collègues étudiants en post-graduation: Mr Kahloul, Mr Khelfali, M^{elle} Hattab, M^{me} Touil, Mr Achouri, Mr Abed...

Merci à tout le personnel de la direction de l'Education de Biskra, spécialement Mr. A.Djoudi, M^{me} F.Becha, Lyes, Faycel,...

Merci à tous mes amis Sif Eddine, Lakhdar, Brahim, Anouar, Messaoud, Kamel,....

James B. Conan,
Science and common sense.

"La route chaotique sur laquelle même le plus capable des chercheurs a dû, à chaque génération, se frayer un chemin entre les bosquets d'observations erronées, des généralisations trompeuses, de formulations inadéquates et de préjudices inconscients est rarement appréciée de ceux qui tirent leurs savoir des livres."

ملـخص

لا تتوقف تكنولوجيا الفيديو الرقمي عن التطور . فإضافة إلى التطبيقات المعروفة والتي تحتـــاج إلى مزيد من التحسينات، هناك تطبيقات جديدة كالبريد الإلكتروني، الهواتف المحمولـــة مـــن الجيـــل الثالث، الملتقيات عن بعد الخ، والتي تدفع البحث في مجال تشفير مقطوعات ا لفيديو أبعد فأبعد. كـــل المعلومات من نوع الفيديو تحتاج إلى الضغط أو الدمج للتخفيف من حجمها.

تعتبر الصور المكون الأساسي في مقطوعات الفيديو ويسهِّ ل ضغطها ويسمح بتطبيقـــات عديـــدة ومهمة، إلا أن ضغط كل صورة على حدة لا يعتبر حلا امثلا ولا يأخذ التشابه الموجود بين الـــصـ ور المتتالية بعين الاعتبار . تسمح تقنية توقع واسترجاع الحركة بين صور المقطوعـــة بتـــشفير الـــصور اعتمادا على صور أخرى وهي التقنية التي تستعملها المقاييس المعروفة MPEG و H26x .

تعتمد تقنية توقع واسترجاع الصور على البحث عن الكائنات الموجودة في الصورة الحالية فـــي الصورة المرجع (سابقة أو لاحقة) وتستعمل في معظم التطبيقات تقنية أخرى تسمى "مقارنـــة الكتـــل " نظرا لنجاعتها وسهولة تطبيقها في الدارات . تعتمد هذه التقنية على تقسيم الصورة إلى كتل مربعة من النقاط الغير متداخلة والبحث عن هذه الكتل في المراجع.

في هذا العمل، صُمم و أنجز مشفر فيديو اعتمادا على تقنية توقع واسترجاع الحركة مـــن اجـــل تطبيق ودراسة مختلف خوارزميات هذه التقنية . تُقترح أيضا تقنية مهجنة تعتمد على التعرف على نوع حركة الكتل (ثابتة، بطيئة أو سريعة) من خلال المعلومات الملتقطة من الصور الـــسابقة واســـتعمالها لاختيار الخوارزم المناسب. تعتمد التقنية المقترحة أيضا على توقع نقطة بداية البحـــث والبحـــث فـــي مكان محدود حولها عوضا عن تخصيص مجال بحث واسع كما هو معمول به.

كلمات مفاتيح: الفيديو – الضغط – ضغط الفيديو –توقع الحركة - استرجاع الحركة – مطابقـــة الكتل

Abstract

The digital video coding technology is growing more and more. In addition to the known applications needing more refinement, there are new applications (for example: Emails, mobile telephones of third generation, teleconferences…) which push the video coding research to develop more. All video data need to be compressed.

Images are the principal element in video and their compression allows very important applications. However, the compression of each frame (image) separately is not optimal; the redundancy between successive frames must be considered. Motion estimation is a technique based on the compression of frames by referencing others. It is used in the famous standards such as MPEG, H26x.

Motion estimation searches current frame objects in the reference one (past or future). The most used searching methods are the Block Matching algorithms owing to their performance and adaptability for hard implementations. A block matching algorithm subdivides the frame into non-overlapped rectangular blocks and searches these blocks in the reference frame.

In this work, a motion estimation based video codec is designed and implemented for the application, study and comparison of motion estimation algorithms.

A hybrid search technique is proposed too, based on block motion type detection from the information gathered from the precedent estimations. The block motion type (stationary, slow or large) is used to choose the appropriate algorithm to apply in search.

The hybrid technique is based also on search starting point estimation instead of considering a large searching zone.

Key words: video - compression – coding – motion estimation – motion compensation – block matching

Table des matières

TABLE DES MATIERES IV

LISTE DES FIGURES VIII

LISTE DES TABLES X

LEXIQUE DES ABREVIATIONS XI

INTRODUCTION GENERALE 1

I. COMPRESSION VIDÉO 4

I.1. INTRODUCTION 5

I.2. PRINCIPE DE LA VIDÉO 6

I.3. COMPRESSION VIDÉO 7

I.4. HISTORIQUE 8

I.4.1. COMPRESSION PREMIERE GENERATION 8

I.4.2. COMPRESSION DEUXIEME GENERATION 8

I.4.3. COMPRESSION TROISIEME GENERATION 9

I.5. COMPRESSION CONSERVATRICE ET NON CONSERVATRICE 9

I.6. CODAGE INTRAFRAME 11

I.6.1. PASSAGE DU SYSTEME RGB AU SYSTEME YUV 12

I.6.2. SOUS-ECHANTILLONNAGE 13

I.6.3. TRANSFORMATION DCT 13

I.6.4. QUANTIFICATION 15

I.6.5. SATURATION 16

I.6.6. CODAGE RLE (RUN LENGTH ENCODING) 16

I.6.7. CODAGE DE HUFFMAN 16

I.7. CODAGE INTERFRAME 18

I.7.1. CODAGE PREDICTIF 18
I.7.2. CODAGE BIDIRECTIONNEL 19
I.8. SCHÉMA GÉNÉRAL D'UN ENCODEUR VIDÉO. 21
I.9. NORMES ET PRODUITS 22
I.9.1. NORME H261 22
I.9.2. NORME H263 23
I.9.3. NORME H26L 24
I.9.4. NORME MPEG1 24
I.9.5. NORME MPEG-2 25
I.9.6. NORME MPEG-4 25
I.9.7. NORME MPEG-7 26
I.9.8. NORME MPEG-21 26
I.9.9. AUTRES NORMES 26
I.10. CARACTÉRISTIQUES PRINCIPALES D'UN ENCODEUR VIDÉO 26
I.10.1. TAUX DE COMPRESSION 27
I.10.2. DUREE DE COMPRESSION 27
I.10.3. DEBIT 28
I.10.4. QUALITE 29
I.10.5. RESISTANCE AUX ERREURS 29
I.11. CONCLUSION 30

II.ESTIMATION DE MOUVEMENT POUR LA COMPRESSION VIDÉO 31
II.1. INTRODUCTION 32
II.2. CONTRAINTES DE L'ESTIMATION DU MOUVEMENT 34
II.3. ESTIMATION DE MOUVEMENT ET STANDARDS 34
II.4. MÉTHODES D'ESTIMATION DE MOUVEMENT 36
II.4.1. METHODES BASEES SUR L'EQUATION DU FLOT OPTIQUE 36
II.4.2. METHODES DE PIXEL-RECURSIF 36
II.4.3. BLOCK MATCHING 37
II.5. CLASSEMENT DES SÉQUENCES VIDÉO 37
II.5.1. SEQUENCES RAPIDES 38
II.5.2. SEQUENCES LENTES 38
II.5.3. SEQUENCES STATIONNAIRES 38
II.5.4. SEQUENCES SIMPLES 38
II.5.5. SEQUENCES COMPLEXES 38
II.6. CONCLUSION 39

III.ETUDE ET COMPARAISON DES ALGORITHMES DE BLOCK MATCHING 40
III.1. INTRODUCTION 41
III.2. CONCEPTS DE BASE 41

III.2.1. LARGEUR DE LA ZONE DE RECHERCHE 42
III.2.2. NOMBRE DE BLOCS CANDIDATS 43
III.2.3. TAILLE DES BLOCKS 43
III.2.4. PIXELS COMPARES 44
III.2.5. FONCTION DE RESSEMBLANCE 45
III.2.6. SEUILS D'ACCEPTABILITE 46
III.3. COMPLEXITÉ DES ALGORITHMES DE BLOCK MATCHING 46
III.4. ALGORITHMES DE BLOCK MATCHING 47
III.4.1. ALGORITHME DE RECHERCHE COMPLETE (FULL SEARCH :FS) 47
III.4.2. RECHERCHE EN TROIS PAS (THREE STEP SEARCH: 3SS) 48
III.4.3. ALGORITHME DE RECHERCHE 2DLOG 49
III.4.4. ALGORITHME DE RECHERCHE ORTHOGONALE (ORTH. SEARCH: OS) 49
III.4.5. ALGORITHME DE RECHERCHE CROISEE (CROSS SEARCH: CS) 50
III.4.6. RECHERCHE DESCENDANTE DE GRADIENT (GRADIENT SEARCH: GS) 51
III.4.7. NOUVELLE RECHERCHE EN TROIS PAS (NEW 3SS) 51
III.4.8. ALGORITHME DE RECHERCHE EN QUATRE PAS (4SS) 52
III.4.9. BLOCK MATCHING HIÉRARCHIQUE (HIERARCHICAL BM) 53
III.4.10. BLOCK MATCHING A FRACTION DE PIXEL (BMFP) 54
III.5. COMPARAISON DES MÉTHODES DE BLOCK MATCHING 55
III.5.1. SEQUENCES DE TEST 56
III.5.2. STRUCTURE DE L'ENCODEUR 58
III.5.3. FORMAT DES DONNÉES 60
III.5.4. SCHÉMA DU DÉCODEUR 63
III.5.5. RÉSULTATS 63
III.5.6. COMPARAISON 65
III.6. CONCLUSION 70

IV. MÉTHODE HYBRIDE POUR L'ESTIMATION DU MOUVEMENT 71
IV.1. INTRODUCTION 72
IV.2. PRINCIPE DE LA MÉTHODE HYBRIDE 72
IV.2.1. ESTIMATION DES TYPES DES BLOCS 72
IV.2.2. ESTIMATION DE LA METHODE DE LA RECHERCHE 74
IV.2.3. ESTIMATION DE LA ZONE DE RECHERCHE 75
IV.2.4. ESTIMATION DU POINT DE DEPART DE LA RECHERCHE. 75
IV.2.5. CAS DES IMAGES P 76
IV.2.6. CAS DES IMAGES B 77
IV.3. RÉSULTATS ET DISCUSSION 79
IV.4. AUTRE ACCÉLÉRATION: LA RECHERCHE PAR INTERRUPTION 82
IV.5. CONCLUSION 83

CONCLUSION ET PERSPECTIVES 84

1. CONCLUSION 85

2. PERSPECTIVES 86

VI. BIBLIOGRAPHIE 88

Liste des figures

FIGURE 1 : ESTIMATION ET COMPENSATION DU MOUVEMENT 2
FIGURE 2: PRINCIPE DU BALAYAGE UTILISE DANS LA VIDEO PAL/SECAM 6
FIGURE 3: COMPOSANTS DE L'IMAGE COULEUR 7
FIGURE 4: GROUPE D'IMAGES DANS MPEG 9
FIGURE 5: COMPRESSION SPATIALE ET TEMPORELLE 10
FIGURE 6:COMPOSITION D'UNE SEQUENCE VIDEO [VLAN] 11
FIGURE 7:PRINCIPE DU CODAGE JPEG [JAIN 02] 11
FIGURE 8:EFFET DE LA TRANSFORMATION YUV 12
FIGURE 9: SOUS-ECHANTILLONAGE DES MATRICE U ET V 13
FIGURE 10: EXEMPLE DE LA DCT 14
FIGURE 11: DCT HORIZONTALE ET VERTICALE 14
FIGURE 12: EXEMPLE DE LA DCT ET DE LA QUANTIFICATION 15
FIGURE 13: CODAGE EN LONGUEUR DE SEQUENCE 16
FIGURE 14: CODAGE DE HUFFMAN 17
FIGURE 15: CODAGE PAR DIFFERENCE 18
FIGURE 16: COMPRESSION VIDEO PAR ESTIMATION ET COMPENSATION DU MOUVEMENT 19
FIGURE 17: CODAGE BIDIRECTIONNEL 19
FIGURE 18: REFERENCES DU CODAGE BIDIRECTIONNEL 20
FIGURE 19: ORDRE D'EMISSSION DES IMAGES DANS LE CODAGE BIDIRECTIONNEL [CRES 01] 20
FIGURE 20: SCEMA GENERAL D'UN ENCODEUR VIDEO 21
FIGURE 21: FLUX DES DONNEES VIDEO 22
FIGURE 22: COMPRESSION DES IMAGES I DANS H261 23
FIGURE 23: COMPRESSSION DES IMAGES P DANS H261 23
FIGURE 24: ELEMENTS D'UNE SCENE MPEG4 25
FIGURE 25:NOTION DE VECTEUR DE MOUVEMENT 32
FIGURE 26:ESTIMATION DU MOUVEMENT D'UN BLOC 33

FIGURE 27:CODAGE ET DECODAGE DANS L'ESTIMATION DU MOUVEMENT 35
FIGURE 28:MODES D'ESTIMATION DU MOUVEMENT DANS LA NORME MPEG1 35
FIGURE 29:ZONE DE RECHERCHE EN BLOCK MATCHING 42
FIGURE 30:COMPARAISON EN TABLE DE DAMIER 44
FIGURE 31: MODELE DE CORRESPONDANCE 44
FIGURE 32: SEUILS D'ACCEPTABILITE DE RESSEMBLACE DES BLOCS 46
FIGURE 33: RECHERCHE COMPLETE (FULL SEARCH) 47
FIGURE 34: RECHERCHE SPIRALE 48
FIGURE 35:ILLUSTRATION DE LA RECHERCHE EN TROIS PAS 48
FIGURE 36: ILLUSTRATION DE LA RECHERCHE EN 2DLOG 49
FIGURE 37: ILLUSTRATION DE RECHERCHE ORTOGONALE 50
FIGURE 38: ILLUSTRATION DE LA RECHERCHE CROISEE 50
FIGURE 39:ILLUSTARION DE LA RECHERCHE DESCENDATE DE GRADIENT 51
FIGURE 40:ILLUSTRATION DE LA NOUVELLE RECHERCHE EN TROIS PAS 52
FIGURE 41:ILLUSTRATION DE LA RECHERCHE EN QUATRE PAS 53
FIGURE 42:ILLUSTRATION DU BLOCK MATCHING HIERARCHIQUE 53
FIGURE 43:ESTIMATION DE MOUVEMENT A DEMI PIXEL 54
FIGURE 44:PREMIERES IMAGES DES SEQUENCES DE TEST 57
FIGURE 45:SCHEMA DE L'ENCODEUR UTILISE 60
FIGURE 46:SCHEMA DU DECODEUR 63
FIGURE 47:ILLUSTRATION DU PSNR EN FONCTION DU TC 64
FIGURE 48:INFLUENCE DU PARAMETRE MINMDB SUR LE TC/PSNR 65
FIGURE 49:COMPARAISON DES ALGORITHMES DE BM 69
FIGURE 50:SIMILITUDE DES VECTEURS DE MOUVEMENT DES BLOCS VOISINS SPATIALEMENT 73
FIGURE 51:SIMILITUDE DES VECTEURS DE MOUVEMENT DES BLOCS VOISINS TEMPORELLEMENT 73
FIGURE 52:DISTRIBUTION DES VECTEURS DE MOUVEMENT DANS LES SEQUENCES DU MONDE REEL 74
FIGURE 53:ESTIMATION DU POINT DE DEPART 75
FIGURE 54:ESTIMATION DES BLOCS P AVEC LA METHODE HYBRIDE 76
FIGURE 55:ESTIMATION DES BLOCS B PAR LA METHODE HYBRIDE 78
FIGURE 56:AMELIORATION DU TEMPS DE CALCUL EN FONCTION DU NOMBRE D'IMAGES B 81

Liste des tables

TABLE 1: TAUX DE COMPRESSSION PAR TYPE D'IMAGE DANS MPEG1 [CRES 01] 27
TABLE 2: BESOINS EN TEMPS REEL DES DIFFERENTES ETAPES DU CODEUR H261 [CHOK 98] 28
TABLE 3: SEQUENCES UTILISEES POUR LES TESTS 58
TABLE 4: RESULTATS DE L'ENCODEUR COMPARES A MPEG2 ET H263 63
TABLE 5: COMPARAISON DES ALGORITHMES DE BLOCK MATCHING 66
TABLE 6: RESULTATS DU CODAGE PAR LA METHODE HYBRIDE 80
TABLE 7: AMELIORATION AVEC LA RECHERCHE PAR INTERRUPTION 82

Lexique des abréviations

PSNR: Peak Signal to Noise Ratio (Rapport signal sur bruit).

MSE: Mean square Error (Erreur quadratique moyenne).

MAE: Mean Absolute Error.

DFD: Displaced Frame Difference (Différence de mouvement d'une image).

DCT: Discet cosinus Transform.

JPEG: Joint Picture Expert Group.

MPEG: Motion Picture Expert Group.

GIF: Graphic interchange format

CIF: Common intermediate format.

Intra: Image codée sans référence à une autre image.

Inter: Image codée par référence à une autre image.

Block matching : Méthode de correspondance des blocs.

EM: Estimation du mouvement.

BM: Bloc matching.

MB: Macro bloc.

MDB, BDM: Mesure de distorsion des blocs.

3SS: Méthode de recherche en trois pas.

OS: Méthode de recherche orthogonale.

CS: Méthode de recherche croisée.

FS: Méthode recherche complète.

2DLog: Méthode de recherche logarithmique en deux dimensions.

N3SS:Nouvelle 3SS.

4SS: Méthode de recherche en quatre pas.

HBMS: Méthode de recherche hiérarchique en block matching.

GS: Méthode de recherche descendante en gradient.

Introduction générale

Le développement actuel des technologies a donné naissance à des nouvelles techniques de communication dans lesquelles les informations visuelles jouent un rôle très important.

Plusieurs applications connaissent de plus en plus un développement exponentiel et une large utilisation publique, tels que les lecteurs DVD, la télévision numérique, la télévision par satellite, la transmission vidéo par Internet, la vidéo conférence, la sécurité vidéo, les caméras numériques, troisième (et +) génération des téléphones mobiles, …etc. Malheureusement, le développement de ces applications est confronté aux limites imposées par la capacité limitée des mémoires ou des bandes de transmission des réseaux utilisés.

En réalité et contrairement au données textes et audio, les informations vidéo nécessitent et utilisent des quantités très importantes d'informations. Par exemple, pour stocker une minute d'une séquence vidéo de 25 images par seconde, d'une résolution de 720X480 pixels à 24 bits de couleur, il faudrait 60 X 25 X 720 X 480 X 3 octets soit 1.44 GigaOctet d'espace mémoire ou un débit de 248 Mbps pour les transmettre en temps réel.

Ces besoins exponentiels et influant directement sur tout un marché d'imagerie numérique ont poussé à des recherches très intensives, depuis le début des années 90 pour trouver des techniques de compression significatives des données vidéo.

En général, les techniques de compression vidéo exploitent trois types de redondance existant dans une séquence d'images :

1. Une redondance spatiale existant au sein de chaque image matérialisée par la corrélation existant entre les pixels voisins.
2. Une redondance temporelle existant entre les images successives dans la séquence (corrélation entre les images).
3. Une redondance psychovisuelle due aux propriétés de la vue humaine (sensibilité aux basses fréquences).

En réalité, c'est la troisième redondance qui différencie la compression des images de la compression des autres types de données (texte, audio,…) du fait qu'elle présente des avantages

pour modifier les données initiales à compresser sans modifier d'une façon significative le contenu des images.

La compression spatiale appelée aussi Intraframe ou des images fixes a connu un développement très important qui a aboutit à plusieurs normes telles que TIF, GIF, CIF, JPEG,… etc, mais la plus récente et la plus utilisée est la norme JPEG. Dans cette norme l'image subit un ensemble de modification la préparant à une compression classique efficace. Plusieurs principes sont utilisés tel que la DCT, les Ondelettes, les Fractales, … etc.

Pour la compression temporelle (images animées), les efforts investis durant les dix dernières années dans les universités et l'industrie se sont traduits par l'élaboration de plusieurs normes (H261, H263, MPEG1, MPEG2, MPEG4, MPEG7, H26L). Ces Normes viennent répondre aux exigences des différents types d'applications quant au débit binaire, à la qualité des images, à la complexité, à la tolérance aux erreurs, au délai et au taux de compression.

Pour en faire, ces normes utilisent en plus d'une technique de compression fixe (DCT, ondelettes, …), un système de codage vidéo hybride par estimation et compensation de mouvement basé sur des blocs et par transformée [WING 02].

Le principe d'estimation et de compensation de mouvement est le résultat des recherches en codage vidéo, effectuées durant les vingt dernières années, son principe est le suivant :

Le déplacement des objets entres les différentes images est calculé (estimation) pour être utilisé dans un codage efficace inter-images (compensation). L'erreur obtenue est ensuite enregistrée (ou envoyée) au lieu de l'image elle-même (*Figure 1*).

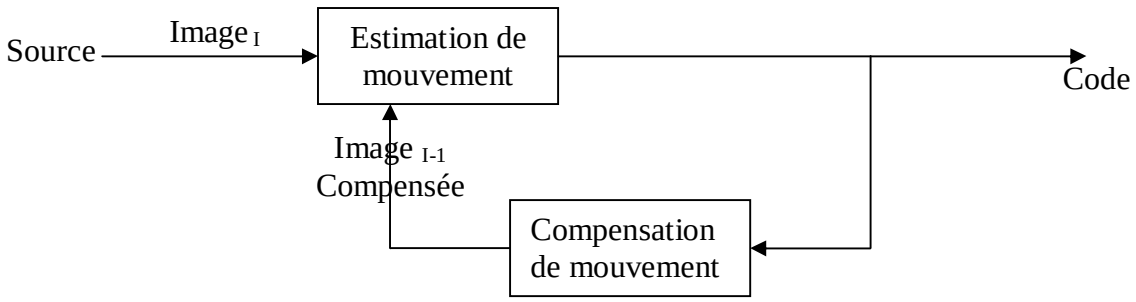


Figure 1 : Estimation et compensation du mouvement

C'est l'objectif du codage (compression, élimination du bruit, vision par ordinateur,…) qui détermine la façon de représentation des objets et leur déplacement. Dans le cas du codage pour compression, les objets sont souvent des blocs rectangulaires de pixels et l'estimation du mouvement consiste en la recherche du bloc de l'image précédente (référence déjà codée) correspondant le plus au bloc courant de l'image courante. Ce principe est appelé correspondance des blocs ou "Block Matching".

Le Block Matching est le moteur fondamental du codage vidéo, c'est sa vitesse et sa précision qui détermine la vitesse et la précision des séquences compressées fournies.

Plusieurs méthodes de Block Matching ont été développées pour les différents types de mouvements (statique (vidéo-conférence) ou rapide (séquences sportives)). Le problème est de trouver un modèle général qui donne le meilleur résultats quelque soit la séquence à compresser [CHOK 98,MOSC 01, KUHN 99, SEOK 01].

Dans ce travail, l'objectif est de réaliser un encodeur vidéo qui permet d'obtenir, à partir d'un ensemble d'images brutes (séquence), un fichier vidéo compressé en implémentant plusieurs modèles de Block Matching pour en tirer les avantages et les inconvénients. On propose en fin une méthode hybride permettant de détecter au cours du codage le type du mouvement et en choisir la méthode du Block Matching qui convient en essayant d'améliorer le temps de codage.

Ce mémoire est organisé comme suit : cette introduction est suivie d'un premier chapitre représentant l'état de l'art de la compression des images animées.

Le deuxième chapitre étudie l'estimation du mouvement et son application dans la compression des images animées. Le troisième chapitre présente les algorithmes de block matching les plus utilisés et leur comparaison. Le quatrième chapitre expose la méthode hybride proposée. Enfin, une conclusion résumant le travail élaboré et des perspectives envisageables sont présentées.

Premier chapitre:

Compression vidéo

I.1. Introduction

Une séquence vidéo est une série d'images accompagnées de son, mais on utilise souvent le terme vidéo pour désigner une simple séquence d'images [TOUR 00]. C'est la fréquence des images et leurs dimensions qui différencient un système vidéo d'un autre. La fréquence des images peut aller jusqu'à 30 images par seconde, ce qui rend inévitable leur compression. Plusieurs normes et produits sont apparus représentant différentes techniques et répondant aux besoins des différentes applications.

Dans ce chapitre, les origines et les principes de la vidéo et de sa compression sont exposés, et les techniques utilisées dans les standards connus sont présentés pour aboutir enfin à un schéma général de compression vidéo qui sera utilisé par la suite.

I.2. Principe de la vidéo

La vidéo est une séquence d'images animées, son principe fondamental est le suivant : l'œil humain peut retenir toute image imprimée sur sa rétine pendant un certain temps de l'ordre du dixième de seconde. En défilant un nombre suffisant d'images par seconde, l'œil ne se rend pas compte qu'il s'agit d'images animées, et il aura l'impression qu'il s'agit d'un mouvement [HERV 95, CRES 01].

Le principe de la vidéo analogique utilisé dans le passé est retrouvé dans tous les systèmes vidéo actuels. Dans l'ancienne télévision en noir et blanc, la caméra balaye l'image avec un faisceau d'électrons se déplaçant rapidement de gauche à droite et plus lentement de haut en bas; elle produit une tension en fonction du temps. Elle enregistre ainsi l'intensité lumineuse. A la fin du balayage, on obtient une trame et le faisceau revient à l'origine pour recommencer l'opération (*Figure 2*).

Le récepteur reçoit cette intensité en fonction du temps, et pour reconstruire l'image, il répète le processus du balayage sur un écran phosphorique.

C'est la différence des paramètres de ce balayage qui a donné naissance aux systèmes vidéo connus tels que le système PAL/SECAM (*Phase ALternating Line/ SEquenciel Couleur Avec Mémoire*) utilisant 625 lignes et 25 trames par seconde, et le système NTSC (*National Television Standards Committee*) utilisant 525 lignes et 30 trames par seconde.

Avec l'affichage de 25 images par seconde, on trouve des personnes qui peuvent voir un papillotement sur l'image [CRES 01]. Pour régler ce problème et au lieu d'augmenter le débit des trames, on a préféré afficher d'abord les lignes impaires puis les lignes paires; ce principe est appelé l'Entrelacement.

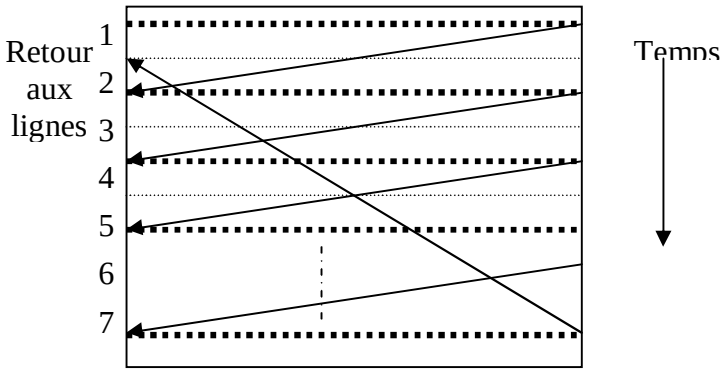


Figure 2: Principe du balayage utilisé dans la vidéo PAL/SECAM

La télévision et la vidéo couleur utilisent le même principe de balayage, mais au lieu d'utiliser un seul faisceau, on en utilise trois, un pour chaque couleur de base (RVB : Rouge –

Vert – Bleu ou RGB : Red – Green – Blue); ces trois signaux sont combinés ensuite en un signal de luminance Y et deux signaux de chrominance U et V.

La vidéo numérique est simplement une suite de trames composées de matrices rectangulaires de pixels. Pour avoir des images en couleur, il faut utiliser au moins un octet par pixel c–à–d 256 couleurs, cela permet aussi d'avoir une vidéo noir et blanc de haute qualité.

Pour la vidéo numérique couleur, on utilise un octet pour chaque couleur RVB (Figure 3) soit 24 bits par pixel, ce qui permet d'avoir environs 16.8 millions de couleurs.

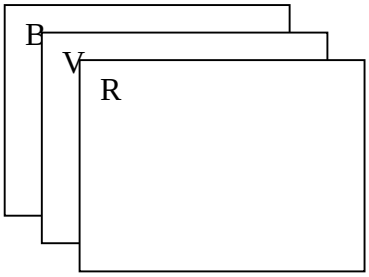


Figure 3: Composants de l'image couleur

Pour afficher une vidéo à 25 trames par seconde sur un écran XVGA (1024X768) en 16.8 millions de couleurs (24 bits), on aura besoin d'un débit minimum de 60 MO/s. D'où la nécessité de la compression.

I.3. Compression vidéo

La compression ou aussi le codage vidéo est utilisée pour faciliter le stockage ou la transmission des données vidéo, son principe est de réduire au maximum possible la redondance dans les données des trames sans que cela ne soit visible d'une façon remarquable pour l'œil humain. Tout le compromis est là : plus le taux de compression s'améliore plus la qualité de l'image devient médiocre.

En général, la compression vidéo se fait en quatre étapes [GALP 02]:

1. Présentation : Les données vidéo sont présentées sous une forme adaptée à la compression.
2. Décorrélation : Les données de départ sont représentées dans un espace qui permet d'éliminer le maximum de redondance.
3. Quantification : Les données sont approximées de manière à éliminer les informations superflues en prenant en compte les critères psychovisuels de la la vision humaine. C'est cette étape qui introduit la dégradation des données de départ et permet de varier le taux de compression.
4. Codage Entropique : permet de réduire la taille des symboles à transmettre en exploitant leur répartition statistique.

I.4. Historique

La compression vidéo s'est développée en commençant par de simples techniques s'appliquant au niveau des pixels en arrivant aux techniques complexes s'appliquant sur des séquences toutes entières. Cette évolution a connu trois générations [GALP 02]:

I.4.1. Compression première génération

Au début de la compression des images vidéo, on ne voyait de l'image qu'un ensemble d'octets et leur compression n'était vue que du coté symbolique.

Par exemple, le codage de Huffman, utilisé dans la norme MPEG et qui sera étudié ultérieurement, était utilisé pour coder les données vidéo brutes.

La DPCM (Differential Pulse Code Modulation), était ensuite utilisée, elle consiste à ne coder que la différence entre un pixel et son voisin.

I.4.2. Compression deuxième génération

Dans cette génération, la compression a commencé à se baser sur une vue globale du signal image, c-à-d prendre en considération les critères psycho visuels de l'œil humain. On essaye de dégrader les images dans les zones de faible signifiance pour l'œil. La mesure de « Qualité » d'image a été introduite et elle est rapprochée du rapport signal sur bruit :

$$PSNR = 10Log10\left(\frac{255^2}{\frac{1}{N}\sum(p-p')^2}\right)$$

Où p et p' sont les pixels des images originale et reconstruite (décompressée) respectivement et N est le nombre de pixels de l'image.

Afin d'exploiter les critères psycho visuels, plusieurs transformations ont été introduites sur l'image avant de procéder à son codage pour la préparer à une compression efficace. La transformation dans un domaine fréquentiel est la transformation qui convient. La transformation DCT (*Discrete Cosine Transform*) est largement utilisée dans les normes JPEG, MPEG, H26x.

La transformation DWT (Discrete Wavelet Transform) (transformation en ondelettes) est utilisée notamment dans la norme JPEG2000.

Dans cette génération, on peut voir aussi les caractéristiques des séquences. La norme MPEG utilise un principe consistant à structurer le flux d'images en GOP (*Group Of Pictures*) contenant une première image codée en mode *Intra* indépendamment des autres; et les autres sont

en mode *Prédictif* (***Figure 4***) c'est-à-dire basées sur la différence avec l'image Intra ou Prédictive précédente.

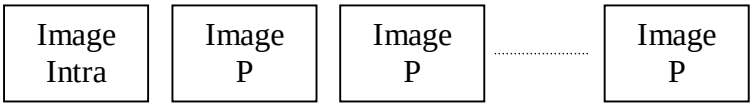


Figure 4: Groupe d'images dans MPEG

I.4.3. Compression troisième génération

Dans cette génération, le signal vidéo n'est pas traité du coté pixel (première génération) ni du coté image (deuxième génération) mais du coté de la scène filmée toute entière. On passe par une étape de segmentation et reconnaissance de formes pour extraire les objets de la scène et ne les coder qu'une seule fois en terme de texture tel que le codage par mosaïque ou par maillage [CHAN 95]. Ceci a permis d'ajouter des connaissances préalables au codage tels que les mouvements du visage humain dans les applications de la vidéo conférence.

La norme MPEG4 utilise une description de la scène à l'aide d'une syntaxe normalisée (BFSD : *Binary Format of Scene Descripsion*).

I.5. Compression conservatrice et non conservatrice

Après ce survol historique, on peut voir que la compression conservatrice (ou sans perte ou encore non destructive) et qui n'a pas pu dépasser les taux de 50%, a été rapidement abandonnée au bénéfice de la compression non conservatrice (ou avec perte ou encore destructive) et qui a pu atteindre des taux de l'ordre de 99.67 %. Néanmoins, après modification de l'image et l'élimination des informations superflues, une compression conservatrice peut être utilisée pour augmenter le taux de compression.

Les modifications que subissent les images d'entrée dans la compression destructive tirent leur origine des caractères psycho visuels résultant de l'œil humain [HERV 95, GALP 02, CRES 01] :

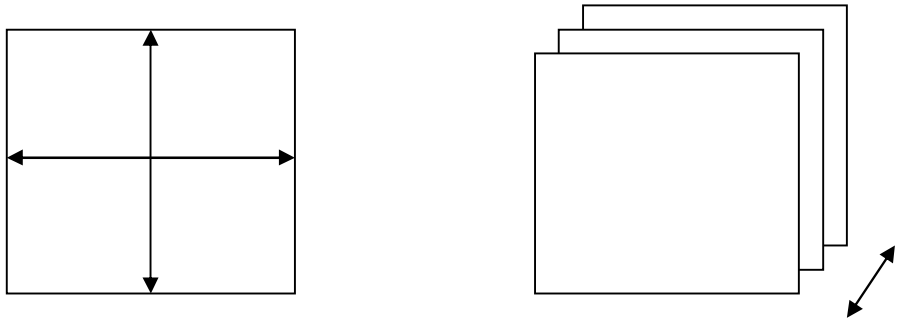
1. L'œil humain ne peut pas distinguer deux points séparés par une distance inférieure à une minute,
2. Il est sensible aux basses fréquences qu'aux hautes fréquences dans le cas du changement des couleurs,
3. Il est plus sensible à l'intensité lumineuse (luminance) qu'à l'intensité des couleurs (chrominance),

4. Il ne peut pas distinguer deux images séparées par moins d'un quinzième de seconde; en ajoutant le temps de persistance sur la rétine, on peut constater qu'une fréquence supérieure à 20 images par seconde est nécessaire pour donner l'impression d'animation.

Ces caractéristiques peuvent être exploitées à deux niveaux dans la compression des séquences vidéo (***Figure 5***) :

1. Niveau spatial : utilisé au niveau d'une seule image prise indépendamment des autres. Les techniques de la compression des images fixes sont utilisées.

2. Niveau temporel : utilisé au niveau d'une série d'images en analysant le changement dans les images et en ne codant que ce changement.



Compression spatiale **Compression temporelle**

Figure 5: Compression spatiale et temporelle

Il est clair que les caractéristiques de la corrélation spatiale et les caractéristiques de la corrélation temporelle sont très différentes, ce qui a conduit à des procédés séparés pour chaque type.

Les techniques de décorrélation spatiale sont appelées codage Intraframe et les techniques de décorrélation temporelle sont appelées codage Interframe.

Les données vidéo sont organisées d'une façon hiérarchique. Un fichier vidéo (une séquence) commence par une entête suivie d'un ou plusieurs groupes d'images suivi d'un code de fin. Chaque groupe d'images débute également par une entête et comporte une ou plusieurs images.

Chaque image est composée de tranches (lignes) et chaque tranche est composée de macro-blocs composés à leur tour de blocs de 8X8 pixels (***Figure 6***).

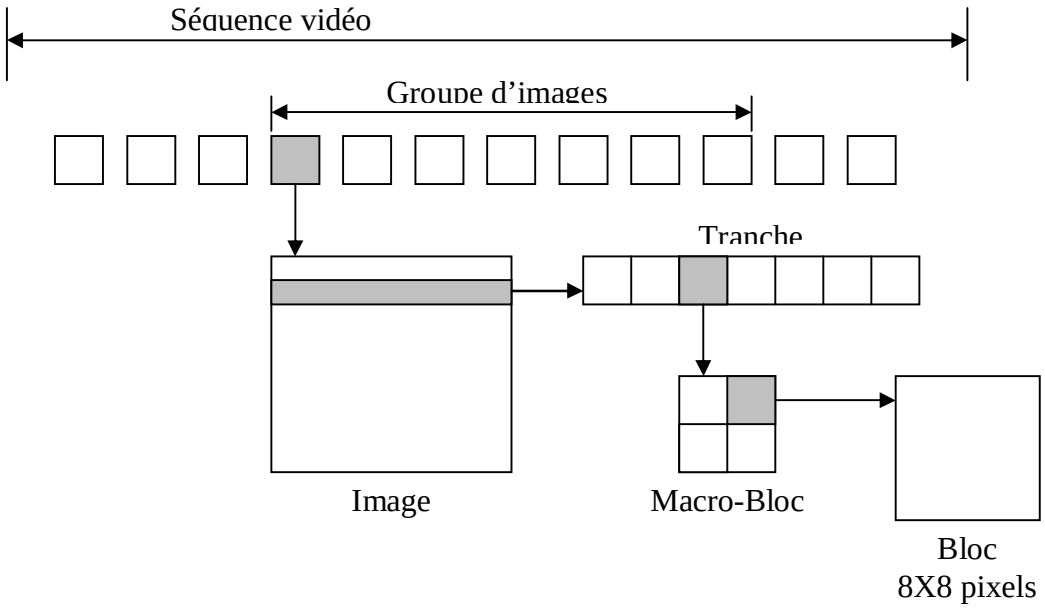


Figure 6:Composition d'une séquence vidéo [VLAN]

I.6. Codage IntraFrame

Pour mieux comprendre ce type de codage, prenons le cas concret de la compression JPEG (Joint Photo Expert Group) utilisée dans l'encodeur réalisé. Cette dénomination provient du groupe d'experts internationaux qui a établi en 1991 la norme que nous utilisons actuellement. Le codage JPEG se fait selon les étapes suivantes (***Figure 7***) :

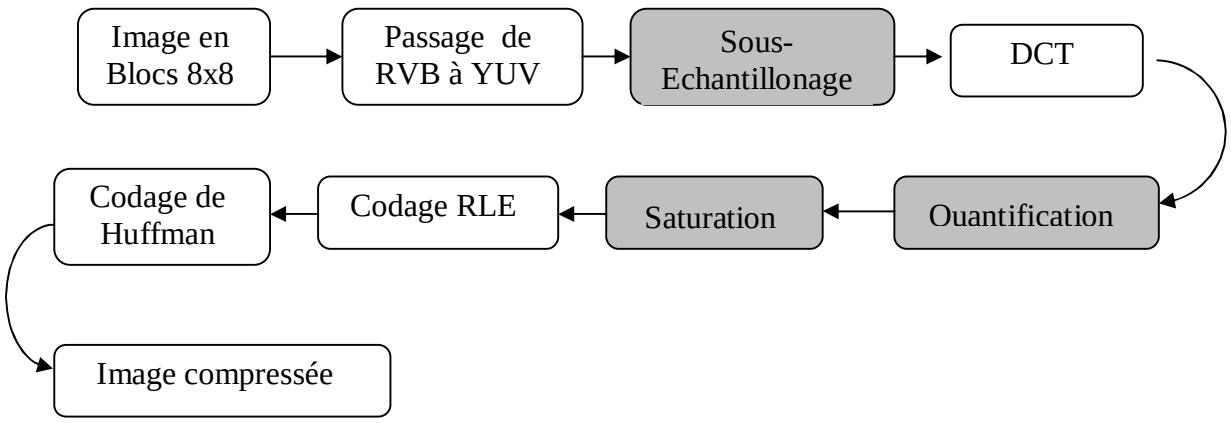


Figure 7:Principe du codage JPEG [JAIN 02]

Le décodage se fait en inversant les étapes du codage. C'est dans les étapes en surbrillance que l'image subit une modification de son contenu.

I.6.1. Passage du système RGB au système YUV

L'œil humain est sensible à la luminance qu'à la chrominance; cependant, la luminance existe dans les trois composantes de couleurs RVB. Donc, le système RVB n'est pas optimal pour la compression. On utilise alors une transformation au système YUV ou Y est la composante de luminance contenant l'essentiel des informations de l'image et U et V sont les composantes de chrominance bleue est rouge respectivement [SHAO 97]. L'exemple suivant (Figure 8) montre l'importance de chaque matrice (ou l'autre est fixée à 128).

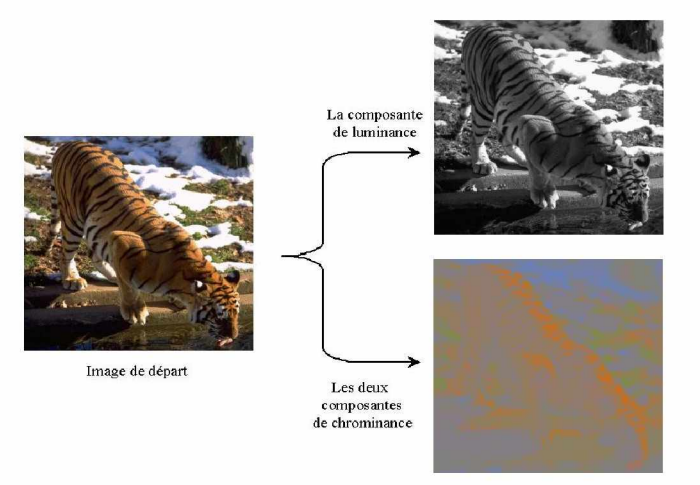


Figure 8:Effet de la transformation YUV

Cette étape permet ensuite d'effectuer une quantification plus intense sur les composantes de chrominance.

Le passage du RVB au YUV se fait selon le système suivant:

$$\begin{matrix} Y \\ U \\ V \end{matrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 & 0 \\ -0.1687 & -0.3313 & 0.5 & 1 \\ 0.5 & -0.41874 & -0.0813 & 1 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \\ 128 \end{bmatrix}$$

Le passage inverse se fait selon le système suivant

$$\begin{matrix} R \\ V \\ B \end{matrix} \begin{bmatrix} 1 & 1.402 & 0 \\ 1 & -0.34414 & -0.71414 \\ 1 & 0 & 1.772 \end{bmatrix} \times \begin{bmatrix} Y \\ U-128 \\ V-128 \end{bmatrix}$$

Dans la littérature, plusieurs travaux ont été élaborés et visent à trouver des systèmes de passage RGB-YUV sans perte lors de la récupération des RGB [SHAO 97, JAIN02]. Des algorithmes sont développés pour réaliser ce passage.

I.6.2. Sous-Echantillonnage

Dans cette étape, on néglige le plus d'informations possibles des matrices de chrominance U et V en les divisant par deux. On prend les matrices 8x8 de ces composantes et on les transforme en matrices 4x4 en calculant les moyennes des éléments quatre à quatre (***Figure 9***).

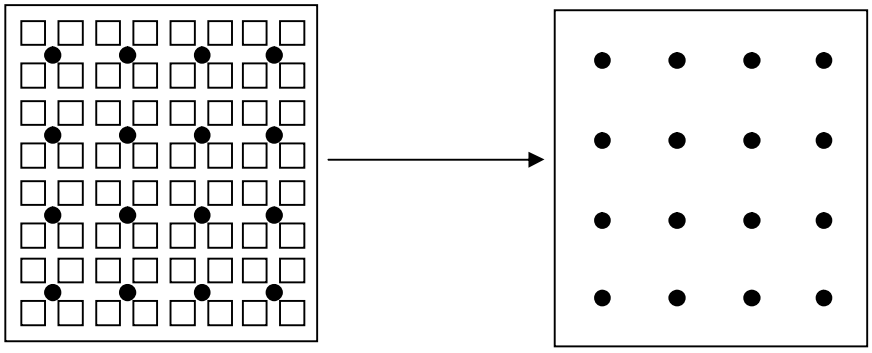


Figure 9: Sous-echantillonage des matrice U et V

Cette étape permet de diminuer la taille de l'image par deux sans introduire des changements visibles.

Pour reconstruire les composantes U et V, il suffit de prendre un élément de la matrice 4x4 et de remplir quatre cases de la matrice U ou V correspondante.

I.6.3. Transformation DCT

La DCT (Discrete Cosinus Transform) sur laquelle est basée la norme JPEG a été proposée en 1974 par le professeur Rao de l'université du Texas. C'est une transformation de Fourier qui permet de transformer un signal d'entrée (représenté en trois dimensions : deux dimensions de l'écran et une dimension représentant les couleurs de pixels) et le décomposer en un ensemble de signaux ; chaque signal contient une fréquence spatiale. L'amplitude de ces signaux est donnée par le résultat de la DCT [NELS 93, HERV 95, JAIN 02].

Après la transformation, on remarque que les basses fréquences de l'image sont concentrées au coin gauche supérieur, ce qui permet d'éliminer les hautes fréquences peu significatives pour la vision (***Figure 10***).

La transformation DCT est donnée par la formule :

$$DCT[i,j] = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \text{Image}(x,y) \cos\left(\frac{(2x+1)i\pi}{2N}\right) \cos\left(\frac{(2y+1)j\pi}{2N}\right)$$

$$\text{Avec } C(x) = \begin{cases} 1/\sqrt{2} \rightarrow x = 0 \\ 1 \rightarrow \text{sinon} \end{cases}$$

La transformation inverse IDCT (Inverse DCT) est donnée par la formule :

$$Image(x,y) = \frac{1}{\sqrt{2N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C(i)C(j)DCT[i,j] \cos\left(\frac{(2x+1)i\pi}{2N}\right) \cos\left(\frac{(2y+1)j\pi}{2N}\right)$$

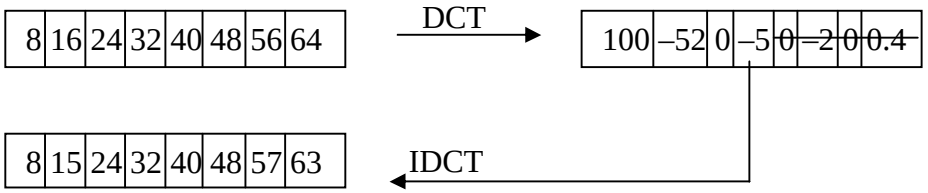


Figure 10: Exemple de la DCT

Le calcul de la DCT est une boucle imbriquée de N^4 itérations. Si on prend la DCT de l'image toute entière, le calcul devient rapidement insupportable avec l'augmentation des dimensions de l'image. Pour éviter cela, on divise l'image en blocs de 8x8 (taille choisie par le groupe JPEG).

La DCT est l'opération la plus gourmande en temps de calcul dans le codage JPEG. Plusieurs techniques ont été développées pour optimiser le temps de calcul [TOUR 00]:

1. Ne calculer le produit des deux cosinus qu'une seule fois et le stocker dans une table avant de commencer le codage.
2. Effectuer les calculs sur les colonnes puis sur les lignes (***Figure 11***).

$$DCT[u,v] = \frac{1}{2} \sum_i C(u) Cos(\frac{(2i+1)u\pi}{16} G[i,v]$$

$$G[i,v] = \frac{1}{2} \sum_i C(v) Cos(\frac{(2j+1)u\pi}{16} \mathrm{Im}age[i,j]$$

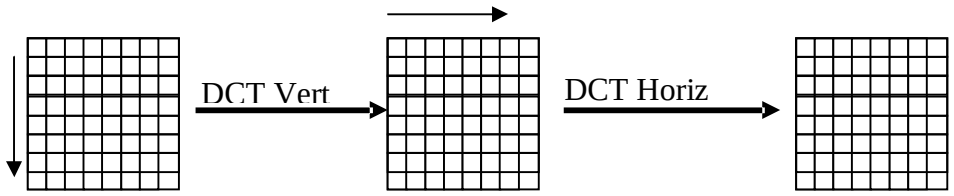


Figure 11: DCT horizontale et verticale

Des optimisations sont également développées et consistent à n'utiliser que des décalages et des multiplications. La meilleure optimisation actuelle est de 11 multiplications et 29 additions.

I.6.4. Quantification

Après avoir rassemblé les basses fréquences dans le coin gauche supérieur da la matrice DCT, on peut maintenant éliminer les hautes fréquences. On multiplie les valeurs de la DCT par des coefficients permettant de négliger ces éléments. La table de ces coefficients est appelée: table de quantification.

Les valeurs choisies de cette table déterminent la qualité de l'image décodée. Une valeur de 1 pour toute la table donne la meilleure qualité.

Bien que la spécification JPEG n'impose aucune contrainte sur la table de quantification, l'organisme de standardisation ISO a développé un ensemble standard de tables utilisées par les programmeurs du code JPEG [JAIN 02]. Les tables les plus répandues sont celles permettant de choisir la perte de qualité acceptable. Habituellement, on prend la matrice :

$$Q = Qij \qquad \text{Avec:} \qquad Qij = \frac{1}{1 + k(1 + i + j)}$$

k étant un facteur de qualité choisi entre 1 et 255.

La table résultante de la quantification est appelée table quantifiée.

On choisit aussi deux matrices de quantification différentes, une pour la luminance et l'autre pour la chrominance. Les coefficients de la deuxième sont plus élevés que ceux de la première à cause de la sensibilité à la luminance de l'œil humain.

L'étape de quantification est la deuxième étape de perte d'informations après le sous échantillonnage Le tableau suivant illustre un exemple de la DCT et de la quantification.

Matrice de pixels d'entrée

140	144	147	140	140	155	179	175
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

Matrice DCT

1210	-18	15	-9	23	-9	-14	-19
21	-34	26	-9	-11	11	14	7
-10	-24	-2	6	-18	3	-20	-1
-8	-5	14	-15	-8	-3	-3	8
-3	10	8	1	-11	18	18	15
4	-2	-18	8	8	-4	1	-7
9	1	-3	4	-1	-7	-1	-2
0	-8	-2	2	1	4	-6	0

Matrice de quantification

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

Matrice DCT quantifiée

403	-4	2	-1	2	-1	-1	-1
4	-5	3	-1	-1	1	1	0
-1	-3	0	0	-1	0	-1	0
-1	0	1	-1	0	0	0	0
0	1	1	0	-1	1	1	1
0	0	-1	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Matrice DCT déquantifiée (décompression)

1209	-20	14	-9	22	-13	-15	-17
20	-35	27	-11	-13	15	17	0
-7	-27	0	0	-15	0	-19	0
-9	0	13	-15	0	0	0	0
0	13	15	0	-19	21	23	25
0	0	-17	0	0	0	0	0
15	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Matrice de pixels de sortie (décompression)

142	143	154	141	133	153	179	179
139	152	129	151	144	154	163	181
150	156	139	166	162	163	154	172
163	145	160	153	151	153	145	154
168	150	156	145	140	139	141	159
148	164	133	164	158	140	136	163
130	159	123	164	165	140	134	145
148	156	140	148	159	146	153	141

Figure 12: Exemple de la DCT et de la quantification

I.6.5. Saturation

L'étape de quantification peut produire des valeurs supérieures à un octet (255). Ces valeurs doivent être arrondies pour qu'elles puissent être stockées sur un octet, sauf pour la première valeur (0,0) qui est proportionnelle à la moyenne de la matrice; cette valeur doit être stockée sur plusieurs octets selon sa valeur.

En réalité, les valeurs dépassant un octet apparaissent dans les images contenant un nombre réduit de couleurs telles que les images en noir et blanc.

I.6.6. Codage RLE (Run Length Encoding)

Le codage RLE ou codage en longueur de séquence est une méthode de compression très simple, elle consiste à remplacer une séquence de caractères identiques « AAAAAAA » par le nombre de caractères suivi du caractère lui même « 7A ». Ce codage convient pour la matrice quantifiée qui contient plusieurs Zéros. Pour obtenir des séquences de Zéros les plus longues possibles, on doit d'abord effectuer un codage différentiel au niveau des DC (Differential Components) (0,0) des blocs, ensuite lire la matrice quantifiée en Zigzag (*Figure 13*).

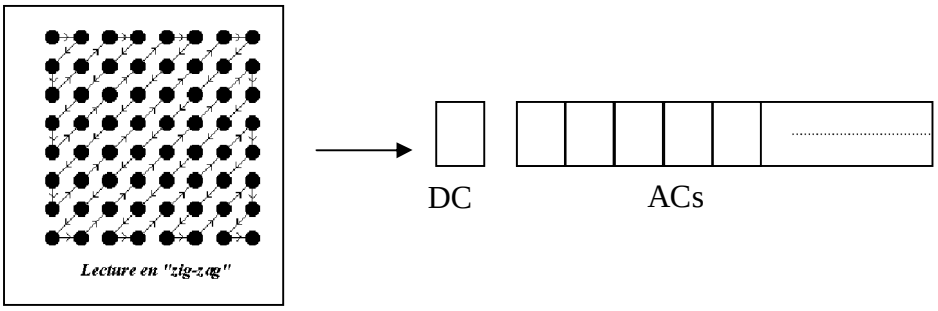


Figure 13: Codage en longueur de séquence

Les valeurs du bloc sont codées comme suit :

(N, val): N est le nombre de Zéros et val la première valeur non nulle qui suit.

(0,0): indicateur de fin de bloc.

I.6.7. Codage de Huffman

Le codage de Huffman, qui date des années 50, repose sur une analyse statistique préalable des données à compresser [NELS 93, HERV 95, JAIN 02]. A l'issue de cette analyse, un arbre est construit et permet d'attribuer à chaque symbole un code dont le nombre de bits est inversement proportionnel à la probabilité d'apparition du symbole. Ainsi, les symboles les plus fréquents se voient affectés aux codes plus courts, et les codes les plus longs sont attribués aux symboles rares. En outre, ces codes sont séparables, c'est-à-dire qu'un code donné ne peut pas être le

préfixe d'un autre code. Bien sûr, pour permettre au décodeur de reconnaître les symboles, il est nécessaire de lui transmettre le dictionnaire obtenu après la phase d'analyse statistique des données.

L'algorithme de construction de l'arbre de Huffman est illustré par l'exemple suivant:

Soit à coder le message :

"EXEMPLE DE CODAGE DE HUFFMAN".

L'algorithme commence par calculer la probabilité de chaque symbole :

(E,6/28), (X,1/28), (M,2/28), (P,1/28), (L,1/28), (espace,4/28), (D,3/28), (C,1/28), (O,1/28), (A,2/28), (G,1/28), (H,1/28), (U,1/28), (F,2/28), (N,1/28).

A chaque étape, les deux symboles de plus petite probabilité sont regroupés en un nouveau symbole ayant pour probabilité la somme des deux probabilités. Le processus est itéré jusqu'à obtenir un unique symbole de probabilité 1 (*Figure 14*).

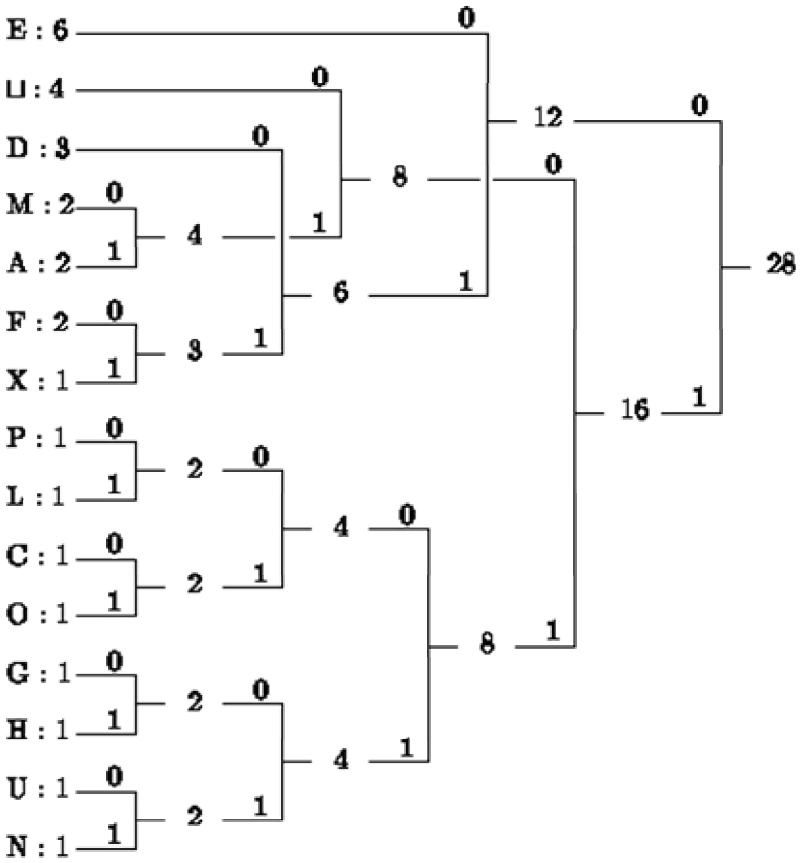


Figure 14: Codage de Huffman

Le code de chaque symbole est ensuite obtenu en suivant le chemin depuis la racine de l'arbre de Huffman jusqu'à la feuille contenant le symbole. Un bit est ajouté à chaque embranchement : 0 si le chemin emprunte la branche supérieure, 1 si le chemin emprunte la

branche inférieure. Ainsi, le code du symbole le plus fréquent (E : 00) est codé sur deux bits, tandis qu'un symbole rare (H : 11101) est codé sur 5 bits.

I.7. Codage InterFrame

C'est le codage dans le domaine temporel, il démarre du principe qu'il n'y a pas une grande différence entre deux images qui se suivent dans la même séquence. Il consiste à coder les éléments (blocs, objets, régions, contours,…) de l'image courante, simplement comme des adresses des éléments correspondants dans une image de référence déjà codée. Selon la ou les références utilisées (images précédentes ou suivantes), on trouve deux modes de codage Prédictif et Bidirectionnel.

I.7.1. Codage Prédictif

Il consiste à coder une image par rapport à l'image précédente. On prend donc l'image courante (cible) bloc par bloc et on cherche dans l'image précédente (référence) les blocs identiques. La différence de position entre les deux blocs (cible et référence) est appelée *Vecteur de mouvement*. La différence entre les deux blocs est codée ensuite comme image Intra (***Figure 15***).

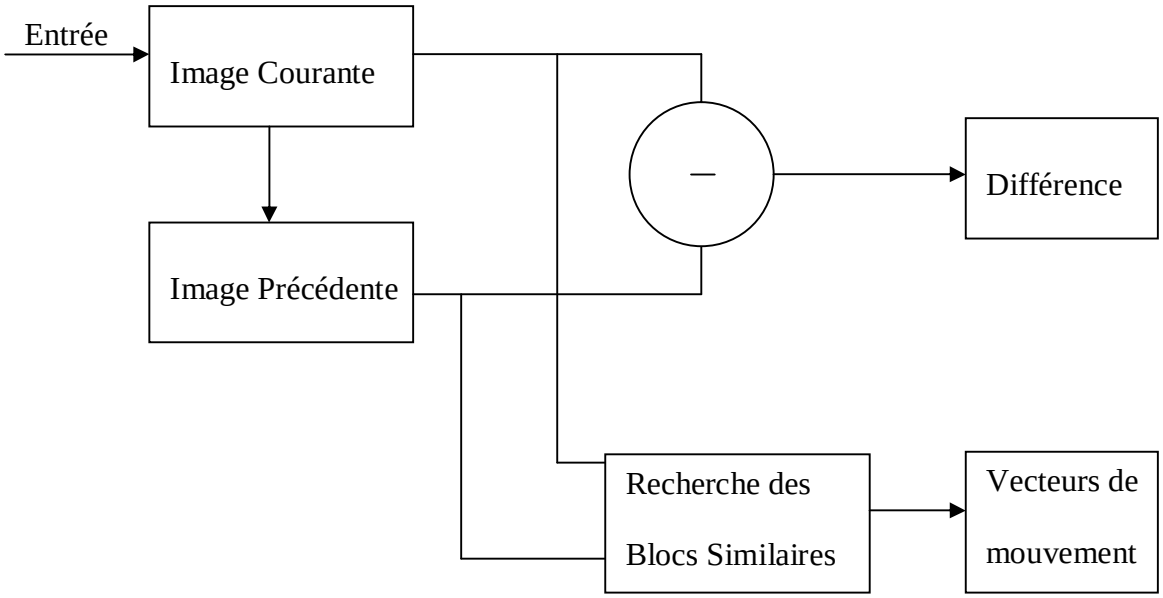


Figure 15: Codage par différence

Mais au niveau du décodeur, ce n'est pas l'image précédente réelle qui est utilisée pour la décompression, mais plutôt l'image décompressée [CHOK 98, MOSC 01, CRES 01]. Donc, pour minimiser l'erreur de calcul on doit utiliser dans le codeur la même image utilisée lors du décodage, c'est à dire décoder l'image précédente au niveau du codeur (***Figure 15***).

La recherche des blocs similaires dans l'image de référence est appelée *Estimation de mouvement*, et la reconstitution de l'image codée est appelée *Compensation de mouvement*. L'opération globale est appelée *Codage par Estimation et compensation de mouvement* (***Figure 16***).

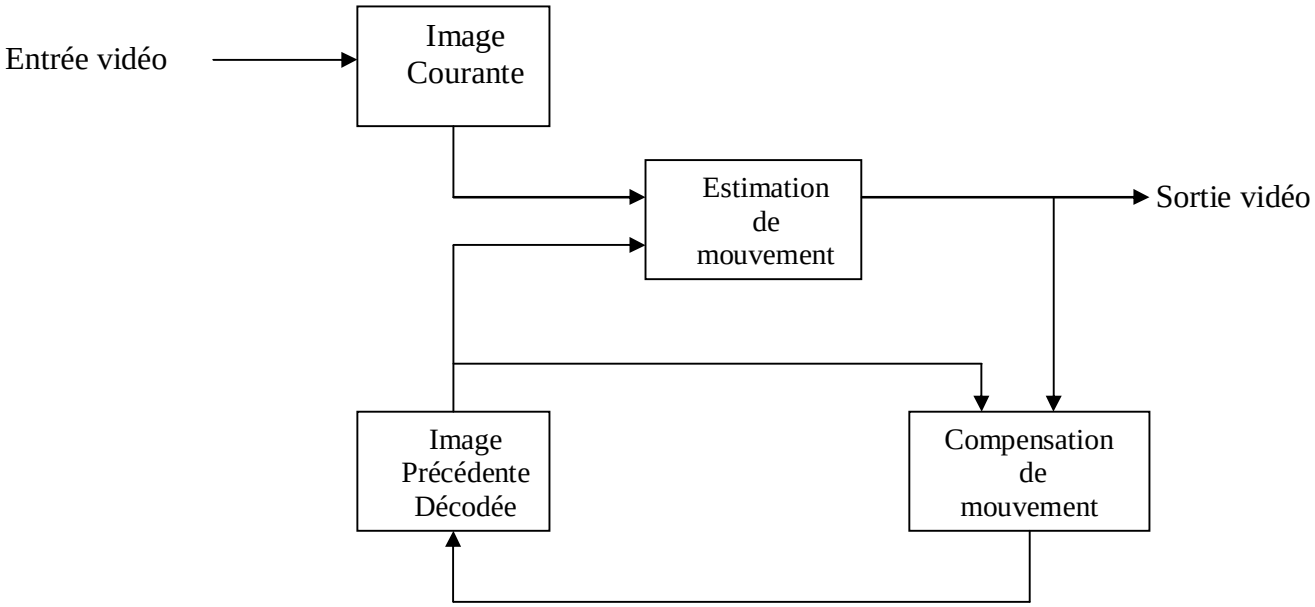


Figure 16: Compression vidéo par estimation et compensation du mouvement

L'opération d'Estimation de mouvement est une étape fondamentale dans l'encodeur, elle détermine dans une grande partie sa vitesse, sa précision et son taux de compression.

I.7.2. Codage Bidirectionnel

La compression en utilisant les images P est très puissante, mais elle ne résoud pas tous les problèmes de la compression vidéo. Considérons par exemple une scène où un personnage ouvre une porte. Il n'y a aucun moyen de prédire les détails de la pièce qui se trouve derrière la porte avec l'image précédente où la porte était fermée. D'où la nécessité de rechercher des blocs même dans les images futures (***Figure 17***) et d'où la nécessité du codage bidirectionnel [CRES 01, ITU 93, HERV 95].

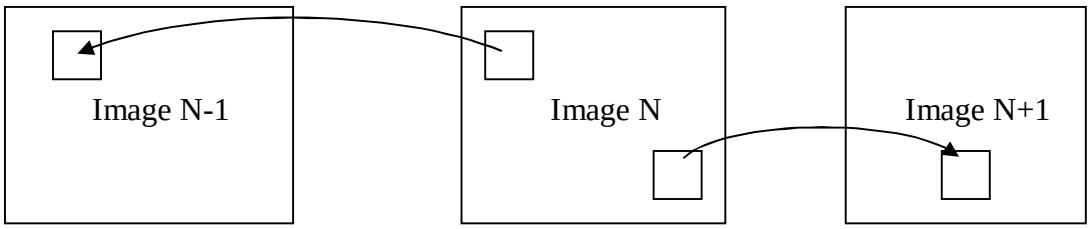


Figure 17: Codage bidirectionnel

Les images résultantes du codage bidirectionnel sont appelées *Images B.*

Ce principe a été introduit dans la norme MPEG1, il a montré une très grande puissance. Une séquence IBBP consomme 60% d'espace moins qu'une séquence IIII, mais le temps de calcul sera plus élevé [CRES 02].

Les images B peuvent se référer soit des images I, soit des images P, soit des deux types, mais jamais d'une autre image B (***Figure 18***) ce qui limite la propagation de leurs erreurs.

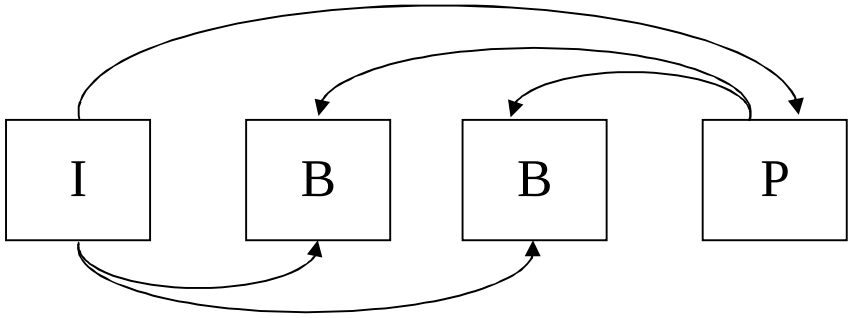


Figure 18: Références du codage bidirectionnel

C'est la nature de la séquence qui détermine la fréquence des images I, P et B. Dans les séquences de changement rapide, les images I sont plus fréquentes (3 par seconde par exemple) telles que les séquences sportives, ou une par seconde (ou moins) dans les séquences lentes telles que les séquences de vidéoconférence. Donc, c'est à l'encodeur de choisir au moment du codage (ou même avec une intervention de l'utilisateur) de tels paramètres critiques.

Puisque les images B se réfèrent des images futures, l'ordre d'envoi des images doit être revue, pour éviter l'inactivité du décodeur en attendant des images clés. Par exemple, Dans la norme MPEG, on envoie les images I et P puis les images B (***Figure 19***).

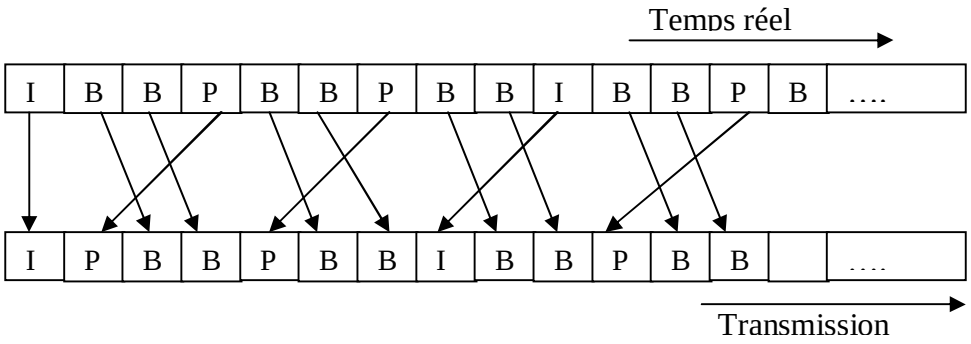


Figure 19: Ordre d'émisssion des images dans le codage bidirectionnel [CRES 01]

I.8. Schéma général d'un encodeur vidéo.

Le schéma illustré en (***Figure 20***) est basé sur les principes précités, et c'est le schéma adapté par la plupart des standards de compression vidéo telles que les normes H26x et MPEGx [WING 02]. Il utilise le principe de compensation de mouvement des blocs et leur transformation ensuite par DCT.

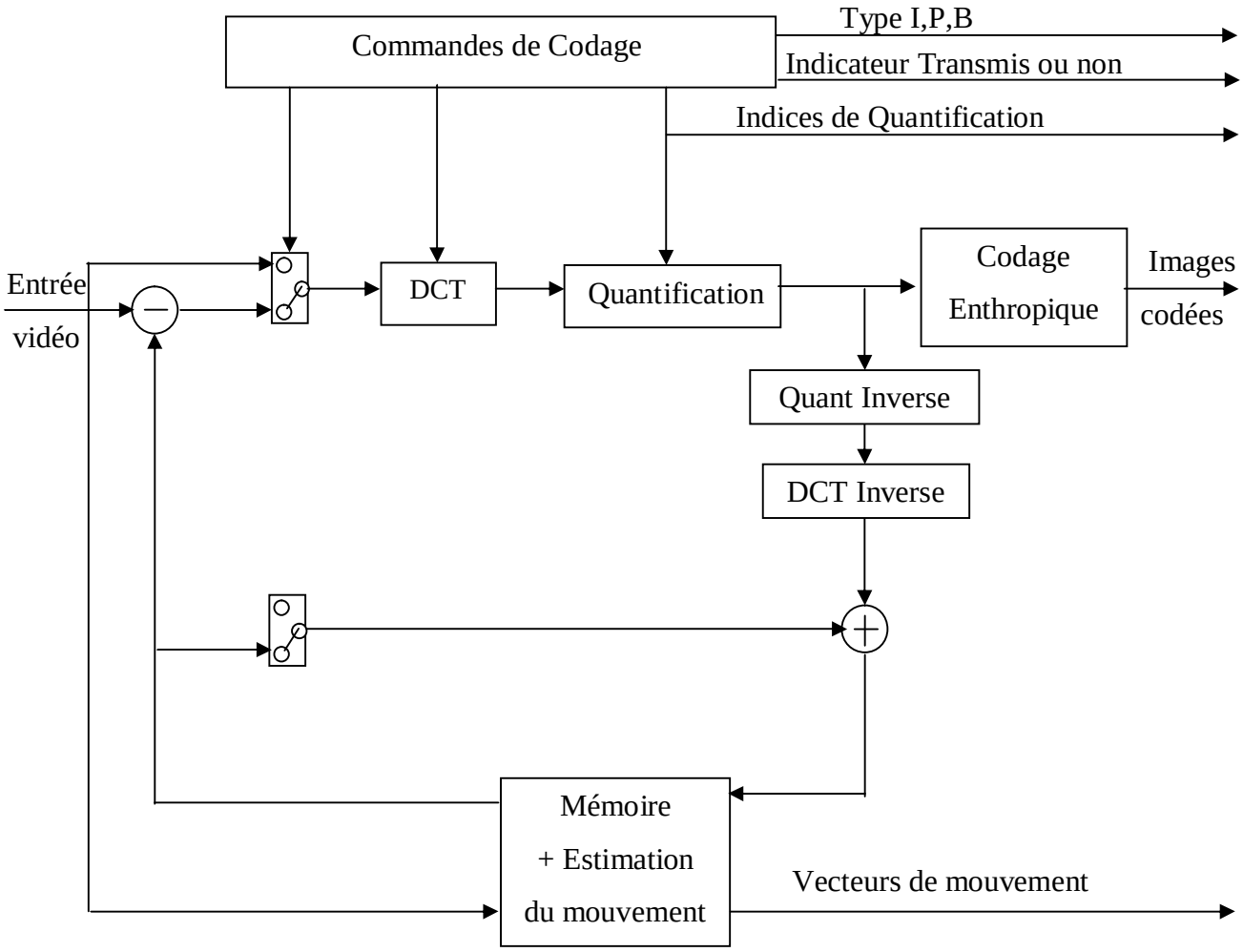


Figure 20: Scéma général d'un encodeur vidéo

Les données produites par cet encodeur sont présentées sous la forme (*Figure 21*) [VCOMP]:

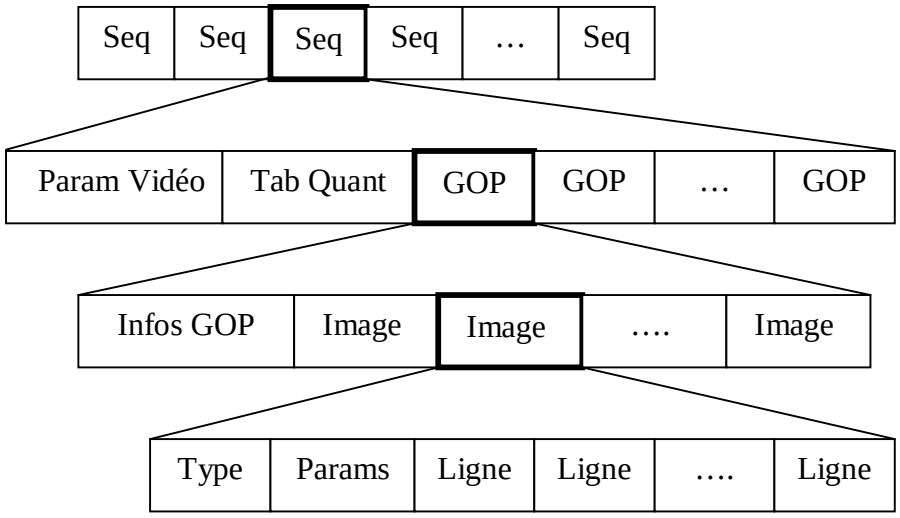


Figure 21: Flux des données vidéo

Le format des données vidéo produites représente les caractéristiques de la norme utilisée; mais, en général les données vidéo ont à peu près ce format. Le flux est composé de séquences, où chaque séquence débute par une entête contenant les paramètres vidéo telle que la taille des images et leur nombre, puis les tables de quantifications pour les images intra et inter, suivies d'un ensemble de groupe d'images GOP. Le GOP débute à son tour par une entête contenant des informations sur le nombre d'images et leur fréquence. Chaque image commence par son type I, P ou B et les paramètres nécessaires pour chaque type.

I.9. Normes et produits

Deux grandes normes sont dans le marché du codage vidéo: la norme H26x et la norme MPEGx. La première développée par la CCITT et la deuxième par ISO. Ce sont les objectifs du codage qui déterminent la structure et les principes de l'encodeur. La norme H26x est destinée initialement à la vidéo conférence qui est caractérisée par des mouvements lents et des arrières plans statiques, tandis que la norme MPEGx est destinée aux applications de la vidéo numérique et de l'informatique ainsi qu'aux applications de la télévision numérique.

I.9.1. Norme H261

Cette norme a été développée par le CCITT en 1990 pour l'appliquer en vidéo conférence et en visiophonie sur les réseaux RNIS (Réseau Numérique à Intégration de Service) (ISDN). Elle offre une transmission des images animées sur des supports à multiplexage de 64 Kbit/s. Elle utilise des séquences de 30 images par seconde de type I et P. Les images I sont codées selon le

principe JPEG déjà cité (Fig. 7). Les images P sont codées selon le principe de compensation de mouvement [ITU 93] (***Figure 22***,***Figure 23***).

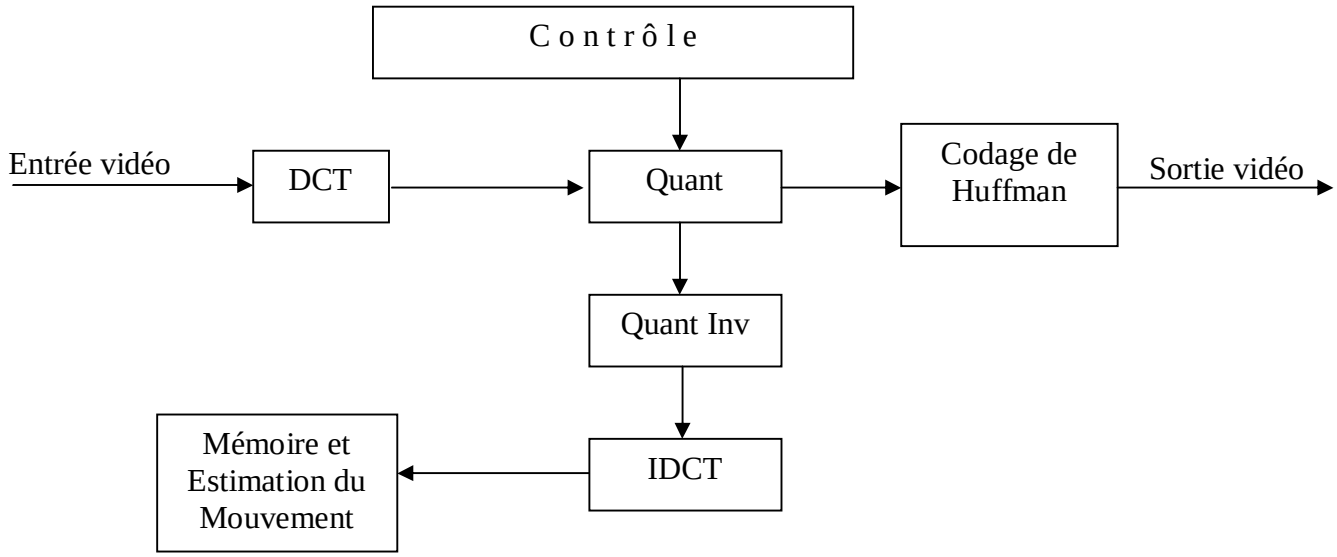


Figure 22: Compression des images I dans H261

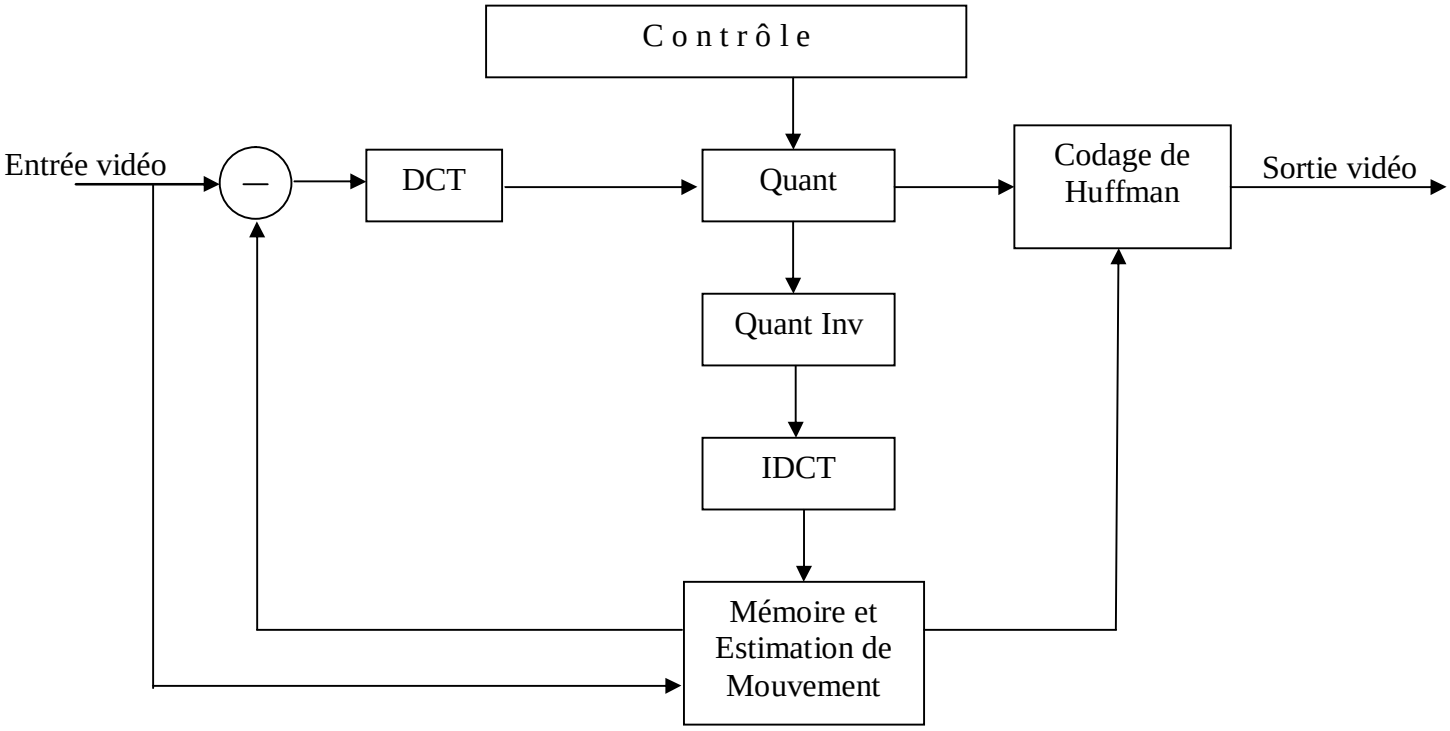


Figure 23: Compresssion des images P dans H261

I.9.2. Norme H263

C'est un standard de l'ITU (International Telecom Union), développé en février 1995. Il a été désigné aux communications à faible débit (moins de 64 Kbit/s) [H26x]. Mais le standard a été ensuite développé pour être utilisé dans des applications à débit élevé et a pu remplacer le

H261 dans plusieurs applications surtout dans la vidéo conférence et a dominé le domaine de la vidéo sur internet.

Dans sa structure, le H263 ressemble au standard H261 à moins qu'il utilise une précision d'un demi pixel (III.4.10) au lieu du pixel tout entier, et plusieurs parties du flux de données fournies sont rendues optionnelles pour s'adapter aux faibles débits, et enfin il utilise une prédiction bidirectionnelle P et B (image précédente et suivante).

I.9.3. Norme H26L

Cette Norme, prévue lors de son développement pour devenir la norme internationale du codage vidéo en début de 2003, utilise un système de codage par compensation de mouvement basé sur des blocs et la transformation DCT [WING 02].

La norme H26L utilise des images I, P et B. Pour les images I, elle utilise des techniques avancées d'extrapolation des blocs pour obtenir des taux semblables à ceux de la norme JPEG2000, en réalisant des codages différentiels horizontaux, verticaux et diagonaux au niveau des blocs pour augmenter le nombre de Zéros.

La recherche des blocs similaires est effectuée selon une interpolation jusqu'à un huitième de pixel (recherche des blocs interpolés dans des blocs 8x des blocs réels).

La recherche peut être effectuée jusqu'à des blocs 4x4 ce qui diminue considérablement le temps de recherche.

Les images sont codées en entrelacement (lignes impaires puis paires) pour s'adapter avec la télévision.

Une nouvelle technique pseudo-DCT est utilisée sur des blocs 4x4 de la luminance, d'une inversion très parfaite (sans disparité) et très rapides.

Un Codage de longueur variable CLV avec des tables universelles est utilisé au lieu du codage de Huffman utilisé dans les normes précédentes.

La norme H26L semble profiter du développement important des microprocesseurs et des mémoires pour augmenter les taux de compression.

I.9.4. Norme MPEG1

La norme MPEG1, enregistré à l'ISO sous le code ISO/IEC 11172, a été finalisée par le groupe MPEG en 1992. Son but principal est de stocker et reproduire de la vidéo sur des supports de stockage en qualité magnétoscope (320 X 240) avec un débit maximum égal à 1,5 MBits/s. Effectivement, MPEG1 est devenue par la suite la norme de stockage vidéo sur CD-ROM ou CD-Vidéo [VLAN].

Le principe de cette norme est décrit dans la section InerFrame de ce chapitre.

I.9.5. Norme MPEG-2

Elle est enregistrée à l'ISO sous le code ISO/IEC 13818; elle à été finalisée en 1996. Son principe de codage est semblable à celui du MPEG-1 sauf qu'elle utilise des images entrelacées (lignes paires puis lignes impaires), contrairement à MPEG-1 qui utilise des images en mode progressif uniquement. Elle produit aussi des vidéos de haute résolution (jusqu'à 1920 X 1152) à une fréquence de 30 images par seconde. Elle utilise aussi une prédiction basée sur des champs (partie de l'image) pour augmenter la vitesse [KEES 96].

La norme MPEG2 permet un codage facultatif en deux passes: une passe d'analyse qui permet d'accumuler des statistiques sur la complexité de la séquence pour les utiliser dans la deuxième passe pour maximiser le taux de compression.

La norme MPEG-3 a été rapidement abandonnée et intégrée dans MPEG-2.

I.9.6. Norme MPEG-4

La norme MPEG4 est enregistrée à l'ISO sous le code ISO/IEC 14496. Cette norme représente une révolution dans le domaine de la compression vidéo. Elle vise à fusionner trois domaines : l'informatique, la télécommunication et la télévision, ses domaines d'application sont très diverses: communication temps réel (visiophone), multimédia mobile, téléconférences, post-production, stockage DVD et recherche d'information basée sur le contenu [MPG4, RICH 01].

La norme MPEG-4 est radicalement différente des normes précédentes. Elle initialise une vidéo orientée objets : une séquence est une scène hiérarchisée en objets média, chaque objet est composé d'un ensemble de paramètres (son , forme, position,…).

Différents types d'objets sont utilisés dans cette norme telles que les images fixes (arrière plan), les objets vidéo (objets en mouvement), les objets audio (voix associée) (***Figure 24***). Ce principe rend cette norme très flexible et évolutive et permet toute une gamme de nouvelles possibilités telles que l'ajout et la suppression des objets, les transformations géométriques, les changements de point de vue, …etc.

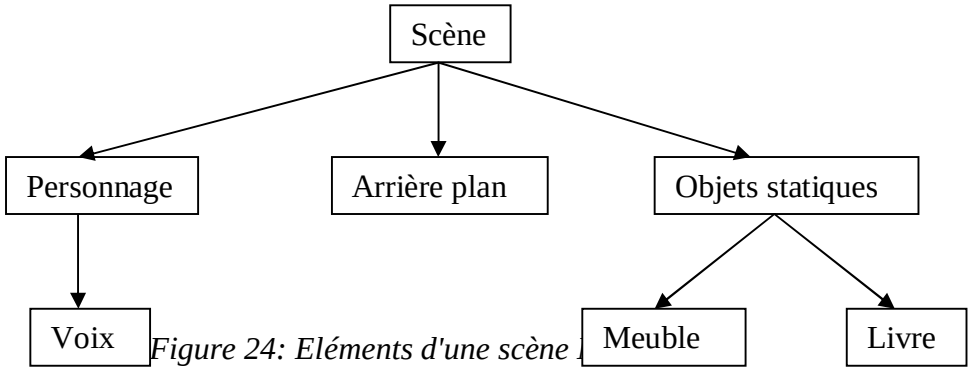


Figure 24: Eléments d'une scène

Les techniques de compression utilisées dans les normes précédentes (DCT, Quantification, RLE, Huffman) sont utilisées ici aussi, mais adaptées aux nouveaux objets.

Plusieurs types d'objets sont définis dans la norme MPEG-4 tels que AVO (Audio Video Object), VOP (Video Object Plan), I-VOP (Intra VOP), P-VOP (Predicted VOP), B-VOP (Bidirectionel VOP), etc. Le même principe d'estimation et de compensation de mouvement est également utilisé.

La norme MPEG-4 est encore en cours de développement; sa deuxième version a été finalisée en 1999 uniquement comme un standard de la visioconférence et du multimédia mobile à un débit de 10 images par seconde.

I.9.7. Norme MPEG-7

La norme MPEG-7 est enregistrée sous le code ISO/IEC 15938, elle est en cours d'élaboration depuis 1997, son objectif principal est d'optimiser et de simplifier les fichiers multimédia, elle ajoute une couche à la norme MPEG-4 décrivant les objets vidéo pour faciliter leur recherche et donnant des informations concernant la séquence tel que l'auteur, la date de création, le genre de la séquence,… etc. Une fois cette norme finalisée, elle touchera plusieurs domaines tel que le journalisme, le commerce électronique, les services éducatifs et culturels [CRES 01].

I.9.8. Norme MPEG-21

Elle est enregistrée sous le code ISO/IEC 18034. Elle est en cours de développement depuis 2000. Elle vise à ajouter une couche de sécurité à la norme MPEG-7.

I.9.9. Autres Normes

MPEG n'a pas le monopole du marché de la compression vidéo. Quick Time développé en 1991 par Apple est un environnement complet de développement orienté vidéo. Il utilise un principe de compression appelé *Indeo* développé par Intel.

I.10. Caractéristiques principales d'un Encodeur vidéo

Les encodeurs vidéo varient selon les objectifs de développement: pour transmission, stockage, traitement en temps réel, …etc. Selon l'objectif visé, varie l'importance des différentes caractéristiques de l'encodeur : taux de compression, temps de codage, qualité des images fournies, résistances aux erreurs.

I.10.1. Taux de compression

C'est le but fondamental de la compression, il est mesuré par le rapport entre le volume des données initiales et celui des données codées. Plus le taux de compression augmente, plus l'espace nécessaire pour le stockage diminue, plus le temps nécessaire pour la transmission diminue.

Le type d'images choisies pour la compression influe considérablement sur le taux de compression. Les images B sont les moins coûteuses en terme d'espace (***Table 1***).

Table 1: Taux de compresssion par type d'image dans MPEG1 [CRES 01]

Type d'image	Taux de compression
I	≈ 85 %
P	≈ 95 %
B	≈ 98 %

Dans les images P et B, plus l'estimation du mouvement est précise; moindre est la différence entre les blocs et meilleur est le taux de compression.

Le taux de compression est calculé par la formule:

$$Taux(\%) = 100 - \frac{Taille\_données\_codées}{Taille\_données\_initiales} \times 100$$

D'autres mesures utilisent le nombre de bits utilisés pour coder un pixel:

$$Taux = \frac{Nbre\_bits\_données\_codées}{Nombre\_Pixels} (Bits / Pixels)$$

I.10.2. Durée de compression

Pour les applications de codage en temps réel telles que les caméras numériques, les mobiles de troisième génération, le temps de codage est très important. Le codeur doit fournir des images codées à une fréquence minimale de 20 images par seconde, ce qui implique que le codage d'une image en moyenne est de 50 ms pour obtenir des images en temps réel.

Le temps de calcul est toujours relatif à la fréquence d'entrée des images. Plus cette fréquence est basse, plus la durée disponible pour le codage est longue et le débit nécessaire pour la transmission est plus réduit.

Si la méthode de codage aboutit à une durée de codage inférieure à 50 ms par image, le temps restant peut être exploité pour réaliser des traitement sur les images tel que l'ajout d'éléments dans la scène ou changement de l'arrière plan en temps réel (si le medium de transmission le permet bien sûr).

Le tableau ci-dessous (Table 2) illustre le nombre d'opérations nécessaires par seconde pour chacune des étapes de codage :

Etape	Nombre d'opération (Millions) par seconde
Transformation RGB-YUV	27
Estimation de mouvement	608
Compensation de mouvement	113
DCT	60
Quantification, Zigzag	44
Codage Enthropique	17
Décompression	99

Table 2: Besoins en temps réel des différentes étapes du codeur H261 [CHOK 98]

L'estimation du mouvement joue un rôle principal dans ces applications, elle consomme plus de 60% du temps nécessaire.

Dans des applications en temps réel tel que la visiophonie et le vidéo-chat, l'encodeur exige une fréquence maximale de dix images par seconde pour éviter d'engorger le moyen de transmission à faible débit disponible.

I.10.3. Débit

Mesuré par le nombre de bits envoyés par seconde. C'est un paramètre très important pour les encodeurs visant un médium (ou des applications) de stockage ou de transmission supportant un débit limité. Dans ce cas l'encodeur doit ajuster ses paramètres pour s'adapter au débit supporté par le médium tout en gardant une fréquence acceptée des images. C'est un paramètre relatif à la vitesse du codage de l'encodeur: plus l'encodeur est rapide, plus le débit est élevé.

Par exemple, si on veut envoyer des images sur un médium de 64 Kbits/s on doit coder nos images de telle sorte qu'on puisse envoyer 25 images par seconde dans un volume de 64 kbits c'est-à-dire 8 KOctets, ce qui est équivalent à produire des images codées en moyenne sur 8/25 KOctets chacune en $1/25^{eme}$ de seconde.

Des encodeurs utilisent carrément des couches de contrôle de débit (Bitrate control) qui réalise un feedback à partir des buffers de transmission pour réajuster les tables de quantification et les paramètres (seuils d'estimation de mouvement) de telle sorte à dégrader la qualité (diminuer la quantité des données) en cas d'engorgement des buffers.

I.10.4. Qualité

La qualité doit être acceptable. Elle est souvent liée aux choix des tables de quantification et à la précision de l'estimation du mouvement. L'encodeur vise toujours la plus grande qualité possible selon le taux et la durée du codage visés, et par conséquent, les tables de quantification et les paramètres de l'estimation du mouvement sont ajustées en fonction de ces deux paramètres.

La qualité est matérialisée par le PSNR mesuré en Décibels (dB) et calculé selon la formule déjà citée (I.4.2).

Dans le cas des images couleurs, la différence est calculée à base des trois couleurs RGB: rouge, vert et bleu, et divisée ensuite par trois pour trouver l'erreur après le décodage de l'image. La formule précédente du PSNR s'écrit donc comme suit:

$$PSNR = 10Log10\left(\frac{255^2}{\frac{1}{N}\sum\left(\frac{|R-R'|+|G-G'|+|B-B'|}{3}\right)^2}\right)$$

La qualité visée a un impact direct sur le taux de compression du codeur et sa vitesse. Si la qualité demandée est importante, les valeurs des tables de quantification doivent être réduites pour minimiser la perte des informations ce qui augmente la taille des blocs codés que se soit pour les blocs des images intra ou inter. Pour les images P et B, les seuils de ressemblance doivent être soigneusement choisis, et la comparaison des blocs doit être la plus significative possible: c'est-à-dire qu'on doit comparer le maximum de points. D'autre part, on doit chercher les blocs dans un espace plus grand pour avoir plus de chances de trouver des blocs identiques, ce qui augmente considérablement le temps du codage.

I.10.5. Résistance aux erreurs

La stratégie de codage doit prendre en compte les cas de pertes des parties des données stockées ou transmises, et prévoir des techniques pour pouvoir continuer le décodage sans ces données, ou tirer au pire des cas l'essentiel des informations telles que les tables de quantification et les données de luminance. Des points clés (drapeaux) doivent être insérés de temps en temps dans les données fournies.

Les images I sont codées indépendamment des autres et par conséquent, elles permettent d'arrêter la propagation des erreurs qui peuvent apparaître dans le flux de données. La fréquence de ces images doit être soigneusement choisie selon le cas de la séquence (par exemple dans le cas des séquences rapides telles que les séquences sportives, trois à cinq images I sont codées par seconde).

I.11. Conclusion

Ce chapitre a été élaboré comme introduction à la compression vidéo; les points clés de ce codage ont été détaillés et les différentes étapes ont été décrites. Deux types de redondances dans les séquences vidéo ont été traitées : la redondance spatiale traitée par les techniques de codage des images fixes, et la redondance temporelle entre les différentes images de la séquence traité par plusieurs procédés dans la littérature allant de sa négligence totale (compression image par image) jusqu'à la vue globale de la scène. L'estimation de mouvement et sa compensation est l'une des grandes techniques utilisées et sera étudiée dans le chapitre suivant.

Les plus importantes normes développées ont été également citées ainsi que les techniques utilisées et les différences existantes entre elles.

Deuxième chapitre:

Estimation de mouvement pour la compression vidéo

II.1. Introduction

L'estimation du mouvement a prouvé son efficacité pour le codage vidéo. Elle est devenue un composant central de la plupart des standards de la compression vidéo.

L'estimation du mouvement consiste à chercher l'origine des objets se trouvant dans l'image cible (à coder) dans l'image référence et d'extraire des vecteurs de mouvement représentant le déplacement de ces objets, en se basant sur une mesure de ressemblance entre ces objets. On essaie donc à trouver un objet dans la référence minimisant cette mesure (***Figure 25***).

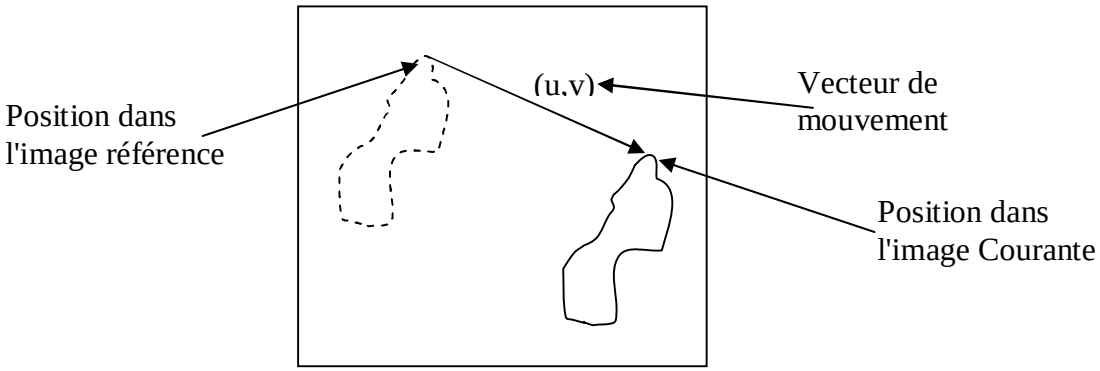


Figure 25:Notion de vecteur de mouvement

L'objet primordial à utiliser est le pixel lui-même. On cherche un vecteur de mouvement pour chaque pixel ce qui est épuisant et ne prend pas en compte les caractéristiques des images composées d'objets et de régions [WENG 93, GALP 02, PACK 97].

La méthode optimale est de reconnaître les objets de la scène après une étape de segmentation ou de reconnaissance d'objets; mais ceci nécessite un temps de calcul important et présente certains problèmes tels que l'ambiguïté, le chevauchement, la déformation… etc.

La méthode la plus efficace est de subdiviser l'image en blocs rectangulaires non chevauchés et baser la recherche sur ces blocs (***Figure 26***). C'est la méthode la plus utilisée dans la compression grâce à sa simplicité pour les implémentations hard et à son efficacité.

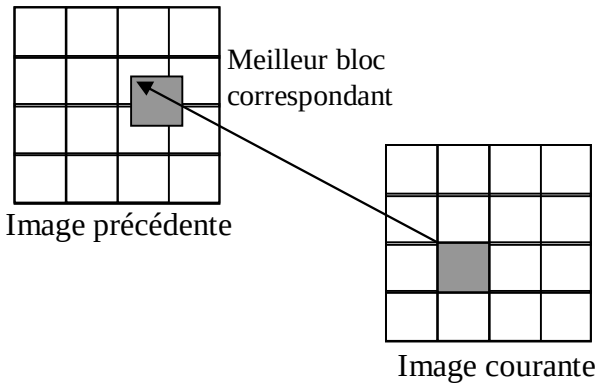


Figure 26:Estimation du mouvement d'un bloc

Il existe d'autres essais qui consistent à transformer les valeurs des pixels en valeurs binaires 0 et 1 puis effectuer les calculs sur ces valeurs. Ces techniques ont prouvé une très grande vitesse, mais une faible précision [GOSS 97, FANG 01].

Dans la littérature, l'estimation de mouvement n'est pas utilisée uniquement pour la compression vidéo, mais elle présente aussi un élément principal de plusieurs applications vidéo tels que l'analyse des séquences, le suivi des objets en mouvement (applications militaires) [LAUR 98], le calibrage des caméras,…etc.

Dans ce chapitre, seront détaillés l'estimation de mouvement, son principe, ses suppositions et algorithmes, et la méthode de Block Matching qui sera utilisée par la suite.

II.2. Contraintes de l'Estimation du mouvement

La recherche des objets similaires dans les images références est basée sur les valeurs de luminance des pixels pour minimiser une mesure de ressemblance. Cette technique demeure valable tant que les pixels des objets gardent les mêmes valeurs de luminance dans le temps, puisque le changement de la luminance produit des déformations dans les objets sans que cela corresponde à un mouvement; d'où les suppositions suivantes :

- Les objets sont des corps rigides c'est à dire que la déformation des objets est négligée au moins pour un minimum d'images voisines pour garantir le même mouvement pour tous les pixels du même objet.
- Le mouvement des objets est translationnel pour un minimum d'images voisines.
- L'intensité est spatialement et temporellement uniforme c'est à dire que l'intensité des objets est constante lors du mouvement.

Avec ces suppositions, le mouvement des objets peut être clairement détecté et utilisé. Les séquences vidéo du monde réel, obéissent généralement à ces contraintes, et leur codage par estimation du mouvement a prouvé sont efficacité.

L'estimation de mouvement consiste donc à résoudre deux problèmes:

- Identifier les objets c'est-à-dire déterminer leurs limites (contours), (c'est la "segmentation du mouvement").
- Estimer les paramètres de leurs mouvements (c'est l'"Estimation du mouvement ").

Dans le cas de considération des objets comme des blocs, le premier problème est négligé et seulement l'estimation du mouvement est considérée.

II.3. Estimation de mouvement et standards

L'utilisation de l'estimation de mouvement dans la compression vidéo remonte au début des années 70 particulièrement par les travaux de Henkel et Limb présentés dans une conférence de codage d'images. Il a été clairement mentionné dans cette conférence [MOSC 01]:

"Dans un système de codage pour la compression vidéo, le déplacement des objets dans deux images successives est estimé et utilisé pour prédire la position de cet objet dans une image suivante. La différence entre cette prédiction et l'image courante est utilisée pour la prédiction au niveau du décodeur. La transmission des informations du déplacement et de la différence uniquement, permet de réduire la capacité requise par le canal de transmission vidéo" (***Figure 27***).

Le codeur doit donc:

- Estimer les paramètres de déplacement des objets.
- Calculer l'erreur par rapport à la référence réelle (décoder la référence).
- Compresser l'erreur.

Et le décodeur doit:

- Décompresser l'erreur
- Restaurer le mouvement.

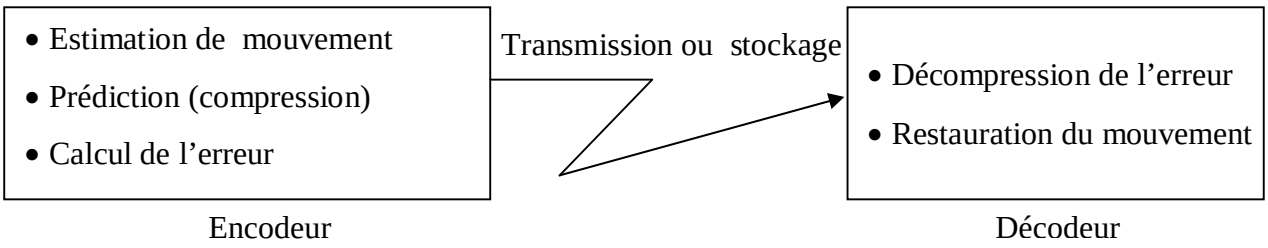


Figure 27:Codage et décodage dans l'estimation du mouvement

La compression et la décompression de l'erreur renferment pratiquement les étapes de transformation par DCT, quantification, et codage en longueur variable.

La plupart des standards vidéo utilisent le même schéma décrit ici pour les deux modes de codage prédictif et bidirectionnel sauf pour la norme H261 qui utilise seulement le mode prédictif. Par exemple, la norme MPEG-1 utilise l'estimation en trois modes : estimation à partir de l'image précédente, suivante ou les deux (***Figure 28***).

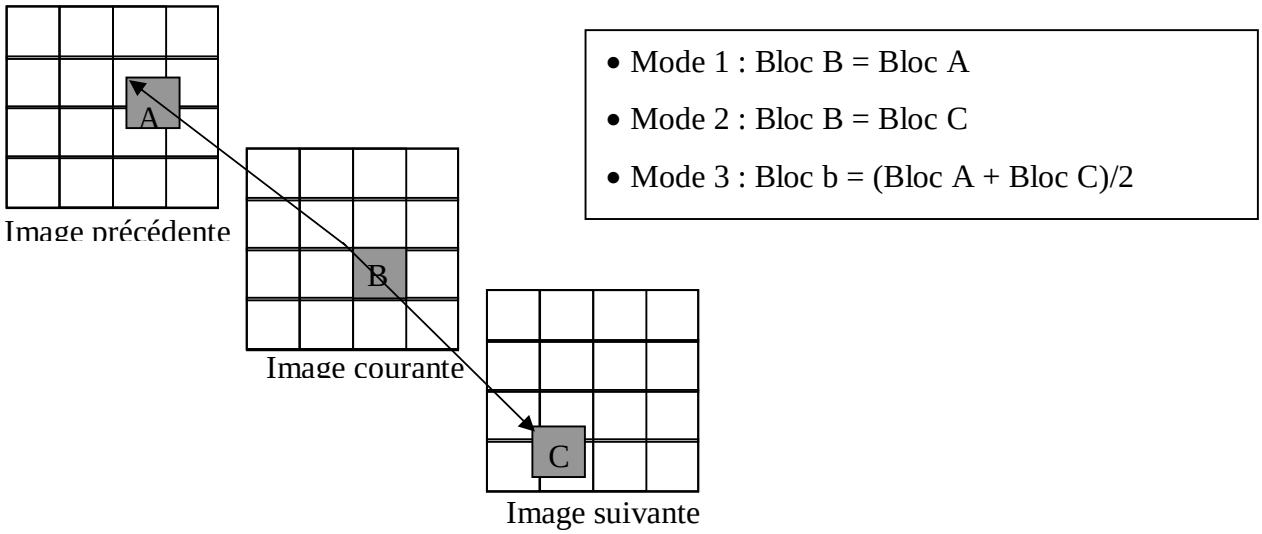


Figure 28:Modes d'estimation du mouvement dans la norme MPEG1

II.4. Méthodes d'Estimation de mouvement

Plusieurs travaux ont été élaborés pour trouver des moyens pour estimer ou calculer le mouvement des objets d'une image à une autre dans une séquence vidéo :

II.4.1. Méthodes basées sur l'équation du flot optique

L'équation du flux optique suppose que le changement des pixels dans les images est dû uniquement au mouvement. Elle comporte deux inconnues : les composantes du vecteur de mouvement.

Soit I(x,y,t) l'intensité du pixel (x,y) à l'instant t.

Trouver les coordonnées du pixel (x, y) à l'instant t + Δt, ou les composantes du vecteur de mouvement de ce pixel, consiste à résoudre l'équation [MOSC 01]:

$$\frac{\partial I(x:t)}{\partial x} v_1(x:t) + \frac{\partial I(x:t)}{\partial y} v_2(x:t) + \frac{\partial I(x:t)}{\partial t} = 0$$

Où $v_1(x:t)$ et $v_2(x:t)$ représentent les composants du vecteur de mouvement du pixel (x,y) à l'instant t.

La résolution de cette équation n'est pas évidente et nécessite un calcul très important pour chaque pixel pour chaque transition ce qui limite son utilisation sauf pour des applications très limitées [BARR 95].

Des techniques visant à minimiser les calculs ont été développées pour extraire des caractéristiques reliant les pixels de la même image (distances, ressemblances de couleurs, mouvements, formes,…) et de ne calculer le flux optique que pour quelques pixels puis extrapoler sur toute l'image en utilisant des généralisations basées sur ces caractéristiques [LIN 95].

II.4.2. Méthodes de Pixel-récursif

Ce sont des méthodes de raffinement récursif par pixel utilisant le calcul des gradients. Elles sont développées pour les applications de traitement des séquences d'images toutes entières. Elles visent à minimiser l'erreur de prédiction DFD (Displaced Frame Difference) [MOSC 01, CHOK 98].

$$DFD(\breve{r}, t, d) = I(\breve{r}, t) - I(\breve{r} - d, t - \Delta t)$$

Où r représente les coordonnées du pixels, t l'instant de la séquence, d le vecteur de déplacement. On cherche donc un vecteur minimisant cette mesure.

On cherche dans le cas idéal : $\left|DFD(\overset{\circ}{r},t,d)\right| = 0$ C'est-à-dire un pixel identique au pixel cherché.

On prend donc pour chaque pixel un vecteur de mouvement d_0 initial et on effectue une récursion jusqu'à l'obtention d'un DFD minimal et ainsi un vecteur de mouvement optimal.

Ces méthodes souffrent de deux effets indésirables :

- L'erreur à minimiser contient des minima locaux ce qui limite la convergence de ces méthodes et augmente leur sensibilité au bruit.
- Les larges mouvements et les discontinuités peuvent déduire en effet des tremblements.

Ces inconvénients, en plus du temps de calcul épuisant, ont limités l'utilisation de ces techniques.

II.4.3. Block Matching

Cette méthode, appelée aussi dans la littérature correspondance de blocs ou encore similarité de blocs, est la plus utilisée et adaptée par la plupart des standards de compression vidéo H26x et MPEG-x, vu sa simplicité pour les implémentations Hardware. C'est une méthode de reconnaissance d'objets où les objets sont de blocs rectangulaires. Le déplacement d'un pixel est déterminé en considérant un bloc de taille NxN contenant ce pixel et chercher le bloc de même taille le plus ressemblant dans une zone appelée *zone de recherche* de l'image de référence. Les autres pixels de ce bloc auront les mêmes caractéristiques du mouvement du bloc tout entier. Le vecteur de déplacement du bloc dans l'image courante par rapport à l'image de référence est appelé *vecteur de mouvement*.

Cette technique permet de réduire le temps de calcul important d'estimation du mouvement si on optimise le nombre de comparaisons effectuées entre les blocs.

II.5. Classement des séquences vidéo

Les séquences vidéo du monde réel sont très variées, en couleurs, vitesse, complexité, … etc., selon le genre de la scène que représente chaque séquence. Dans la compression, il est très utile de connaître des informations préalables sur une séquence vidéo pour effectuer une estimation du mouvement efficace, et obtenir des résultats optimaux. Par exemple, si on peut

connaître d'avance que la séquence à coder ne va pas contenir de larges mouvements, on peut limiter la zone de recherche pour gagner en temps de calcul.

Pour cela, le classement des séquences vidéo peut aider à l'optimisation de la recherche [CHOK 98].

II.5.1. Séquences rapides

Ce sont des séquences où les objets se déplacent d'une image à une autre avec une distance importante qui peut même dépasser les limites de la zone de recherche prédéfinie, ce qui empêche une compression optimale (les séquences des scènes sportives par exemple). Ce type de séquences nécessite donc l'utilisation de grandes zones de recherche pour atteindre les objets en mouvement, au détriment du temps de calcul bien sûr. Des techniques de recherche permettant de minimiser le nombre de blocs testés sont développées pour surmonter le problème du temps de calcul.

II.5.2. Séquences lentes

Ce sont les séquences où les objets se déplacent lentement ou restent fixes (pour la plupart du temps) telles que l'explorations d'une scène par une caméra, les séquences de visiophonie ou les séquences des mobiles de la troisième génération [SHAN 97]. Dans ce cas, une petite zone de recherche suffit pour atteindre un très bon taux de compression en un temps de calcul réduit.

II.5.3. Séquences stationnaires

Ce sont les séquences où les blocs gardent la même position d'une image à l'autre tels que les arrières plans. Dans ce cas, le taux de compression est maximal et le temps de recherche est nul si on sait d'avance qu'une portion d'une séquence est stationnaire.

II.5.4. Séquences simples

Dans ce type de séquences, les images ne contiennent pas beaucoup de détails et les objets connaissent le même mouvement en direction et en vitesse (cas du mouvement de la caméra): c'est-à-dire que les pixels du même objet se déplacent aussi à la même vitesse et la même direction, ce qui permet de trouver des techniques qui convergent rapidement vers le bloc le plus ressemblant.

II.5.5. Séquences complexes

C'est le contraire du dernier type. Les objets connaissent un mouvement aléatoire (ne peut être prévu) c'est-à-dire qu'un objet peut être jugé meilleur localement mais ne l'est pas

globalement (problème des minima locaux) ce qui peut conduire à de faibles résultats en taux de compression et en qualité.

II.6. Conclusion

L'estimation de mouvement utilisée dans plusieurs applications telles que le suivie des objets et le calibrage des caméras, est vue dans ce chapitre comme une solution pour compresser les informations émises ou stockés des séquences vidéo. Plusieurs techniques sont utilisées, mais la plus efficaces est celle de Block Matching. Elle représente actuellement le moteur de base dans tous les standards. Dans le chapitre suivant cette technique sera étudiée.

Troisième chapitre:

Etude et comparaison des algorithmes de block matching

III.1.Introduction

La méthode de block matching, largement utilisée aujourd'hui et développée pour remédier au temps de calcul épuisant de recherche des blocs, a connu le développement de plusieurs algorithmes. Ces algorithmes visent deux objectifs:

- Réduire la charge de calcul des vecteurs de mouvement.
- Augmenter la précision de ces vecteurs.

Pour les applications de codage en temps réel où on cherche généralement à fournir des images codées à une fréquence de 25 Hz tout en gardant une qualité acceptable des images, le premier objectif est essentiel.

Dans ce chapitre, le principe, et les algorithmes de cette méthode sont étudiés. Un encodeur vidéo est également présenté pour la comparaison de ces algorithmes.

III.2.Concepts de base

Le Block Matching est une méthode de corrélation cherchant le bloc le plus ressemblant (dans l'image de référence) au bloc courant dans l'image cible. Il suppose la subdivision de l'image en blocs non chevauchés de tailles identiques qui sont vus comme des blocks indépendants où les pixels composant chaque bloc ont la même allure de mouvement. Le vecteur de mouvement est connu lorsque la position d'origine du bloc de référence est identifiée.

La recherche des blocs similaires dans l'image de référence s'effectue seulement dans une zone appelée zone de recherche de largeur 2d et de centre les coordonnées du block cherché. On cherche dans cette zone le bloc minimisant une mesure de distorsion (BDM: Block Distorsion Mesure), tout en supposant que les pixels du même block ont le même mouvement translationnel et auront ainsi le même vecteur de mouvement (Figure 29).

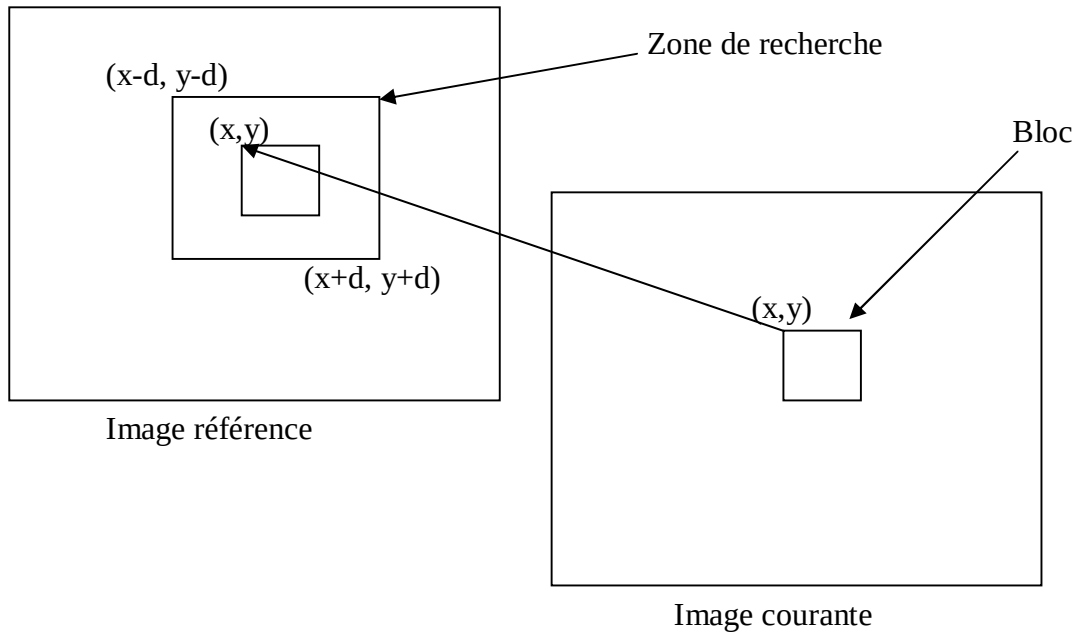


Figure 29:Zone de recherche en Block matching

L'opération de base de block matching est de prendre un bloc candidat et le comparer au bloc cherché. On utilise généralement la différence entre les valeurs de la composante de luminance Y, qui contient l'essentiel des informations, pour évaluer la correspondance. Tous les blocs candidats sont comparés au bloc cherché, celui qui minimise la BDM est pris comme bloc de référence.

Plusieurs paramètres entrent en jeux:

III.2.1. Largeur de la zone de recherche

La largeur de la zone de recherche affecte directement la vitesse et la précision de l'encodeur vidéo. Dans la plupart des standards, c'est un paramètre limité à partir des expériences ou imposé par les contraintes des implémentations hard [MOSC 01]. Cependant, les séquences de mouvement rapide, où les objets changent leurs positions d'une image à l'autre d'une distance considérable, doivent subir une recherche dans une zone couvrant ce déplacement, engendrant bien sûr un temps de calcul plus long. Par contre, les séquences de mouvement lent doivent minimiser la zone de recherche pour profiter du temps de codage.

Le choix de cette distance peut être fait par intervention de l'utilisateur en choisissant directement la taille de la zone de recherche ou tout simplement en spécifiant le type de mouvement (rapide, lent, statique).

Elle peut être aussi choisie et modifiée dynamiquement par l'encodeur selon les informations collectées sur les images précédentes et selon le PSNR obtenu [KESS 96, BOUT 87].

Dans certaines applications, la distance est fixée une fois pour toute pour des contraintes hard (par exemple, la norme MPEG-1 utilise des zones de recherche de taille 15x15, la télévision haute définition (HDTV) utilise une taille de 128x128) [TOUR 00].

Le nombre de blocs comparés peut dépasser le nombre de pixels de la zone de recherche si on utilise une précision de l'ordre d'une fraction de pixel. Des techniques utilisent des blocs de demi, quart ou même huitième de pixel [PACK 97, MEMI 96], elles subdivise l'intervalle entre deux pixels en quatre intervalles délimités par des pixels virtuels approximés (interpolation des deux pixels, et on compare le bloc d'origine avec le bloc débutant par ces pixels), ce qui a permis aussi de donner une précision très importante. Cette technique est adaptée par la norme MPEG et la norme H26 actuelles.

III.2.2. Nombre de blocs candidats

Le nombre de blocs candidats, dans l'image référence, à la comparaison avec le bloc cherché détermine dans une grande partie les résultats de l'estimation du mouvement. Il diffère d'un algorithme à l'autre selon la stratégie de sélection des candidats de la zone de recherche utilisée.

Un nombre important de blocs candidat engendre une faible vitesse de codage et n'implique pas une précision importante, la dispersion des blocs dans la zone de recherche a aussi une importance.

Dans les séquences complexes, un nombre de candidats concentrés dans une région limitée peut écarter de meilleurs blocs. Donc, plus les blocs candidats sont éparpillés dans la zone de recherche, plus la chance d'atteindre le bloc optimal est grande.

III.2.3. Taille des blocks

La taille du bloc choisie affecte clairement la recherche et la comparaison des blocks et ainsi la performance de l'estimation de mouvement. En choisissant une taille importante des blocs (32,64,…), on réduit la précision des vecteurs du mouvement obtenus du fait que les grand blocs contiennent des objets de mouvements différents en vitesse et en direction, mais d'un autre coté on gagne en matière de la taille des informations émises concernant les vecteurs de mouvement puisque le nombre de blocs est réduit.

Le choix d'une taille réduite (2,4,…) permet d'expliciter le mouvement des objets et les limites des objets seront bien identifiées et permet aussi de produire des vecteurs de mouvement très précis. D'un autre coté, les petits blocs souffrent des problèmes de sensibilité au bruit et d'ambiguïté des objets en plus du temps de calcul important.

Les tailles de 8x8 et de 16x16 des blocs sont considérées généralement adéquates pour les applications de compression vidéo. Les standards H26x et MPEG-x utilisent une taille de 16x16.

Dans la littérature [PACK 97, EVAN 99, MEMI 96], on trouve des techniques adaptant une taille variable des blocs et qui ressemblent aux techniques de maillage utilisées dans la synthèse d'image pour profiter des avantages disponibles dans chaque cas de la séquence.

III.2.4. Pixels comparés

Le calcul de la mesure de ressemblance (la comparaison de deux blocs) utilise deux boucles imbriquées pour balayer tout le bloc et calculer la différence des pixels deux à deux, et ceci pour chaque bloc candidat. Plusieurs techniques ont été utilisées pour minimiser le nombre de pixels comparés entre deux blocs [MOSC 01, XAVI 98] et ainsi réduire considérablement le temps de calcul, tout en gardant le maximum de précision pour la fonction de correspondance.

Une technique de comparaison en table du damier a été utilisée (***Figure 30***):

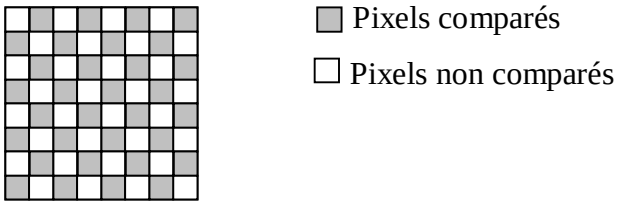


Figure 30:Comparaison en table de damier

Cette technique réduit le temps nécessaire à la moitié, mais elle fonctionne mal en cas des séquences où les objets sont de forme régulière (rectangles, triangles, …).

Une autre technique consiste à sous échantillonner les images (réduire la taille des blocks) et ainsi comparer les blocks (BDM) d'un coût très réduit.

D'autres techniques utilisent des modèles de correspondance carrément (Matching Pattern) où sont déterminés les pixels comparés de ceux non comparés selon le type de la séquence à traiter (***Figure 31***).

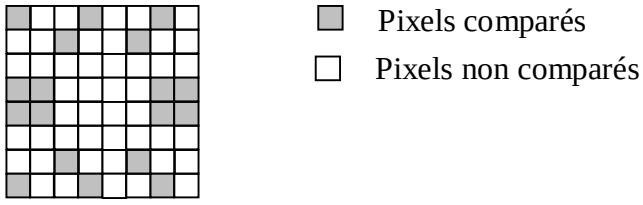


Figure 31: Modèle de correspondance

Ces tables, si elles sont bien choisies, permettent de gagner un gain très important en temps de calcul tout en gardant une précision acceptable.

III.2.5. Fonction de ressemblance

C'est la fonction principale du Block Matching, elle décide à quel point un bloc est similaire à un autre; elle est appelée aussi Mesure de Distorsion de Bloc (MDB). Son choix est nécessaire pour arriver à un bloc optimal. Elle a aussi un impact direct sur la complexité de calcul et sur la précision de l'estimation de mouvement.

Plusieurs fonctions ont été utilisées dans la littérature [PATR 02].

Soit NxN la taille du bloc et I_c l'image cible et I_r l'image de référence, (x, y) les coordonnées du bloc cherché et (dx, dy) le vecteur de mouvement.

1. Moyenne des différences absolues (MDA)

$$MDA = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|I_c(x+i, y+j) - I_r(x+i+dx, y+j+dy)\right|$$

2. Erreur quadratique moyenne (EQM)

$$MEC = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(I_c(x+i, y+j) - I_r(x+i+dx, y+j+dy)\right)^2$$

3. Nombre de différences seuillées (NDS)

$$NDS = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}F(I_c(x+i, y+j), I_r(x+i+dx, y+j+dy))$$

avec,

$$F(\alpha, \beta) = \begin{cases} 1 \rightarrow |\alpha - \beta| \leq t \\ 0 \rightarrow \text{sinon} \end{cases}$$

Dans l'estimation du vecteur de mouvement, on essaye de minimiser la fonction choisie parmi ces fonctions.

Ces fonctions ont prouvées leur performance pour les images réelles. La fonction NDS peut être ajustée pour s'adapter aux caractères de la vision humaine. La fonction MDA semble la plus utilisée en pratique grâce à sa simplicité de calcul et de son adaptabilité pour les implémentations hard.

III.2.6. Seuils d'acceptabilité

Après la recherche des blocs, on aboutit à un bloc (vecteur de mouvement) qui minimise la fonction de ressemblance (MDB) utilisée. La valeur de cette fonction pour ce bloc nous permet de prendre l'une des décisions suivantes [CKOK 98] :

1. Le bloc référence est similaire au bloc cible, ce qui implique l'envoi du vecteur de mouvement uniquement. Ceci est vrai si la valeur de la fonction est inférieure au seuil de ressemblance (***Figure 32***).

$$MDB(B_c, B_r) \leq MinMDB$$

2. Le bloc référence ne ressemble pas au bloc cible, ce qui implique l'envoi du bloc cible tout entier (luminance et chrominance) comme une image intra. Ceci est vrai si la valeur de la fonction dépasse le seuil de non ressemblance.

$$MDB(B_c, B_r) \geq MaxMDB$$

3. Le bloc référence ressemble au bloc cible mais nécessite une légère modification, ce qui implique l'envoi du vecteur de mouvement et la différence de luminance entre les deux blocs. Ceci est vrai si :

$$MinMDB \leq MDB(B_c, B_r) \leq MaxMDB$$

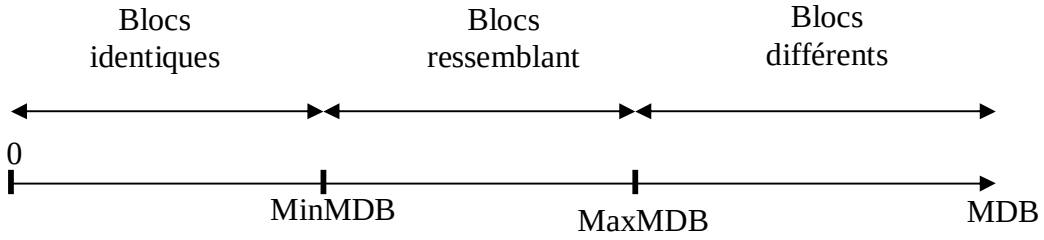


Figure 32: Seuils d'accèptabilité de ressemblace des blocs

III.3. Complexité des algorithmes de Block Matching

L'optimisation des algorithmes de recherche en Block Matching a été largement étudiée au cours des deux dernières décennies [MOSC 01, PACK 97, MART 97] vu son impact fondamental sur l'efficacité de la compression et sur ses différents besoins en calcul et en bande de transmission. La complexité (le coût) d'un algorithme de Block Matching peut être mesurée grâce à la fonction :

$$Coût = N_{test} \times N_b \times Cp + C_{ctrl}$$

Où:

- N_{test} est le nombre de blocs testés ou vérifiés. Il est relatif à la largeur de la zone de recherche et à la stratégie de candidature des blocs utilisés dans cette zone.
- N_b est le nombre de pixels du bloc comparés.
- Cp est le coût de mesure de correspondance entre deux pixels (pixel de référence et pixel cible) en nombre d'opérations de calcul (multiplications, additions, décalages, valeurs absolues,…). Il est relatif à la fonction MDB choisie.
- C_{ctrl} est le coût de la décision en fonction des résultats obtenus de la comparaison.

Plusieurs algorithmes ont été développés visant la réduction des opérations de calculs en sacrifiant un peu de précision des vecteurs de mouvement. La plupart de ces algorithmes se basent sur la réduction de l'un (ou plus) des paramètres déjà cités.

III.4. Algorithmes de Block Matching

Un algorithme de Block Matching commence la recherche d'un point de départ et choisit un ensemble de points candidats autour de ce point selon une distance appelée *"Pas de recherche"*. Le pixel (parmi les candidats) minimisant la mesure de distorsion est pris comme point de départ de l'étape suivante avec une réduction du pas de recherche. Et ainsi itère l'algorithme jusqu'à obtenir un bloc satisfaisant ou atteindre les bords de la zone de recherche.

Dans l'étude de ces algorithmes, on utilise souvent une zone de recherche de dimensions 14x14 pour simuler leur fonctionnement. Le centre de cette zone correspond aux coordonnées du bloc cherché. Dans quelques cas, le nombre de pas effectués pour atteindre les bords de cette zone (14x14) donne le nom de l'algorithme lui-même tel que 3SS, 4SS.

III.4.1. Algorithme de recherche complète (Full Search :FS)

C'est l'algorithme le plus simple, et n'effectue aucune minimisation. On teste tous les blocks de la zone de recherche. On peut distinguer deux techniques différentes (puisque la recherche peut être interrompue en trouvant un bloc similaire):

- Recherche en balayage (***Figure 33***):

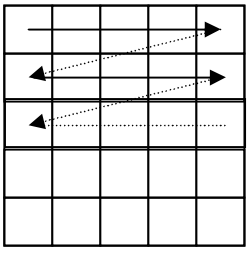


Figure 33: Recherche Complète (Full Search)

- Recherche spirale : elle suppose que plus le bloc est proche de la position du bloc cible, plus la probabilité d'élection est élevée.

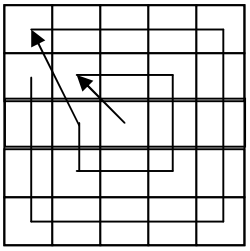


Figure 34: Recherche spirale

Le coût de cet algorithme est de : d^2N^2Cp

Il est évident que cet algorithme est d'une haute précision, mais ses besoins insupportables en temps de calcul défavorisent son adaptation.

III.4.2. Recherche en trois pas (Three Step Search: 3SS)

C'est un algorithme présenté par Koga et all; il commence la recherche du centre de la zone de recherche avec un pas *P* = d/2 (d est la largeur de la zone de recherche). Dans chaque pas, les neuf points qui se trouvent autour du centre d'une distance *P* sont testés (***Figure 35***). Le point minimisant la mesure de distorsion est pris comme centre du pas suivant, avec la réduction de *P* d'un demi. Le processus s'arrête si le pas *P* atteint 1 ou la recherche atteint les bords de la zone de recherche.

Le nombre de poins testés est : $N_{test} = 1 + 8Log_2(d+1)$

Cet algorithme représente un nombre réduit de points testés (logarithme de la largeur de la zone de recherche), mais l'augmentation linéaire du pas de recherche risque de tomber aux minima locaux dans les cas de séquences complexes.

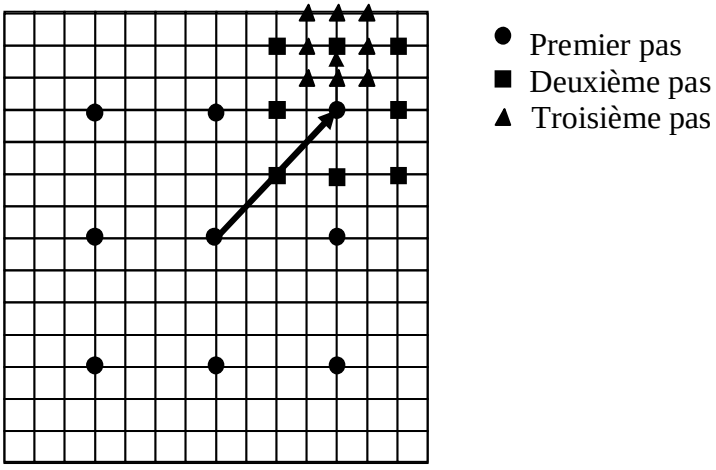


Figure 35:Illustration de la recherche en trois pas

III.4.3. Algorithme de Recherche 2Dlog

C'est un algorithme de recherche logarithmique proposé par Jain et All en 1981. Il commence la recherche par un pas P = d/4. Dans chaque étape, cinq points sont testés : le centre et les points supérieur, inférieur, gauche et droite (***Figure 36***). Si le point minimisant la MDB se trouve au centre ou aux frontières de la ZDR, le pas est divisé en deux pour l'étape suivante, sinon il reste inchangé. Lorsque P atteint 1, les huit points voisins du centre seront testés.

La figure suivante illustre deux cas:

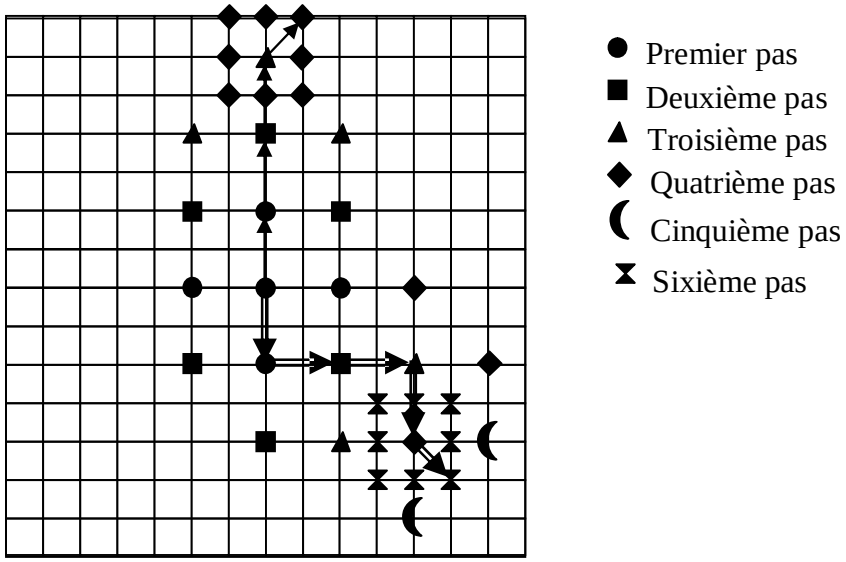


Figure 36: Illustration de la recherche en 2Dlog

Cet algorithme nécessite un temps de calcul plus élevé que le précédent, mais il est plus précis du fait que le nombre de points testés est plus élevé, mais, il est plus sensible aux minima locaux vu le pas de recherche réduit.

III.4.4. Algorithme de recherche orthogonale (Orth. Search: OS)

Proposé par A. Puri en 1987, il commence avec P = d/2. Chaque pas est constitué de deux étapes : une étape horizontale composée de trois points: le centre, les points gauche et droit (Premier pas) (***Figure 37***). Le minimum de ces trois poins est pris comme centre de la deuxième étape verticale avec les deux points haut et bas. Le nouveau point minimum sera considéré comme centre du pas suivant, et le pas de recherche est divisé en 2.

Le nombre de points testés est : $N_{test} = (1 + 4Log_2(d+1)$, ce qui représente un coût raisonnable.

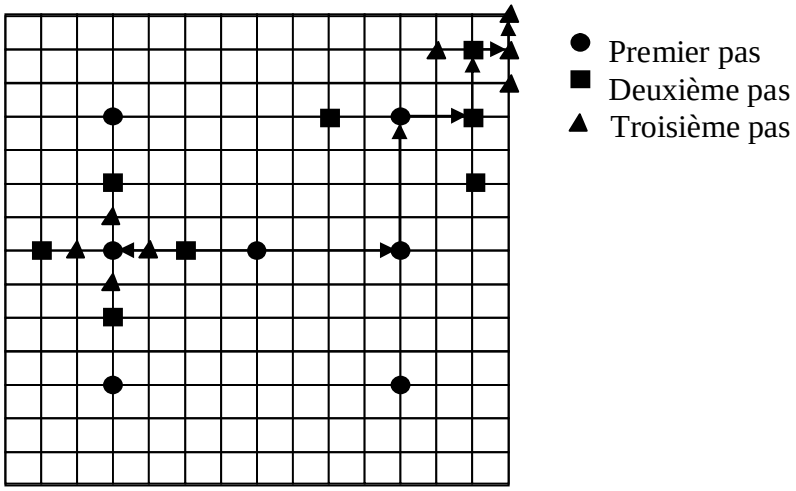


Figure 37: Illustration de recherche ortogonale

III.4.5. Algorithme de recherche Croisée (Cross Search: CS)

C'est un algorithme proposé par Ghanbari en 1990, qui utilise aussi une recherche logarithmique. Il démarre par un pas de recherche égal à d/2. Il prend les quatre points des coins en plus du centre; si celui qui minimise la mesure de distorsion est le centre alors on fait le pas suivant avec une recherche en (+) , sinon on le fait avec une recherche en (X) avec une largeur de pas divisé en deux (***Figure 38***).

Le nombre de points testés est de 5 + Log_2d, il est presque similaire à l'algorithme de recherche orthogonale, mais il utilise un nombre plus réduit de points c'est-à-dire plus rapide.

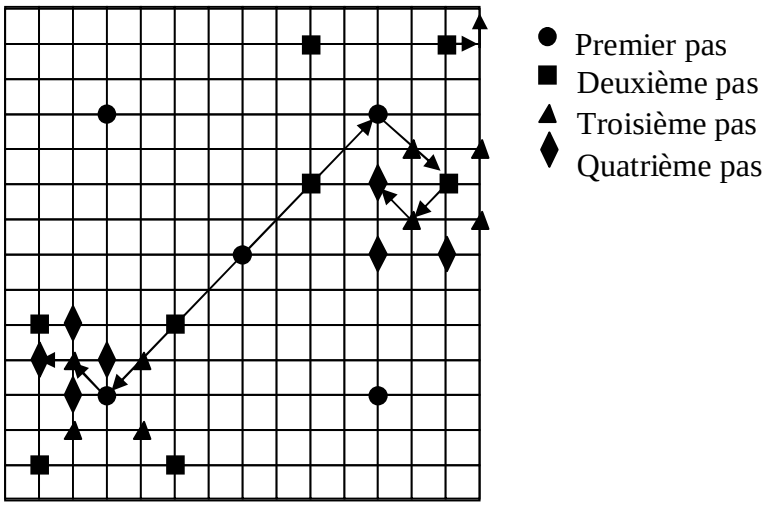


Figure 38: Illustration de la recherche croisée

III.4.6. Recherche descendante de gradient (Gradient Search: GS)

C'est un algorithme qui utilise un pas de recherche fixe (égal à 1 par exemple) et arrête la recherche si le point minimisant la MDB est le point du centre ou la recherche atteint les bords de la zone de recherche (***Figure 39***).

C'est un algorithme qui fonctionne efficacement dans le cas des séquences stationnaires. Le nombre de points testés est de :

$$9 + \alpha * Pas \quad \text{où} \quad \alpha \in R \text{ et } 3 \leq \alpha \leq 5$$

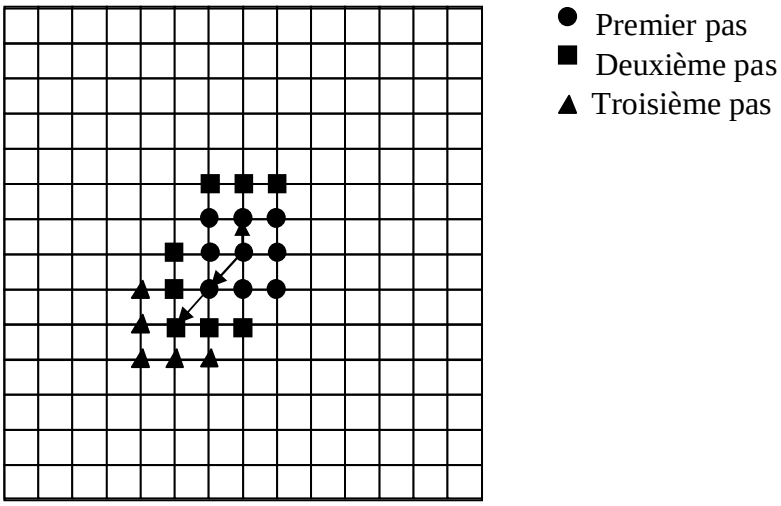


Figure 39:Illustarion de la recherche descendate de gradient

Cet algorithme présente les avantages suivants :

1. Il fonctionne avec des zones de recherche limitées.
2. Un minimum de points testés.

Mais il présente les inconvénients suivants:

1. Il fonctionne mal avec les scènes rapides
2. Il tombe rapidement dans les minima locaux.

III.4.7. Nouvelle recherche en trois pas (New 3SS)

C'est une version modifiée de la recherche en trois pas pour s'adapter aux séquences lentes telle que la vidéo conférence. Il commence par tester neuf points autour du centre; si le minimum reste le centre, il effectue deux autres pas avec un pas de recherche égal à 1. Sinon l'algorithme traditionnel à trois pas est appliqué (

Figure 40).

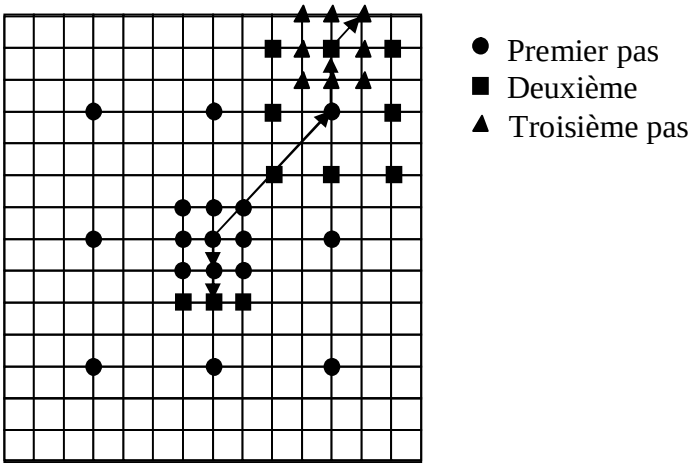


Figure 40:Illustration de la nouvelle recherche en trois pas

Cet algorithme présente les avantages suivants :

1. un nombre réduit de candidats.
2. une très bonne qualité en cas de simples séquences.

Par contre, il risque de tomber dans les minima locaux en cas des blocs non centrés autour du (0,0).

III.4.8. Algorithme de recherche en quatre pas (4SS)

Il commence la recherche avec un pas fixe égal à d/4. Il utilise le même principe que le précédent mais il nécessite quatre pas pour atteindre les bords de la ZDR avec d=1/4. Dans sa première et deuxième étape, si le minimum est au centre, il passe directement à la quatrième étape en divisant le pas en deux. Si on arrive à la quatrième étape et le pas est toujours supérieur à 1, on recommence une nouvelle recherche en quatre pas avec un pas égal au dernier pas obtenu (***Figure 41***).

Le coût de cet algorithme est dans le cas le plus défavorable:

$$N_{test} = 9 + 18Log_2(\frac{d+1}{4})$$

Il représente un nombre réduit de blocs candidats et une convergence rapide. Il soufre des minima locaux dans les séquences complexes.

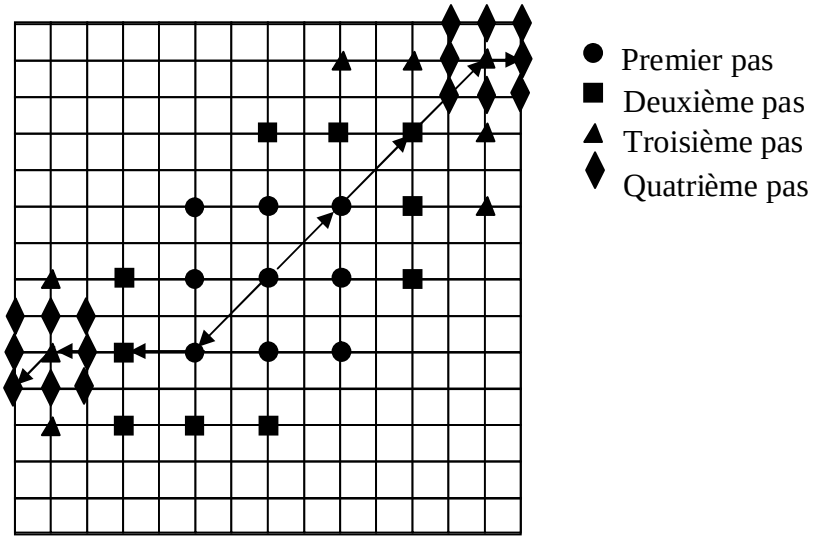


Figure 41:Illustration de la recherche en quatre pas

III.4.9. Block Matching hiérarchique (Hierarchical BM)

Il consiste à effectuer la recherche sur différentes résolutions (multi-résolutions) de l'image c'est-à-dire sur des images sous-échantillonnées de l'image originale. On commence par une simple résolution (1 :16) et en cherche le bloc de taille 4x4 à l'aide de l'un des algorithmes cités précédemment, puis on prend le vecteur de mouvement trouvé comme point de départ du niveau suivant avec une résolution (1 :8) et on effectue une nouvelle recherche. Et ainsi de suite jusqu'à arriver à la résolution complète (***Figure 42***).

Cet algorithme nécessite un taux de calcul très réduit du coté des tailles des blocs, et il est très performant en cas des séquences rapides puisqu'il couvre un espace de recherche important pour chaque bloc à un coût réduit. En contre partie, il nécessite la présence des images sous échantillonnées (mémoire et calcul), ainsi qu'une résistance faible aux minimas locaux.

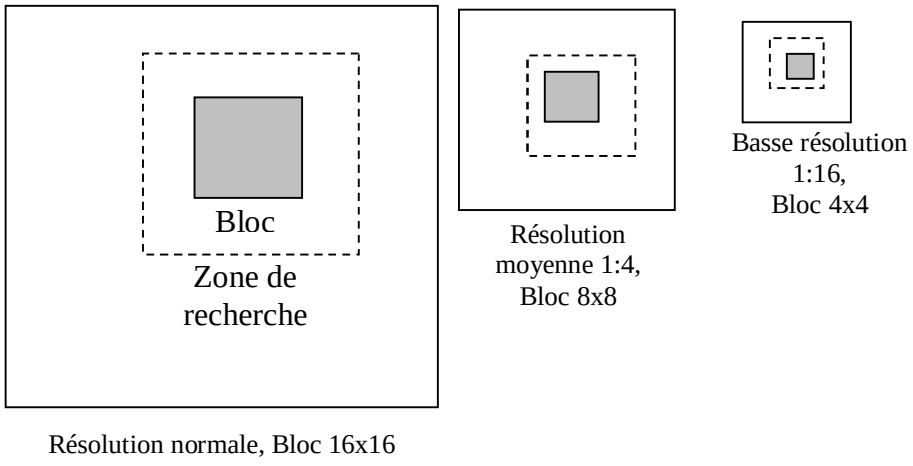


Figure 42:Illustration du Block Matching hiérarchique

III.4.10. Block Matching à fraction de pixel (BMFP)

Dans les séquences vidéo réelles, le mouvement des blocs n'est pas lié à la nature entière du nombre de pixels relatif au matériel d'acquisition utilisé, mais plutôt à des fractions de pixels: 1/2 pixel, 1/4 pixel, 1/8 pixel, voire moins. On utilise l'idée du block matching hiérarchique mais à une plus haute précision.

Un codage précis pour les séquences vidéo réelles cherche les blocs même dans les intervalles entre chaque deux pixels et trouve des vecteurs de mouvement en fraction de pixel très précis.

Pour pouvoir atteindre cette précision, on doit augmenter la résolution des images de référence deux, quatre, ou huit fois, voire plus. Par exemple, dans le cas d'une précision en demi pixels, on multiplie la taille de l'image par quatre (2x2) et on aura au lieu des blocs 16x16 des blocs de 32x32. On doit donc chercher notre bloc dans ce nouveau grand espace et trouver des vecteurs de mouvement en fraction de pixel tel que (3,-2.5), (5.5 , 0.5).

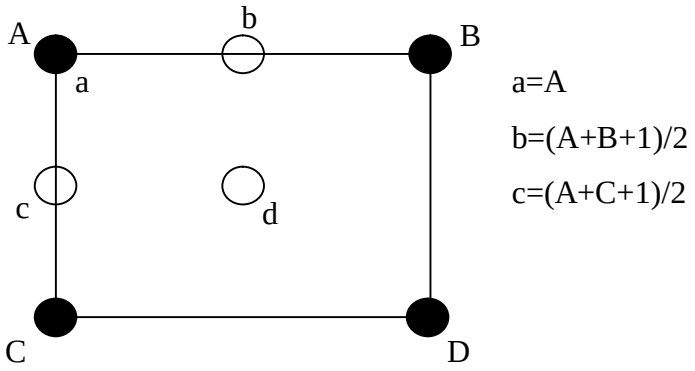


Figure 43:Estimation de mouvement à demi pixel

La recherche peut utiliser n'importe quel algorithme parmi ceux déjà décrits. Plusieurs standards utilisent cette technique pour augmenter la précision des séquences fournies.

Il est clair qu'une telle technique nécessite un temps de calcul important que ce soit pour le nombre de blocs testés ou pour le nombre de pixels testés entre deux blocs. Mais avec des moyens de calcul adéquats, cette technique peut donner les meilleures précisions.

Des techniques d'optimisation dans cette méthode évitent l'agrandissement des images références et limitent le raffinement de la précision au bloc choisi par l'un des algorithmes précédents.

III.5.Comparaison des méthodes de Block Matching

Les algorithmes de block Matching présentés ci-dessus ne sont pas les seuls; il en existe d'autres, mais qui sont destinés à des applications précises telles que les scènes 3D [GALP 02] ou le suivi des objets [LAUR 98], ou pour des accélérations maximales au détriment de la précision [FANG 01].

Les algorithmes, les plus utilisés dans le codage des séquences du monde réel présentés ici, représentent des solutions proposées pour différents domaines de codage telles que la visiophonie, les téléconférences, la télévision, …etc. Chaque domaine est caractérisé par ses propres problèmes liés aux caractéristiques de ces séquences telles que la taille, la vitesse, les couleurs, la fréquence, la précision, …etc.

L'amélioration de tels algorithmes passe par leur comparaison pour tirer leurs propriétés, avantages, inconvénients et les types de séquences pour lesquelles ils donnent les meilleurs résultats.

Du point de vue théorique, on remarque que chacun des algorithmes précédents, mis à part le Full Search, est conçu pour un type précis de séquences, puisque la valeur du pas de recherche représente en quelque sorte la vitesse attendue (estimée) par l'algorithme.

Après croire cerner le bloc ressemblant au bloc cherché dans une zone précise, l'algorithme de block matching commence à raffiner en considérant des pas plus courts.

L'algorithme GS suppose que le bloc cherché est très proche de l'origine, c'est-à-dire que son mouvement est lent est par conséquent son pas de recherche est très réduit (égal à 1). Cette considération limite l'efficacité de cet algorithme pour les séquences lentes (vidéo conférence, téléphones mobiles, vidéo chat).

L'algorithme 3SS s'adapte avec les séquences à large mouvement avec son important pas de recherche initial égal à d/2. Cependant, du coté de précision, cette valeur devient une punition, puisque cerner très tôt la zone de recherche risque de rater de bons blocs et perdre en conséquent en taux de compression.

Le N3SS vient combler cet inconvénient tout en essayant de prendre en considération les séquences lentes. Il applique la même technique du GS si le meilleur bloc se trouve au centre de la zone de test.

La présences des blocs non concentrés autour de l'origine (0,0) risque de trembler le fonctionnement du N3SS du fait que ce cas présente des minima locaux, ce qui a poussé à l'amélioration apportée par le 4SS.

L'algorithme 4SS utilise le même principe de concentration à la fin de la recherche mais en dispersant plus les candidats pour mieux éviter les minima locaux.

L'algorithme de recherche hiérarchique permet de réduire considérablement le temps de recherche en réduisant la taille des blocs et des zones de recherche pour cerner le bloc cherché en sous échantillonnant les images. La recherche dans ces zones réduites utilise les algorithmes précédents (GS, 3SS,…), c'est-à-dire que l'amélioration de ces algorithme améliore le fonctionnement de l'algorithme de recherche hiérarchique. Cet algorithme représente une technique très intéressante en cas des séquences rapides mais, il faut remarquer comme même que le temps additionnel de sous échantillonnage représente une charge inutile en cas des séquences lentes et stationnaires.

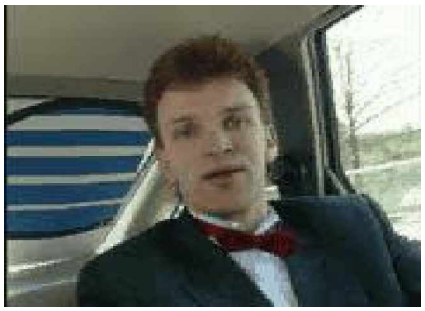
Les caractéristiques des algorithmes étudiés ont été confirmées en codant des séquences standard sur une application de codage vidéo réalisée selon les principes étudiés dans les sections précédentes tels que le JPEG, le codage prédictif, le codage bidirectionnel,…etc.

Cette application représente une plate-forme sur laquelle on peut exécuter ces algorithmes c'est-à-dire un encodeur/décodeur (Codec) vidéo capable de donner des mesures de performance pour chaque algorithme pour différentes scènes. Pour cet objectif, un encodeur est réalisé en se basant sur les principes Intra et Inter déjà présentés.

Les séquences vidéo utilisées comme des références de test contiennent la plupart des caractéristiques des séquences du monde réel. Ce sont les séquences utilisées pour les tests par la plupart des standards, et des recherches en codage vidéo.

III.5.1. Séquences de test

Des séquences vidéo (Figure 44) représentant les types de séquences connues (rapides, lentes, statiques, complexes et simples, couleurs et en noir et blanc) ont été utilisées pour évaluer les performances de l'encodeur utilisé et des algorithmes de block matching étudiés. Ces séquences sont utilisées par la plupart des normes comme des références de test [HANZ 94, H26x].

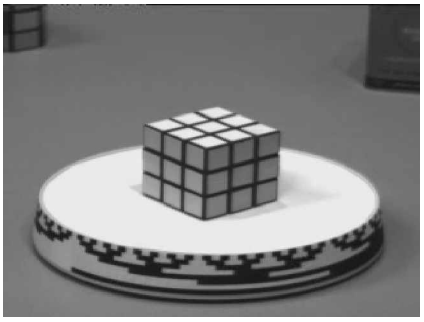

Carphone


Miss America

News


Forman


Trees


Rubic


Ski


Lab

Figure 44:Premières images des séquences de test

Le tableau suivant (***Table 3***) illustre les caractéristiques de chacune de ces séquences:

Séquence	Dimensions	Nombre d'images	Couleurs	Taille brute (Octets)	Description
1. Carphone	144 x 176	100	RGB 24 Bits	7603200	séquence rapide, simple, plusieurs objets en mouvement
2. Miss America	240 x 352	150	RGB 24 Bits	38016000	Séquence, lente, simple, un seul objet en mouvement
3. News	144 x 176	100	RGB 24 Bits	7603200	Blocs rapides + blocs lents, plusieurs objets en mouvement
4. Forman	144 x 176	100	RGB 24 Bits	7603200	Séquence rapide, simple
5. Trees	256 x 232	21	RGB 24 Bits (N & B)	3741696	Séquence Lente, déplacement de la caméra
6. Rubic	240 x 256	22	RGB 24 Bits (N & B)	4055040	Séquence simple, un objet en rotation rapide
7. Ski	180 x 240	41	RGB 24 Bits	5313600	Séquence rapide, et mouvement de caméra
8. Lab	144 x 176	100	RGB 24 Bits (N & B)	7603200	Séquence lente, simple

Table 3: Séquences utilisées pour les tests

III.5.2. Structure de l'Encodeur

L'encodeur réalisé pour l'évaluation utilise une compression en trois modes (Intra, Prédictif et bidirectionnel) organisés en quatre modules:

III.5.2.1. Module de synchronisation

Il prend en charge le codage des paramètres nécessaires au décodage tels que la fréquence des images, les tables de quantification, les types d'images,… etc.

Il réalise aussi la conversion des images en entrée du système RGB au système YUV, ensuite il décide le type de codage utilisé I, P ou B selon la série modèle (IBBP) définie par l'utilisateur, en déclenchant le module approprié et lui fournissant les données nécessaires, ou en mettant l'image dans la file d'attente en attendant l'arrivée de ses références dans le cas des images B:

- Dans le cas du codage fixe, il lui fournit les matrice de quantification de luminance et chrominance.
- Dans le cas du codage prédictif ou bidirectionnel, il fournit la table de quantification de luminance, la méthode du codage utilisée, le modèle de comparaison des blocs, la mesure de correspondance et les seuils d'acceptabilité.

III.5.2.2. Module Intra

C'est le module chargé du codage fixe. Par défaut, Il est appelé à chaque seize images, sauf en cas d'intervention de l'utilisateur. Il utilise le modèle décrit par la norme JPEG avec de codage de Huffman (*Figure 14*).

Après le codage, l'image est décodée et stockée pour être utilisée dans des codages ultérieurs P ou B.

III.5.2.3. Module Prédictif

Il permet de coder l'image donnée avec une estimation de mouvement en block matching à partir de la dernière image P ou I codée. Après le codage, une compensation de mouvement est effectuée pour restaurer l'image pour l'utiliser dans les codages P et B suivants.

III.5.2.4. Module bidirectionnel

Une fois déclenché, il prend les images stockées dans la file d'attente une par une et les code par estimation de mouvement à partir de l'image I ou P stockée et l'image I ou P qui vient d'être codée.

III.5.2.5. Schéma de l'Encodeur

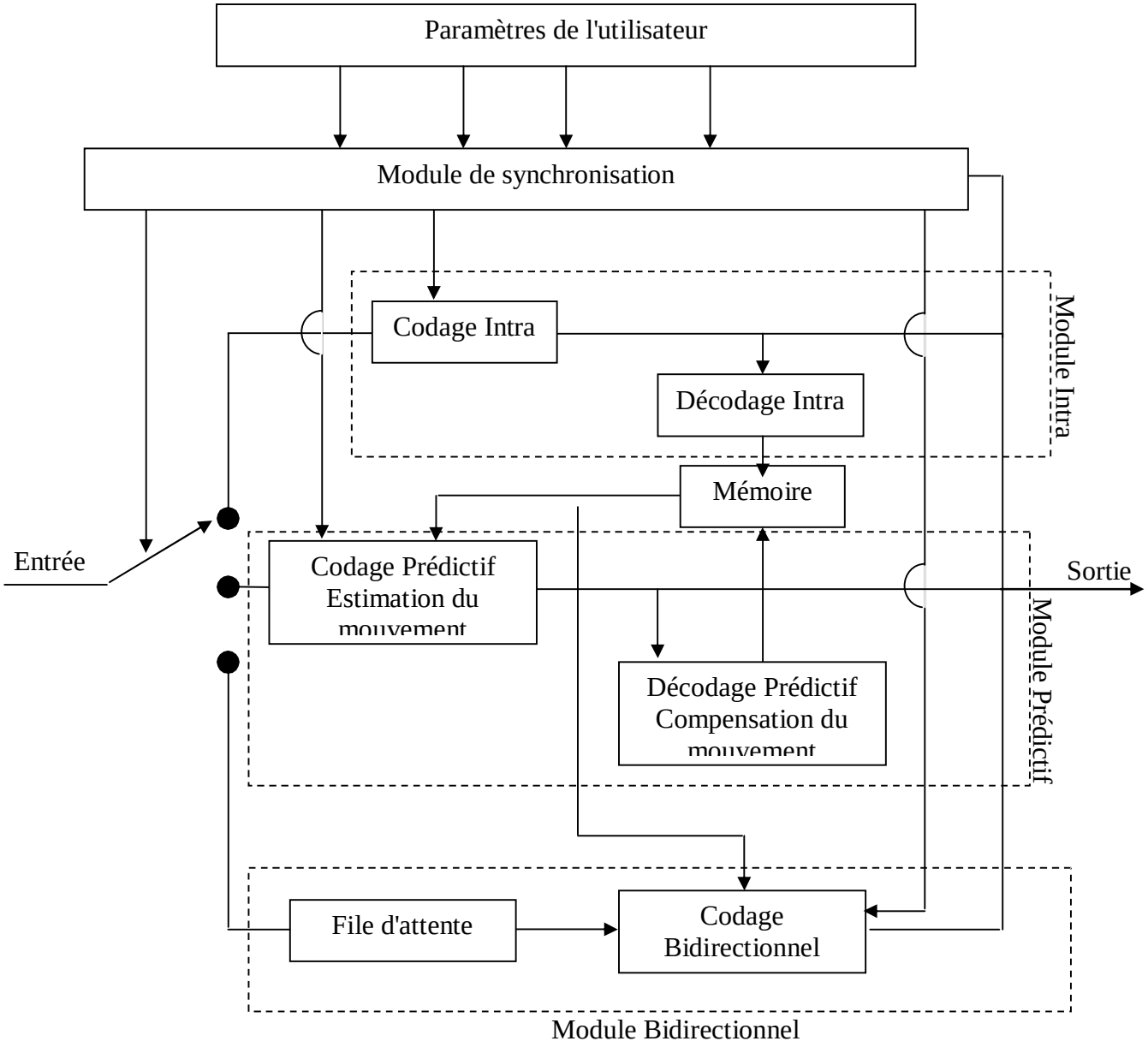
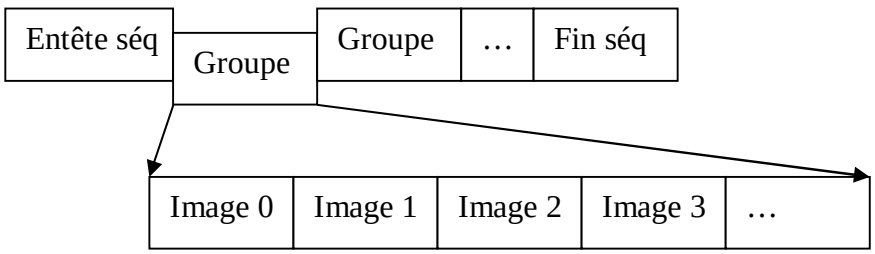


Figure 45:Schéma de l'Encodeur utilisé

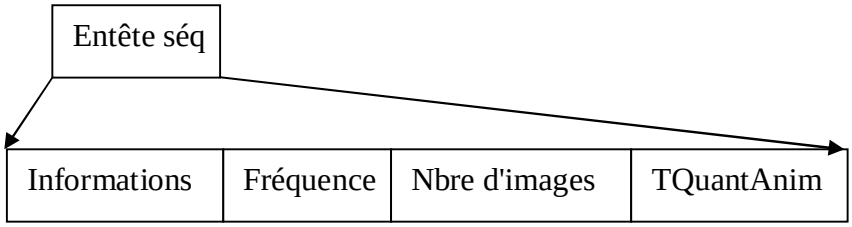
III.5.3. Format des données

Les données codées produites par l'encodeur sont organisées sou la forme suivante:



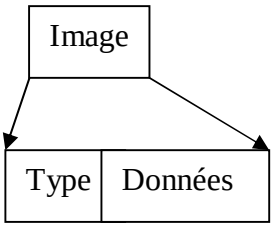
La séquence est composée d'une entête est d'un ensemble de groupes d'images contenant chacun une image I et un ensemble d'images P et B, par exemple IBBPBBPBBP.

- **Entête de la séquence**



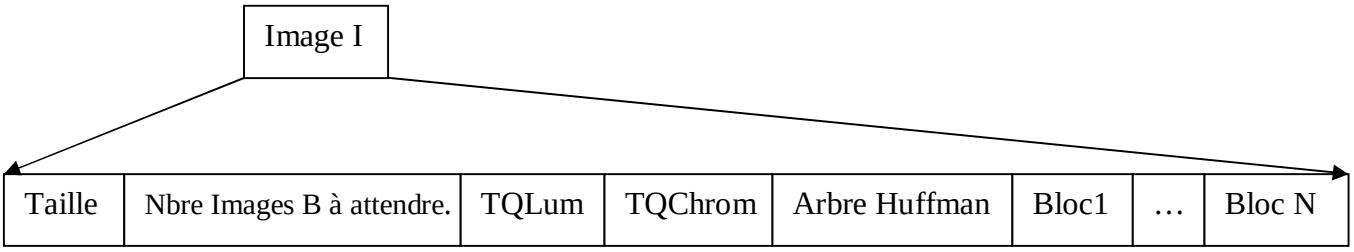
L'entête de la séquence contient d'une part des éventuelles informations générales sur la séquence tel que le titre, la description, la date, le propriétaire, …etc., et la fréquence d'affichage des images pour assurer une même cadence que l'acquisition. D'autre part, le nombre d'images qui est une valeur optionnelle qui représente le nombre total des images dans la séquence utilisée pour estimer sa durée. TQuantAnim contient la table de quantification de luminance utilisée pour quantifier la différence entre les blocs.

- **Image**



Chaque image commence par un indice indiquant son type I, P ou B, puis les données correspondant à ce type:

III.5.3.1. Données des images de Type I

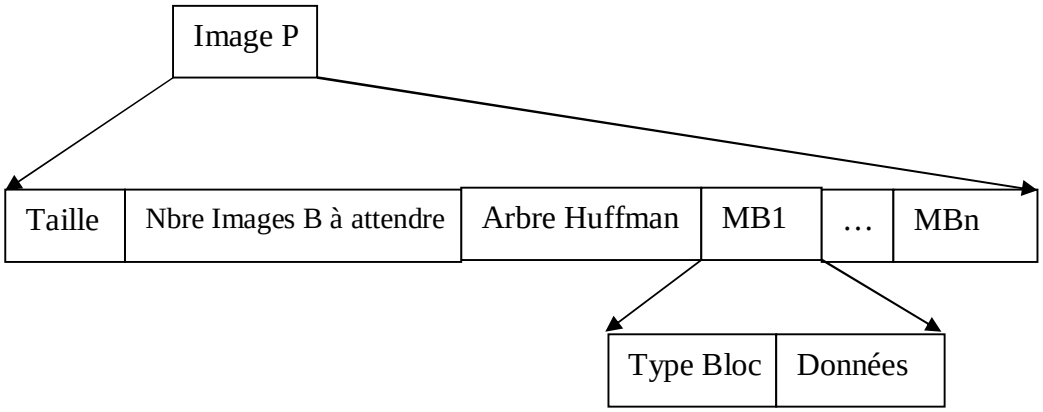


Les données d'une image de type I contiennent sa taille hauteur et largeur, puis le nombre d'images B que doit attendre cette image avant son affichage, puis les tables de quantification de luminance et de chrominance puis l'arbre de Huffman du code des blocs Y, U et V de l'image et enfin les données de ces blocs en code de Huffman.

Les tables de quantifications sont envoyées avec chaque image puisqu'elles peuvent être choisies selon le cas de chaque image et elles ne contiennent que 128 octets.

Chaque Image est codée en codage de Huffman indépendamment des autres pour assurer une résistance aux erreurs en cas de perte ou d'erreurs sur les données d'une image.

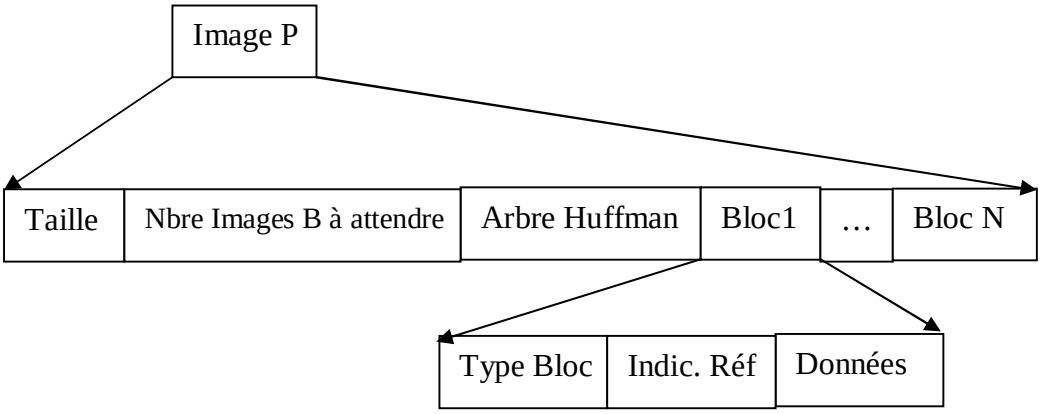
III.5.3.2. Données des Images de Type P



Les données d'une image P contiennent sa taille et le nombre d'images B (codées à base de cette image) qui vont être reçues après, puis l'arbre du code Huffman des macro blocs (MB). Chaque macro-bloc commence par un drapeau indiquant son type: MB complet, MB avec différence ou MB identique:

- Un MB complet contient les données de quatre blocs 8x8 de luminance Y codées en DCT et quantifiées, et deux blocs 8x8 de chrominance U et V échantillonnés et codés en DCT et quantifiés aussi.
- Un MB avec différence contient un vecteur de mouvement par rapport au bloc référence et la différence des quatre blocs Y de ce macro-bloc avec l'image de référence.
- Un MB identique contient seulement le vecteur de mouvement par rapport au macro bloc de l'image référence.

III.5.3.3. Données des images de type B



Une image B contient les mêmes données q'une image P sauf pour les blocs on trouve un indicateur (Indic. Réf) précisant la référence précédente ou suivante.

III.5.4. Schéma du décodeur

Le décodeur reçoit le flux de données codées et décode les images selon leur type. Pour les images I, il utilise le processus inverse du processus JPEG. Pour les images P, il fait la compensation du mouvement à partir de la dernière image I ou P codée.

Pour les images B, il reçoit les images futures et les décode mais ne les affiche qu'après la compensation du mouvement des images B reçues et leur affichage (***Figure 46***).

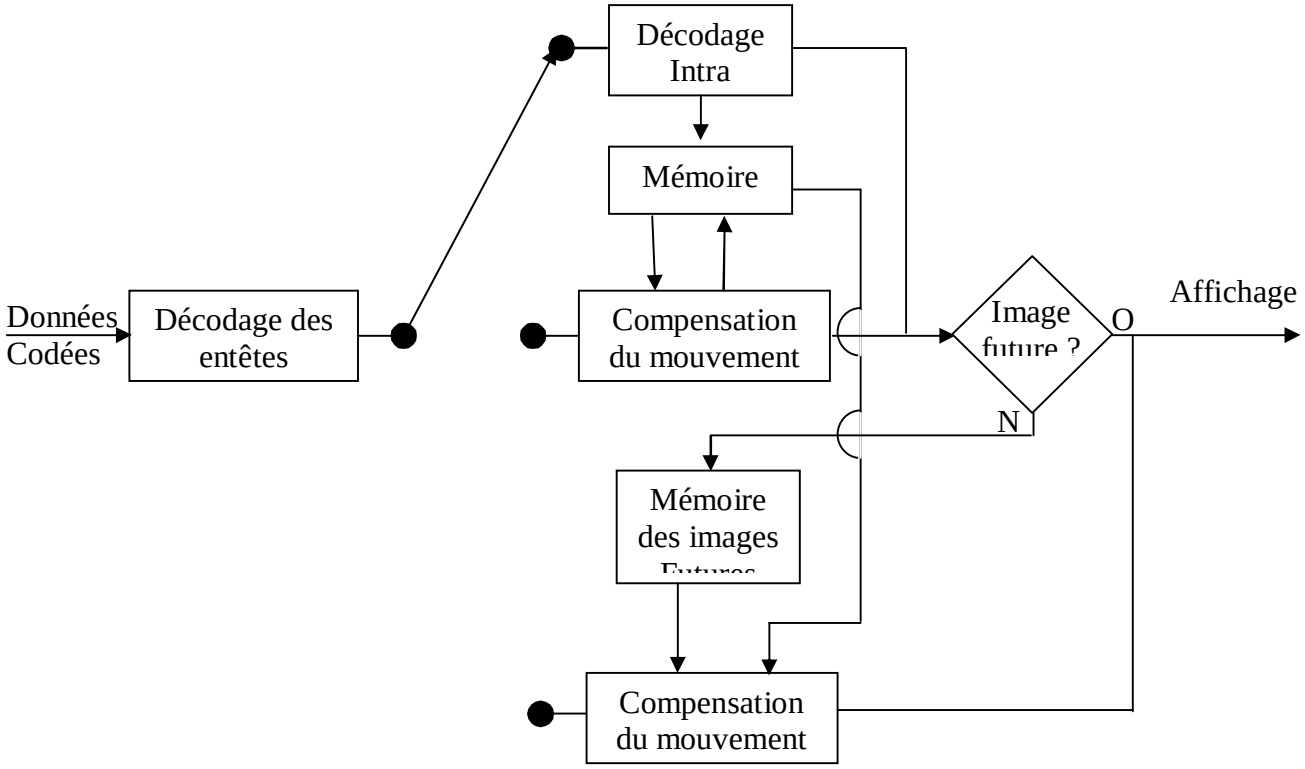


Figure 46:Schéma du décodeur

III.5.5. Résultats

Pour la validation du Codec réalisé selon la description précédente, ses résultats de codage ont été comparés à ceux des encodeurs MPEG2 et H263 utilisés pour coder la séquence Miss América. Le tableau suivant (***Table 4***) illustre les résultats obtenus:

	Codec Réalisé		**Codec H263**		**Codec MPEG2**	
	TC (%)	PSNR (dB)	TC (%)	PSNR (dB)	TC (%)	PSNR (dB)
	99.91	34.38	99.24	38.51	99.50	37.58
	99.50	35.01	98.14	41.75	99.37	38.77
	98.09	39.03	96.29	43.98	99.16	40.27
	96.64	42.08	83.33	48.38	98.95	42.18

Table 4: Résultats de l'encodeur comparés à MPEG2 et H263

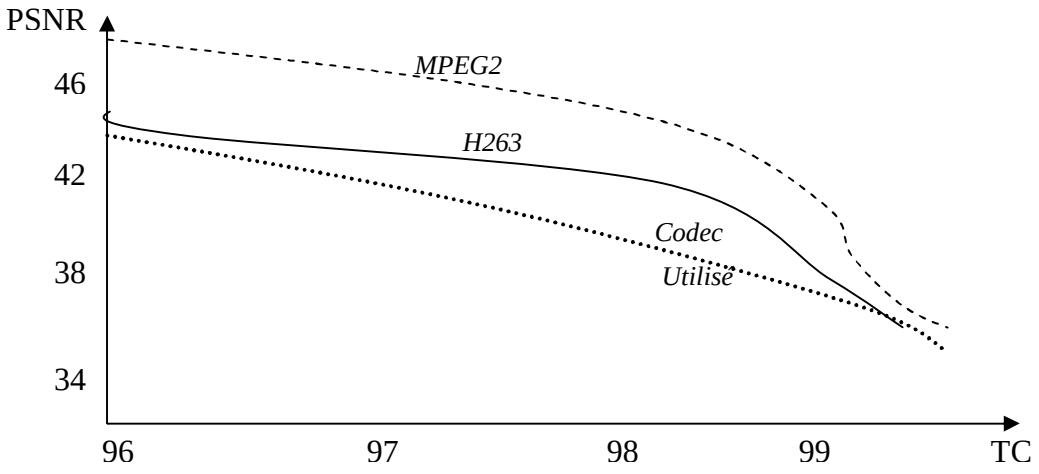


Figure 47:Illustration du PSNR en fonction du TC

Ces résultats sont obtenus en utilisant la configuration suivante:

- La compression se fait en IBBP avec une image I toutes les 16 images.
- Le modèle de comparaison de blocs est en chess-board.
- La mesure de comparaison des blocs est la MAE (moyenne des valeurs absolues).
- La méthode de recherche est 3SS.
- La zone de recherche est d'une largeur de 32 pixels.

Les tables de quantification et les seuils d'acceptabilité sont les meilleurs possibles. Le seuil MinBDM représente le degré maximal de ressemblance entre deux macroblocs c'est-à-dire le seuil séparant la compression d'une différence de blocs et la compression d'une simple référence. L'augmentation de ce paramètre (jusqu'à une certaine limite selon le cas de la séquence) permet de gagner en taux de compression et en temps de calcul, mais perdre en qualité (PSNR) (Figure 48). Le paramètre MaxBDM est moins important, il représente le seuil entre la compression d'une

différence de block ou un bloc tout entier, son aiguillage a le même effet que le premier, mais pas avec la même importance.

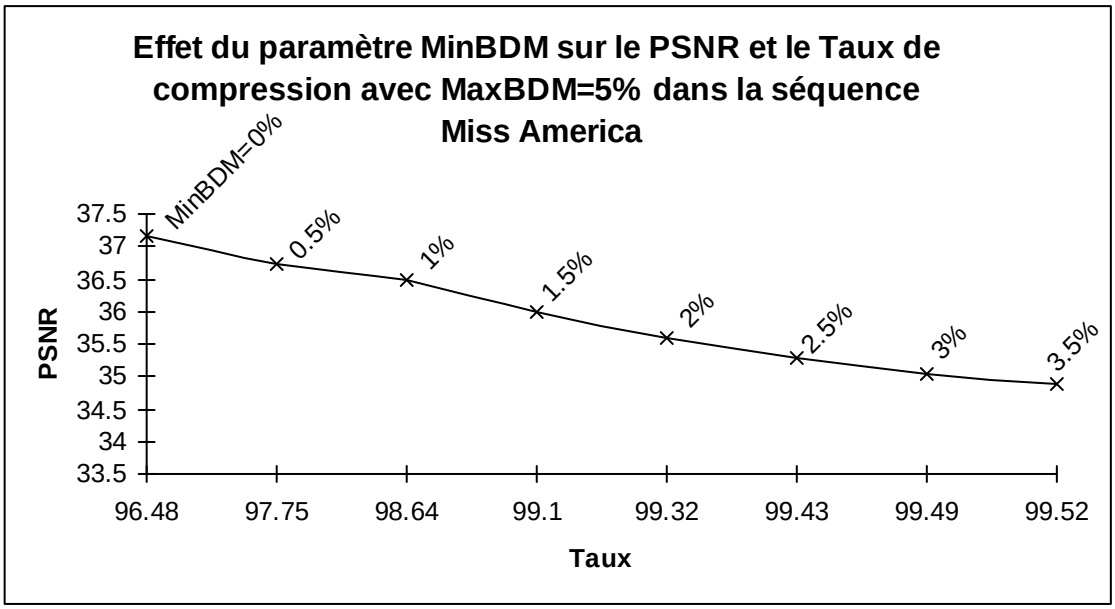


Figure 48:Influence du paramètre MinMDB sur le TC/PSNR

Les deux paramètres MinMDB et MaxBDM fonctionnent avec les deux types d'images P et B et ils peuvent être calculé automatiquement selon les objectifs du codage et l'importance de la qualité ou du temps de calcul ou du taux de compression.

Les résultats trouvés par l'encodeur sont proches de celles des normes MPEG2 et H263, ce qui a permet de comparer les algorithmes de blocs matching sur cette plateforme.

III.5.6. Comparaison

Les algorithmes de block matching vus précédemment, sont comparés afin de tirer leurs avantages et de trouver et conclure pour chaque type de séquences l'algorithme qui convient le mieux. La comparaison est effectuée pour le taux de compression, PSNR et la durée de calcul en terme du nombre de blocs comparés en considérant le même modèle de comparaison des blocs: le chess-board. Les séquences présentées précédemment sont toutes utilisées pour la comparaison. Les résultats suivants (Table 5) sont obtenus avec les meilleures valeurs possibles pour les paramètres MinBDM et MaxBDM ainsi que pour les tables de quantification, et avec une zone de recherche de 32x32 pixels :

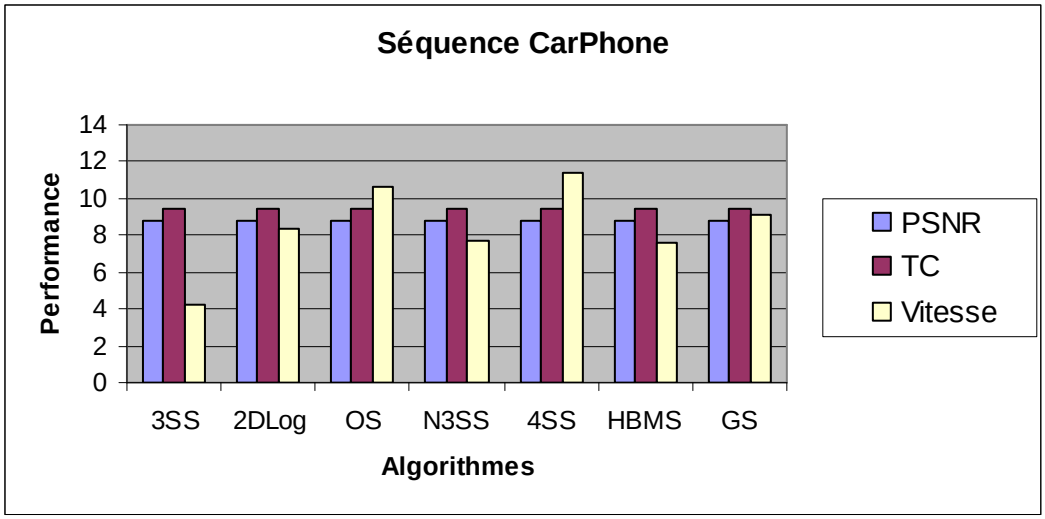
Séquence \ Algorithme		FS	3SS	2DLog	OS	N3SS	4SS	HBMS	GS
Carphone	PSNR	35.29	35.19	35.17	34.96	35.14	35.12	35.14	35.27
	TC	94.05	93.99	93.99	93.94	93.91	93.90	94.02	94.03
	Blocs Testés	13595825	356794	178780	140490	195252	131040	196421	164685
Miss America	PSNR	34.37	33.78	33.90	32.75	33.15	33.07	33.90	34.22
	TC	99.70	99.64	99.66	99.68	99.62	99.63	99.66	99.61
	Blocs Testés	73904060	1681772	783599	252071	910473	453448	787811	632960
News	PSNR	34.67	34.45	34.41	34.33	34.65	34.47	34.41	34.61
	TC	96.52	96.49	96.50	96.48	96.45	96.46	96.50	96.50
	Blocs Testés	13595825	329225	147269	91280	162008	155059	157152	152080
Forman	PSNR	34.61	34.58	34.61	34.58	34.59	34.65	34.60	34.63
	TC	94.01	93.96	93.96	93.94	93.94	93.88	93.97	93.96
	Blocs Testés	13595825	400673	252986	201921	301661	193239	299796	308613
Trees	PSNR	34.06	34.05	34.06	34.05	34.04	34.06	34.06	34.06
	TC	94.20	94.17	94.17	94.18	94.14	94.14	94.19	94.12
	Blocs Testés	6889440	208079	124294	109564	135300	91865	145864	96466
Rubic	PSNR	37.34	37.27	37.29	37.18	37.18	37.21	37.28	37.33
	TC	97.33	97.32	97.32	97.33	97.32	97.31	97.32	97.32
	Blocs Testés	6889440	174598	82762	54650	89406	55660	87742	78097
Ski	PSNR	30.34	30.29	30.39	30.33	30.28	30.34	30.37	30.38
	TC	93.27	93.20	93.18	93.20	93.20	93.15	93.19	93.15
	Blocs Testés	10111920	300427	208442	150410	265251	153477	249765	267736
Lab	PSNR	34.96	34.96	34.96	34.96	34.97	34.96	34.96	34.96
	TC	94.50	94.49	94.49	94.49	94.49	94.49	94.49	94.49
	Blocs Testés	13595825	395563	208781	199460	217634	163432	225518	141343

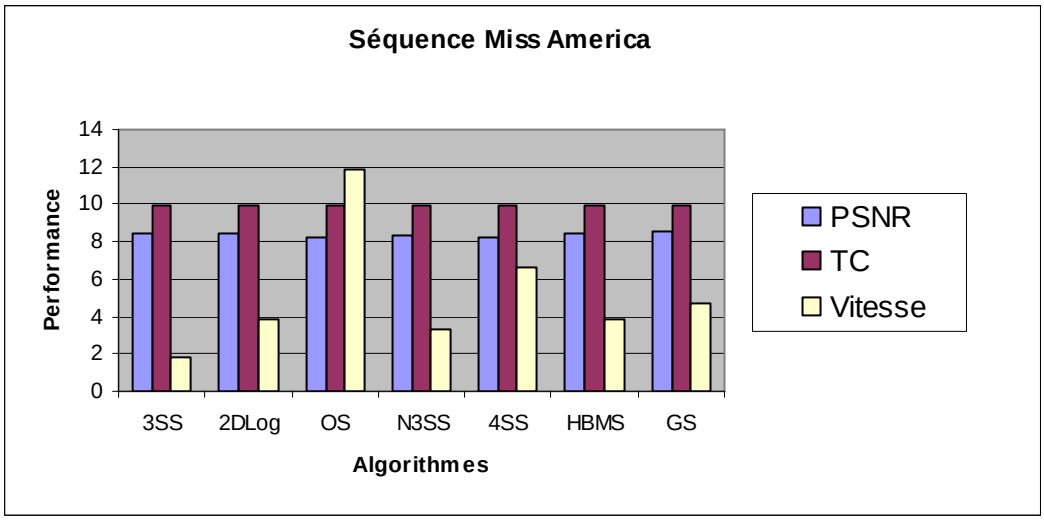
Table 5: Comparaison des algorithmes de block matching

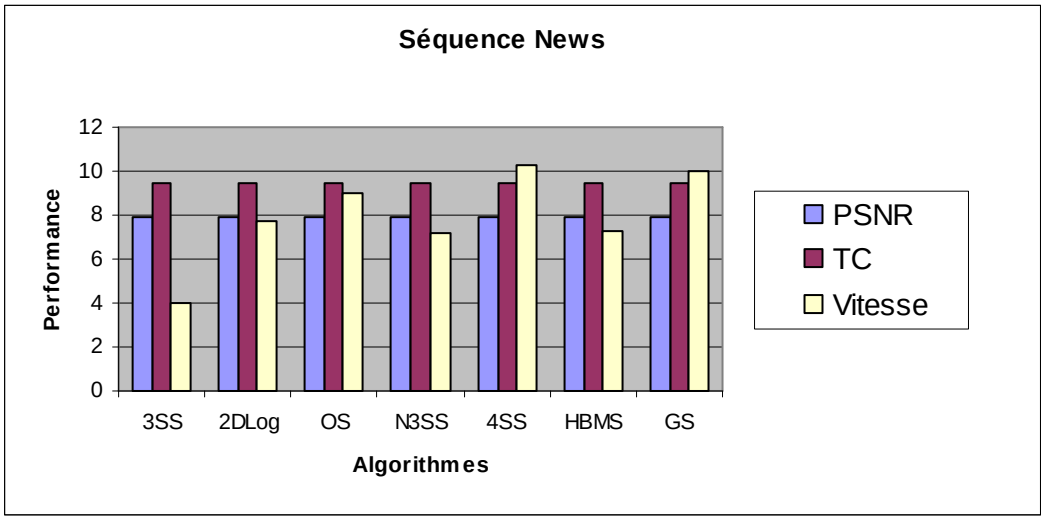
Les résultats présentés dans le tableau 5 montrent que l'algorithme Full search est très gourmant en temps de calcul, il a besoin de plus de 30 fois de blocs à tester que l'algorithme 3SS, mais il arrive toujours aux meilleurs résultats en taux de compression et en qualité des images. Cet inconvénient de temps de calcul limite l'utilisation de cet algorithme aux cas des zones de

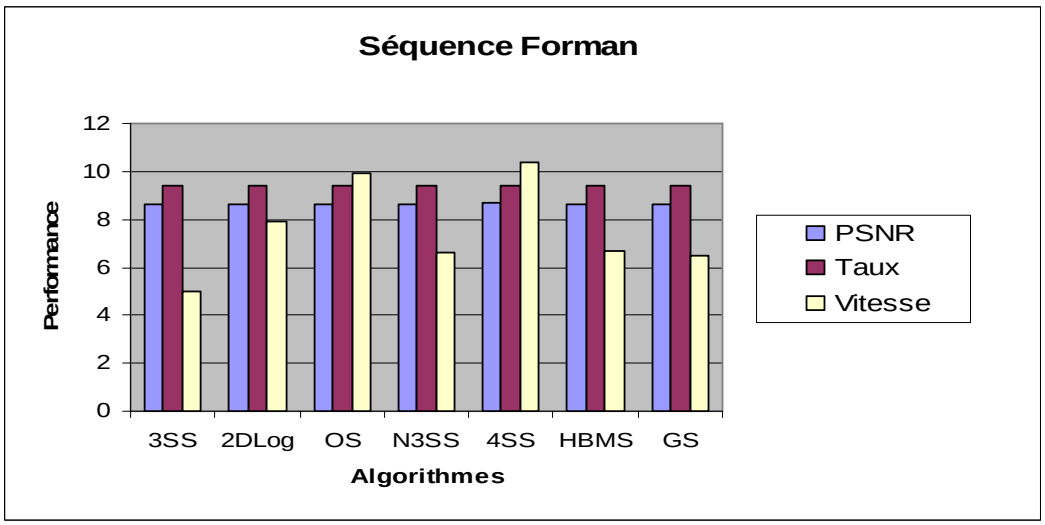
recherche très réduites telles que dans les séquences très lentes ou pour augmenter la précision dans des zones précises des images.

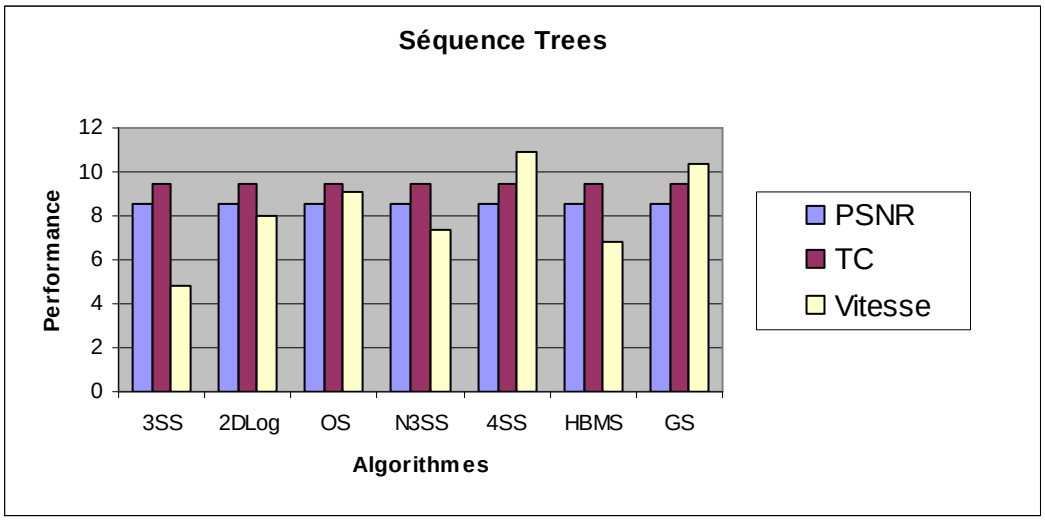
Les performances des autres algorithmes diffèrent d'une séquence à l'autre. Dans les graphes suivants (Figure 49), les valeurs des PSNR et du Taux de compression de compression et du nombre de blocs sont normalisées pour pouvoir comparer les performances pour chaque séquence:

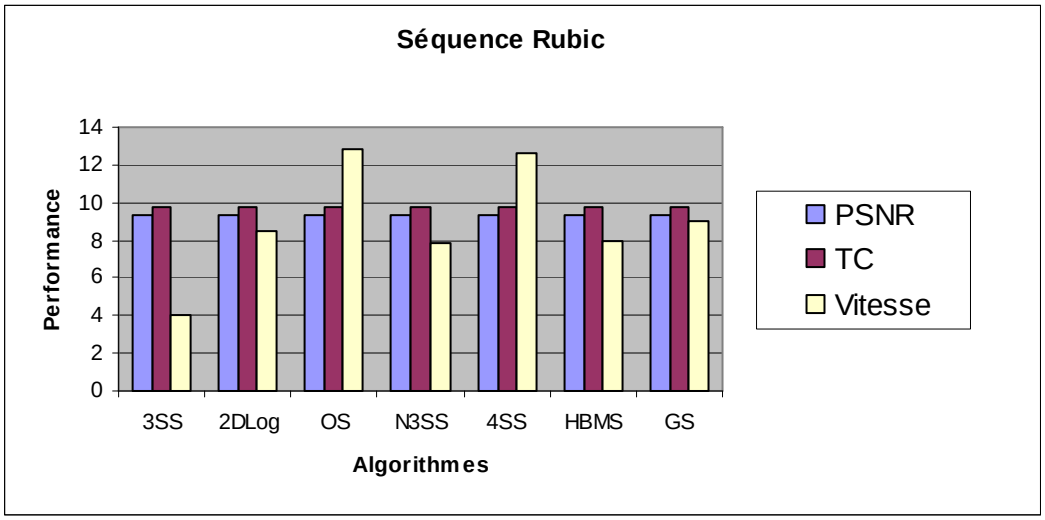


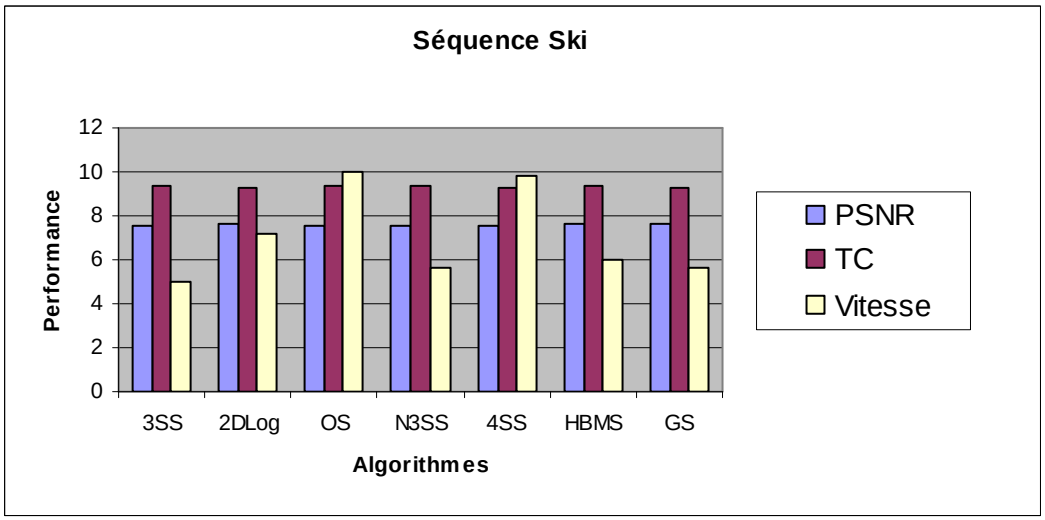



Séquence News
Performance
12
10
8
6
4
2
0
3SS
2DLog
OS
N3SS
4SS
HBMS
GS
Algorithmes
PSNR
TC
Vitesse


Séquence Forman
Performance
12
10
8
6
4
2
0
3SS
2DLog
OS
N3SS
4SS
HBMS
GS
Algorithmes
PSNR
Taux
Vitesse


Séquence Trees
Performance
12
10
8
6
4
2
0
3SS
2DLog
OS
N3SS
4SS
HBMS
GS
Algorithmes
PSNR
TC
Vitesse





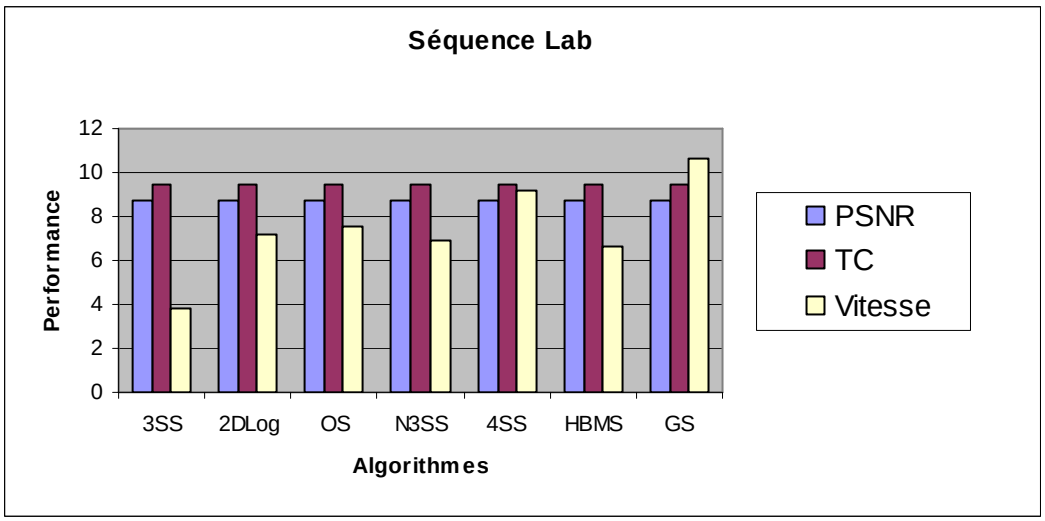


Figure 49:Comparaison des algorithmes de BM

Dans le cas des séquences rapides (Carphone, Ski, Rubic) l'algorithme en quatre pas 4SS peut donner des PSNR et des taux de compression similaires et meilleurs que les autres algorithmes en un temps très réduit. Dans le cas des séquences lentes (Miss America, Trees, Lab)

l'algorithme de recherche descendante de gradient GS, est plus avantageux puisque les meilleurs blocs sont très proches du bloc cherché.

Dans les séquences contenant les deux types de mouvement rapide et lent (News, Forman, Lab, Rubic), la vitesse peut être améliorée en considérant des méthodes de recherche adaptées pour chaque type de bloc.

III.6.Conclusion

Dans ce chapitre nous avons détaillé le principe de Block Matching utilisé dans la compression vidéo et nous avons vu que cette méthode est largement utilisée dans la plupart des standards, mais est toujours en face du compromis précision/temps de calcul. Les algorithmes de Block Matching ont été explorés et comparés, et nous avons vu leurs avantages et inconvénients.

Dans la section suivante, nous proposons une technique qui permet de profiter du maximum des avantages de chacun de ces algorithmes selon le cas du bloc en vue de compression.

Quatrième chapitre:

Méthode hybride pour l'estimation du mouvement

IV.1. Introduction

Généralement, les séquences vidéo contiennent des objets qui changent et se déplacent d'une façon presque constante entre deux images consécutives, c'est-à-dire que les objets gardent la même allure de mouvement (vitesse et direction) pour un certain nombre d'images. En plus, dans la même image, on trouve des objets en mouvement et d'autres statiques, et même ceux en mouvement diffèrent dans leur mouvement rapide ou lent.

Dans ce chapitre, ces caractéristiques seront exploitées pour appliquer pour chaque recherche d'un bloc, l'algorithme qui convient (méthode hybride) afin d'augmenter la vitesse d'estimation de mouvement et la précision des vecteurs de mouvement obtenus, ce qui augmente la qualité des images fournies et le taux de compression.

IV.2. Principe de la méthode hybride

L'utilisation du même algorithme pour toute la séquence vidéo s'avère très coûteux et ne présente pas une solution optimale de vue aux diversités existantes entre les images voire au sein de la même image. Les blocs de la même image n'ont pas le même caractère de mouvement, ce qui nous oblige à classer les blocs au lieu de classer les séquences. Un bloc est classé donc à caractère de mouvement rapide, lent ou stationnaire pour ensuite choisir l'algorithme qui convient pour la recherche du bloc original.

Dans les séquences du monde réel, les blocs des images gardent la même allure de mouvement pour au moins un certain nombre d'images pour qu'elles puissent être aperçues [CHOK 98].

Avant d'estimer donc le vecteur de mouvement d'un bloc, on doit estimer son caractère de mouvement en se basant sur les informations collectées lors du codage des images précédentes et des blocs voisins précédents de la même image pour ensuite choisir la bonne méthode de recherche et la zone de recherche optimale. Et pour se rapprocher le plus possible des coordonnées du bloc original on doit d'abord estimer le point de départ de la recherche.

IV.2.1. Estimation des types des blocs

L'estimation du type d'un bloc consiste à son classement parmi les trois types suivants: stationnaire, lent, ou rapide. Ce classement se base sur deux similitudes de mouvement entre les blocs:

IV.2.1.1. Similitude spatiale

Les blocs voisins (blocs:(i,j), (i-1,j) ,(i-1,j-1), (i,j-1) de la même image) ont une grande probabilité d'avoir les mêmes vecteurs de mouvement ou des vecteurs très proches tel que le cas des objets englobant plusieurs blocs (Figure 50).

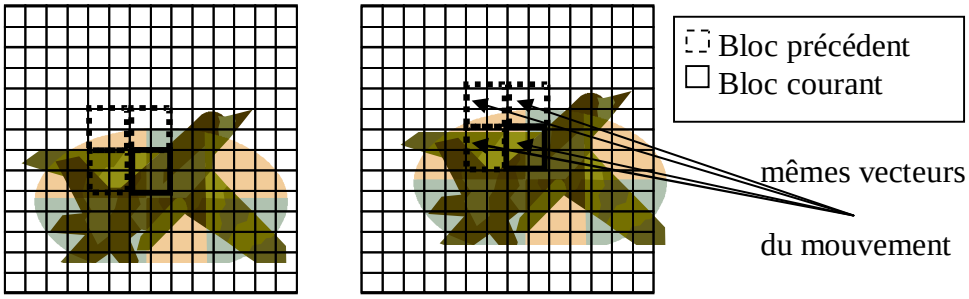


Figure 50:Similitude des vecteurs de mouvement des blocs voisins spatialement

IV.2.1.2. Similitude temporelle

Le même bloc dans deux images qui se suivent peut garder le même vecteur de mouvement ou un vecteur très proche (***Figure 51***).

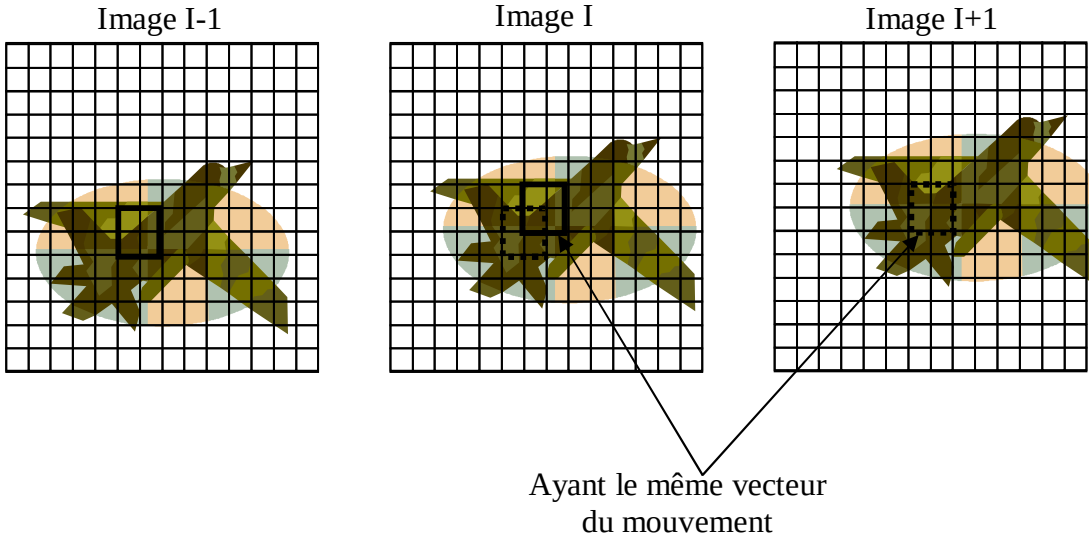


Figure 51:Similitude des vecteurs de mouvement des blocs voisins temporellement

En utilisant les valeurs des vecteurs du mouvement des blocs précédents déjà estimés (blocs voisins spatialement et temporellement), on peut classer un bloc comme:

- **Bloc stationnaire**

C'est un bloc qui garde les mêmes coordonnées que dans l'image référence c'est-à-dire que son vecteur de mouvement est (0,0); pratiquement, ce sont les blocs des objets fixes ou des arrière plans. On peut classer un bloc comme stationnaire si la moyenne des vecteurs de mouvement de ses voisins est nulle.

- **Bloc lent**

Les études statistiques sur les séquences du monde réel montrent que plus de 80% des blocs se déplacent dans une zone de 3x3 pixels (Figure 52) [CHOK 98].

Un bloc est classé lent si la moyenne des vecteurs du mouvement de ses blocs voisins ne dépasse pas la zone de 3x3 c'est-à-dire: $| x | \leq 3$ et $|y| \leq 3$.

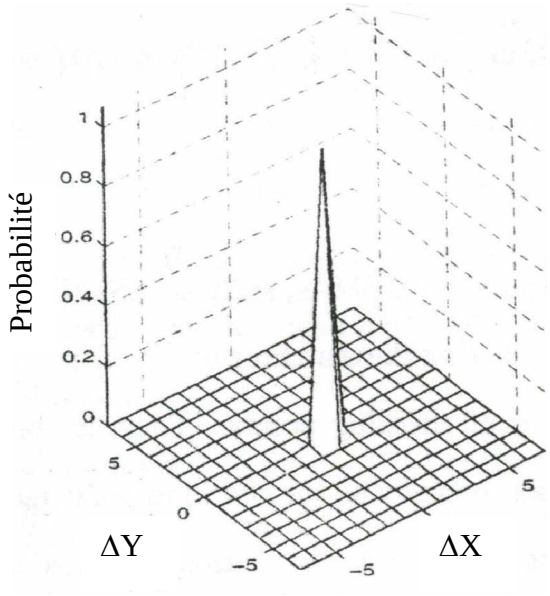


Figure 52:Distribution des vecteurs de mouvement dans les séquences du monde réel

- **Bloc rapide**

C'est un bloc où la moyenne des vecteurs du mouvement de ses blocs voisins dépasse la zone de 3x3 pixels.

IV.2.2. Estimation de la méthode de la recherche

Après le classement du bloc comme stationnaire, lent ou rapide, on peut choisir la méthode de recherche en block matching qui convient:

- Si le bloc est stationnaire alors son vecteur de mouvement est (0,0).
- Si le bloc est lent, on utilise la méthode de recherche descendante de gradient (GS) pour trouver le vecteur de mouvement.
- Si le bloc est rapide, on utilise la méthode à quatre pas (4SS).

Et pour éviter les erreurs dus aux mouvements inattendus dus à l'apparition de nouveaux blocs ou au changement du mouvement des blocs, on mesure la distorsion du bloc et on approfondie la recherche selon sa valeur. Dans le cas d'un bloc stationnaire, on mesure la distorsion avec le bloc original si elle dépasse le seuil de distorsion minimal alors on passe à un bloc lent. Et dans le cas d'un bloc lent, si la mesure de distorsion dépasse le seuil minimal de distorsion alors on passe à un bloc rapide.

IV.2.3. Estimation de la zone de recherche

L'adaptation de la zone de recherche au type du bloc est très importante puisqu'elle permet de limiter considérablement le nombre de blocs testés. La largeur de la zone de recherche est choisie donc selon le type du bloc.

- Les blocs stationnaires n'ont pas besoin de recherche.
- Les blocs de caractère de mouvement lent ont une zone de recherche de largeur égale à 3 selon les statistiques sur les vecteurs du mouvement des séquences du monde réel (***Figure 52***).
- Les blocs rapides sont cherchés dans une zone de largeur égale à la taille d'un macro bloc, ou plus selon les moyens utilisés pour le calcul.

IV.2.4. Estimation du point de départ de la recherche.

L'estimation du type du bloc et le choix de la méthode et de la zone de recherche permettent de réduire le nombre de blocs candidats. Une amélioration consiste à estimer aussi le point de départ de la recherche.

Le choix du point de départ de la recherche dans la zone de recherche est très important; plus il est proche du point optimal, plus la recherche est rapide. Dans les algorithmes présentés précédemment, la recherche du bloc original dans l'image référence commence toujours à partir des coordonnées du bloc cherché dans l'image en cours. On ne considère pas les informations obtenues des estimations précédentes.

On propose ici de commencer la recherche à partir des coordonnées du bloc lui-même décalées par le vecteur du mouvement moyen de ses blocs voisins spatialement et temporellement (Figure 53).

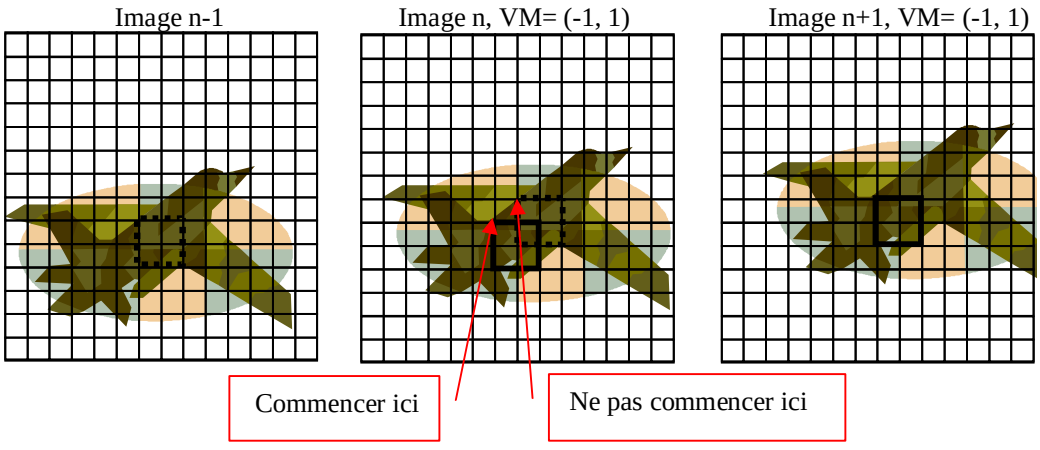


Figure 53:Estimation du point de départ

Toutes ces techniques nécessitent de garder une table des vecteurs du mouvement trouvés et la mettre à jour chaque fois qu'on estime les vecteurs d'une image, cette table contient les vecteurs de mouvement trouvés des blocs dans les estimations précédentes.

IV.2.5. Cas des images P

Dans le cas de codage bidirectionnel, la référence ne précède pas immédiatement l'image codée; elles peuvent être séparées par une, deux, trois images voire plus. On code des images futures avant des images déjà reçues (Figure 54)

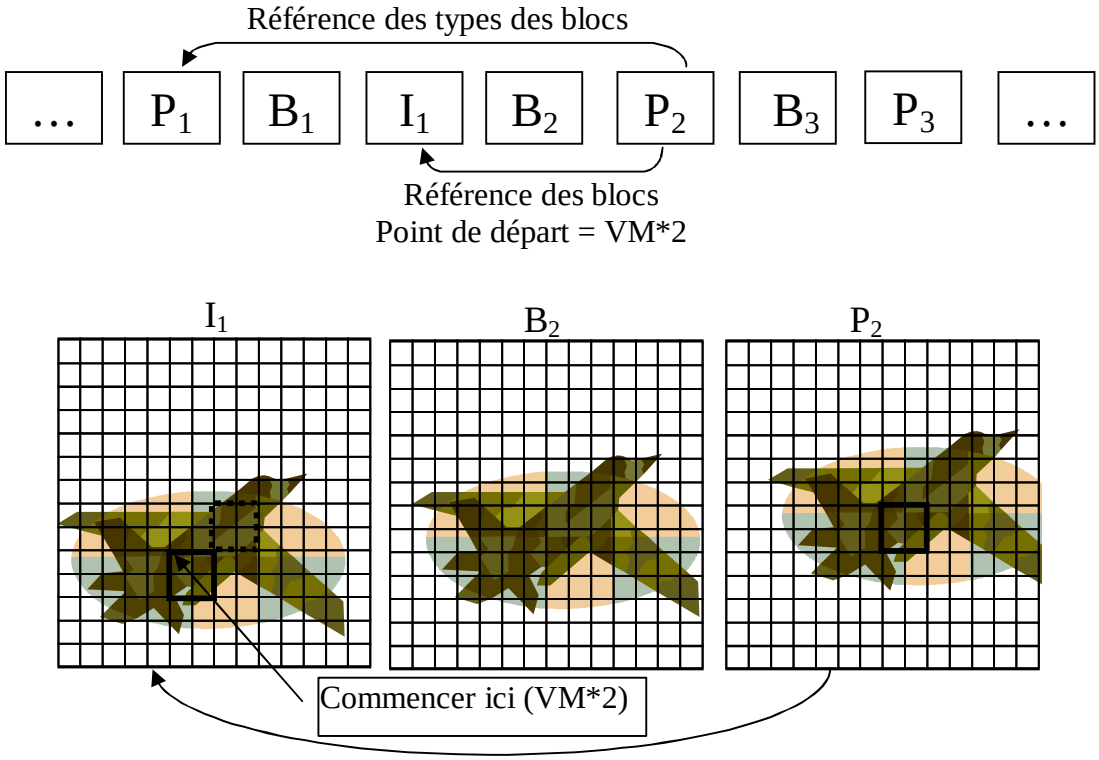


Figure 54:Estimation des blocs P avec la méthode hybride

Dans ce cas, la table des vecteurs de mouvement n'est mise à jour que lors du codage des images P car le codage des images I ne contient aucune estimation, et le codage des images B dispose d'informations antérieures et postérieures sur le bloc, et par conséquent l'estimation sera plus facile.

Par exemple (Figure 54), lors du codage de l'image P2, on référence l'image I1 en utilisant:

- Pour l'estimation des types des blocs, les vecteurs de mouvement des blocs voisins dans P_2 et celui du vecteur trouvé pour le même bloc dans P_1 (dernière estimation),
- Pour l'estimation du point de départ, on multiplie le vecteur utilisé lors de l'estimation du type par la distance séparant des deux images I_1 et P_2 et qui est égale à 2 (***Figure 54***).

Algorithme considéré

Permet d'estimer le vecteur de mouvement d'un bloc de coordonnées (X,Y) d'une image P dans une image précédente d'une distance Dist:

***EstimationP* (X, Y, Dist)**

1: Calculer $(\overline{X},\overline{Y})$ moyenne des vecteurs du mouvement des blocs :

(X-1, Y-1), (X-1, Y), (X,Y-1) et (X,Y) de la table des vecteurs du mouvement

2: si $(\overline{X},\overline{Y}) < (1,1)$ alors Aller à 5

3: si $(1,1) \le (\overline{X},\overline{Y}) \le (3,3)$ alors Aller à 6

4: si $(\overline{X},\overline{Y}) > (3,3)$ alors aller à 8

5: {*Bloc stationnaire*}

Calculer la MDB entre le bloc (X,Y) de l'image courante et le bloc (X,Y) de l'image Référence; si MDB<MinMDB alors retourner (0,0)

6: {*Bloc lent*}

Rechercher (Δx, Δy) par la méthode GS dans une zone de 3x3 pixels en commençant à partir du point $(X*16-\overline{X}*Dist, Y*16-\overline{Y}*Dist)$.

7: Calculer MDB entre le bloc (X, Y) de l'image courante et le bloc (X+Δx, Y+Δy) de l'image précédente, si MDB < MinMDB alors retourner (Δx, Δy)

8: {*Bloc rapide*}

Rechercher (Δx, Δy) par la méthode 4SS dans une zone de 16x16 pixels en commençant à partir du point $(X*16-\overline{X}*Dist, Y*16-\overline{Y}*Dist)$

9: Retourner (Δx, Δy).

Chaque vecteur retourné est mis directement dans la table des vecteurs du mouvement pour le bloc correspondant.

IV.2.6. Cas des images B

Dans le cas des images B, on dispose d'une référence passée et une référence future, ce qui nous donne plus de confiance sur le type du bloc et nous permet de ne faire aucun approfondissement de la recherche, on arrête donc sur la première décision (Figure 55):

- Les blocs stationnaires restent stationnaires.
- Les blocs lents et rapides sont cherchés dans l'image passsée et future en considérant la distance les séparant pour les points de départ.

- Aucune mise à jour n' est effectuée sur la table des vecteurs de mouvement puisque les blocs sont dépassés par l'estimation.

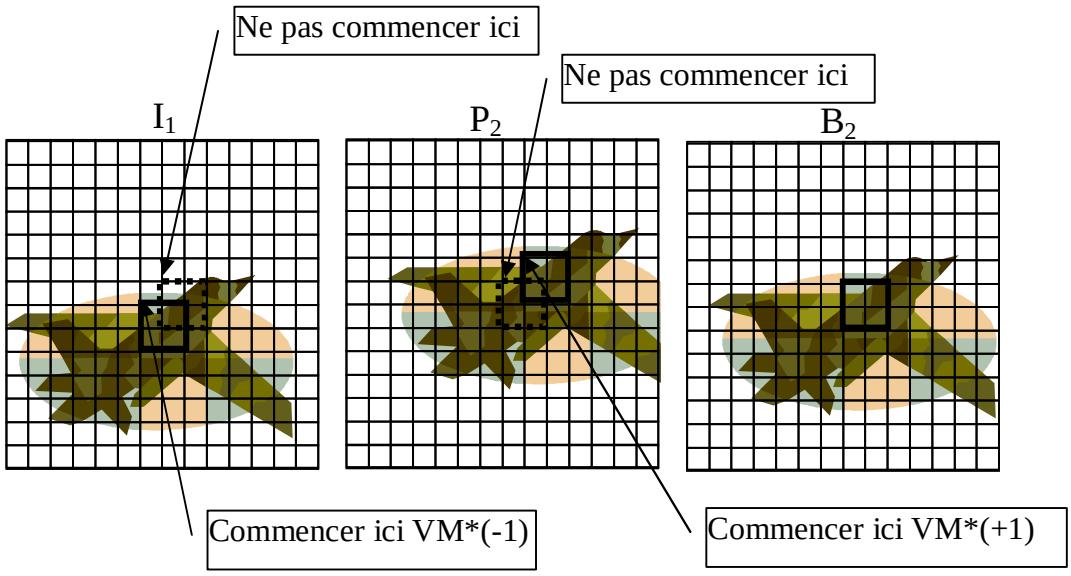


Figure 55:Estimation des blocs B par la méthode hybride

Algorithme considéré

Il permet d'estimer un bloc de coordonnées (X,Y) d'une image B dans une image référence précédente d'une distance Dist (négative) ou une image suivante d'une distance Dist (positive)

***EstimationB* (X, Y, Dist)**

1: Calculer $(\overline{X},\overline{Y})$ moyenne des vecteurs du mouvement des blocs :

(X-1, Y-1), (X-1, Y), (X,Y-1) et (X,Y) de la table des vecteurs du mouvement.

2: si $(\overline{X},\overline{Y}) < (1,1)$ alors retourner (0,0)

3: si $(1,1) \le (\overline{X},\overline{Y}) \le (3,3)$ alors Aller à 5

4: si $(\overline{X},\overline{Y}) > (3,3)$ alors aller à7

5: {*Bloc lent*}

Rechercher (Δx, Δy) par la méthode GS dans une zone de 3x3 pixels en commençant du point $(X*16+\overline{X}*Dist, Y*16+\overline{Y}*Dist)$.

6: Retourner (Δx, Δy)

7: {*Bloc rapide*}

Rechercher (Δx, Δy) par la méthode 4SS dans une zone de 16x16 pixels en commençant à partir du point $(X*16+\overline{X}*Dist, Y*16+\overline{Y}*Dist)$

8: Retourner (Δx, Δy).

Le bloc le plus ressemblant (de l'image précédente ou suivante) est pris comme bloc de référence.

IV.3. Résultats et discussion

L'utilisation de la méthode hybride permet de minimiser le nombre de blocs testés en profitant de la redondance temporelle et spatiale existant dans le comportement des blocs voisins. Cette redondance est détectée au cours de l'estimation du mouvement des blocs des images P.

La méthode hybride vise :

- La limitation de la zone de recherche d'un bloc selon le comportement de ses voisins.
- L'application de la méthode de recherche appropriée pour chaque bloc.
- Le rapprochement du point de départ de la recherche du meilleur bloc.

Dans cette section, les résultats de la méthode hybride par rapport aux méthodes du block matching sont présentés en terme de PSNR, taux de compression et nombre de blocs testés (temps de calcul). Ensuite, les paramètres propres à la méthode hybride sont présentés avec une discussion de leur importance.

Le tableau suivant (Table 6) montre les résultats obtenus en codant les séquences déjà présentées avec un modèle IBBP et une zone de recherche de 32x32 pixels:

Séquence \ Algorithme		4SS	GS	Recherche hybride	Amélioration (%)
Carphone	PSNR	33.63	33.87	33.65	-0.65
	TC	95.06	95.20	94.87	-0.35
	Blocs Testés	175383	165106	73922	55.23
Miss America	PSNR	33.38	35.05	33.11	-5.86
	TC	99.57	99.56	99.56	-0.01
	Blocs Testés	794797	1113395	168124	78.85
News	PSNR	35.16	35.26	35.16	-0.28
	TC	96.45	96.45	96.35	-0.10
	Blocs Testés	157152	152080	58677	61.42
Forman	PSNR	35.24	35.89	35.62	-0.76
	TC	94.31	94.49	94.01	-0.51
	Blocs Testés	239314	315127	185885	22.33
Trees	PSNR	34.48	34.81	32.92	-5.74
	TC	94.07	94.51	93.14	-1.47
	Blocs Testés	101704	97225	39018	59.87
Rubic	PSNR	37.76	38.08	38.16	0.21
	TC	97.68	97.86	97.20	-0.68
	Blocs Testés	70194	73393	13294	81.06
Ski	PSNR	30.01	30.43	30.31	-0.40
	TC	92.91	92.71	92.25	-0.72
	Blocs Testés	197767	277776	121760	38.43
Lab	PSNR	34.09	34.29	34.33	0.12
	TC	98.20	98.26	98.01	-0.26
	Blocs Testés	138182	141679	39029	71.76
Moyenne	PSNR	34.22	34.71	34.16	-1.67
	TC	96.03	96.13	95.67	-0.51
	Blocs Testés	234311.63	291972.63	87463.63	58.62

Table 6: Résultats du codage par la méthode hybride

La méthode hybride permet d'améliorer le temps de calcul jusqu'à 81% (séquence Rubic), sans perte significative en PSNR (0.21dB) et en taux de compression (0.68%). Dans les séquences contenant beaucoup de blocs statiques telles que "Miss America", "Rubic" et "Labl" la méthode hybride donne des améliorations très intéressantes en nombre de blocs testés.

Dans la littérature [CHOK 98], une méthode adaptative est proposée. Elle utilise le classement des blocs basé sur la valeur de l'erreur entre le blocs cherché et le bloc des mêmes coordonnées de l'image de référence, si cette valeur est inférieure à MinBDM alors le blocs est considéré comme stationnaire, si elle se situe entre MinBDM et MaxBDM alors il est considéré comme lent, sinon c'est un bloc rapide. Cette méthode n'a pas pu dépasser les limites de 48.11 % pour l'amélioration du nombre de blocs testés.

L'amélioration est liée au nombre d'images B utilisées aussi, puisque le poids de cette méthode réside dans l'annulation des approfondissements de recherche dans l'estimation de ces images. En fait, les images B disposent d'informations sur les images antérieures et des images postérieures ce qui donne plus de confiance pour les vecteurs de mouvement moyens utilisés. Le graphe suivant (Figure 56) montre l'évolution de l'amélioration en nombre de blocs testés en fonction du nombre d'images B.

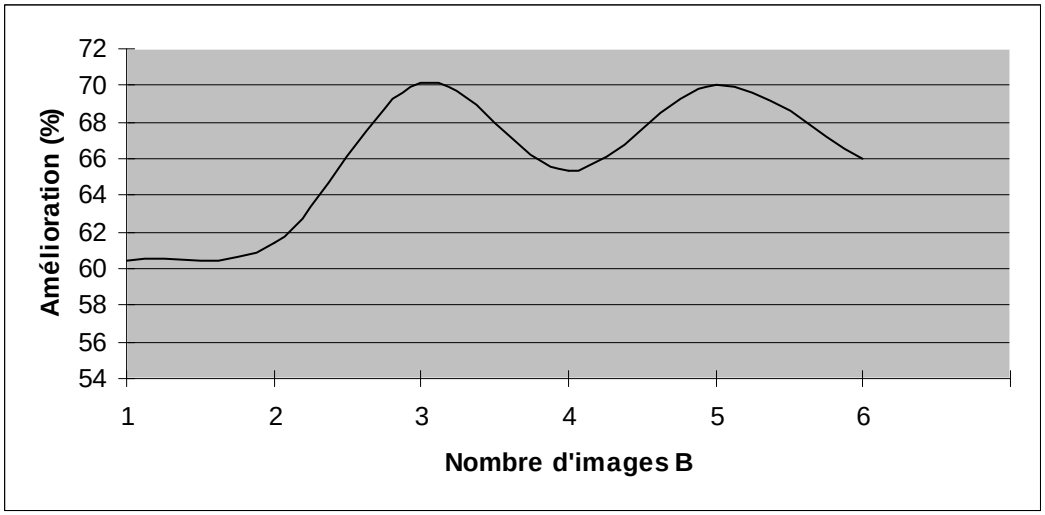


Figure 56:Amélioration du temps de calcul en fonction du nombre d'images B

Après 3 images B on commence à perdre en PSNR, puisqu'on utilise des références plus lointaines.

Le nombre des opérations nécessaires pour la recherche est, dans les cas le plus défavorable, celui de l'algorithme 4SS en cas des mouvements larges, mais dans une séquence lente, le nombre est très réduit.

Les opérations de décision relatives au choix de la méthode ou du point de départ sont négligeables en les comparant avec les opérations de comparaison des blocs ou les opérations de leur codage.

IV.4. Autre accélération: La recherche par interruption

La recherche d'un bloc dans la zone de recherche peut être interrompue dès qu'on trouve un bloc dont la MDB avec le bloc cherché est inférieure au minimum de ressemblance MinMDB. Ce qui permet de négliger un nombre important de test quelque soit l'algorithme utilisé. Le paramètre MinBDM est fixé en fonction de la précision cherchée et en fonction de la mesure de correspondance choisie; il représente le seuil de la MDB inférieur pour lequel deux blocs sont considérés identiques.

Le choix de cette accélération est optionnel, puisqu'il peut représenter un obstacle devant les meilleurs blocs qui peuvent exister dans la zone de recherche et empêcher la recherche de les atteindre ce qui peut diminuer le taux de compression et le PSNR qui peuvent être atteints.

Le tableau suivant (Table 7) montre l'amélioration apportée au nombre des blocs testés et qui peut atteindre plus de 20%.

Séquence \ Algorithme		Recherche hybride	Recherche Hybride avec interruption	Amélioration du temps de calcul (%)
Carphone	PSNR	33.56	33.35	-0.63
	TC	94.93	94.91	-0.02
	Blocs Testés	73922	57958	21.60
News	PSNR	33.59	33.42	-0.51
	TC	96.82	96.82	0.00
	Blocs Testés	56536	46155	18.36
Rubic	PSNR	38.71	38.59	-0.31
	TC	97.00	97.00	0.00
	Blocs Testés	13486	13082	3.00

Table 7: Amélioration avec la recherche par interruption

IV.5. Conclusion

Dans ce chapitre, on propose une méthode hybride basée sur la ressemblance entre les images, non du coté du contenu, mais du coté des comportements des blocs. Cette technique exploite les informations qui peuvent être collectés lors de l'estimation du mouvement des blocs.

La méthode hybride permet de tirer avantage de chacun des meilleurs algorithmes du Block Matching (GS, 4SS) en évitant ses inconvénients, pour améliorer la vitesse de compression sans toucher d'une façon significative le taux de compression et la qualité des séquences décodées.

On propose dans cette méthode l'estimation des types du mouvement des blocs afin de choisir la méthode de block matching qui sera utilisée pour chercher leurs références et la taille de la zone de recherche utilisée. On propose aussi d'estimer un point de départ de la recherche dans cette zone.

Conclusion et perspectives

V.1. Conclusion

Le domaine du multimédia ne cesse de se développer, et ses applications se multiplient de jour en jour. Les images animées constituent une partie très importante de ce domaine et leur compression fait l'objet de recherche continue pour faciliter leur transmission et leur stockage par des moyens qui ne se développent pas au même rythme.

Les normes et standards actuels tels que MPEG et H26x utilisent pour le codage des images animées une méthode appelée « Estimation de mouvement » pour prédire l'image courante de l'image précédente par le calcul et le codage d'une mesure d'erreur entre les deux, et utiliser ensuite cette mesure lors du décodage pour reconstruire l'image courante.

Le calcul de cette erreur pixel par pixel est très coûteux en temps de calcul et ne s'adapte pas avec la vitesse nécessaire pour un codage en temps réel qui représente un paramètre très important pour des applications telles que la visiophonie, la téléconférence et les téléphones mobiles. La solution est de subdiviser l'image en blocs rectangulaires non chevauchés et de comparer les blocs de l'image courante aux blocs de l'image précédente (reconstruite) pour déterminer le bloc qui minimise l'erreur pour l'utiliser. Cette technique est appelée « Correspondance de blocs » ou « Block Matching », elle utilise plusieurs algorithmes de recherche basés sur des mesures de correspondance entre les blocs (MSE par exemple) et une zone de recherche de largeur fixe ou adaptée au type de la séquence à coder pour rechercher les blocs les plus ressemblants aux blocs de l'image courante dans une image précédente.

Dans ce travail, un encodeur des images animées couleurs permettant l'étude et la comparaison des algorithmes de block matching est réalisé en se basant sur la compression JPEG des images Intra et sur la compression prédictive et bidirectionnelle pour les images Inter. Les algorithmes de block matching les plus utilisés sont implémentés tel que Full Search, Descent gradient search, Tree Step search, four step search, Hierarchical block matching,… etc.

L'étude et la comparaison que nous avons faite sur ces algorithmes a démontré qu'ils donnent leurs meilleurs résultats pour des types précis de séquences (lentes, rapides ou statiques) et échouent dans le cas des autres types. Par exemple L'algorithme Full Search est s'adapte mieux aux zones de recherche très réduites (mouvement très lent), l'algorithme GS fonctionne mieux dans le cas des séquences lentes, tandis que l'algorithme 4SS convient pour les séquences à large mouvement (rapides).

En d'autre terme, ces algorithmes ne prennent pas en compte la diversité de mouvement qui existe les différents objets dans la même séquence.

Pour augmenter l'efficacité et la vitesse de recherche, qui est très importante pour les applications actuelles, nous avons proposé de prédire (estimer) pour chaque recherche d'un bloc la méthode, la zone et le point de départ en se basant sur le caractère de mouvement du même bloc dans les images précédentes et aussi des blocs voisins dans l'image courante.

Nous proposons donc pour la recherche une méthode hybride qui utilise pour les blocs lents l'algorithme GS et l'algorithme 4SS pour les blocs rapides.

Cette technique a permet de diminuer le nombre de blocs comparés pour l'estimation de mouvement jusqu'à 81% dans le cas de la séquence "Rubic".

Notre travail a permet donc de:

- Etudier les principes, les normes et les techniques de codage des images animées
- Etudier les méthodes d'estimation de mouvements.
- Réaliser un encodeur vidéo basé sur l'estimation de mouvement et y implémenter les algorithmes les plus utilisés.
- Etudier et comparer l'efficacité de ces algorithmes.
- Proposer une méthode hybride qui permet de tirer avantage des meilleurs algorithmes en but d'augmenter la vitesse d'estimation de mouvement.

V.2. Perspectives

Au nivaux de l'encodeur réalisé, les résultats de codage (taux, PSNR, vitesse) peuvent être améliorés en raffinant plus le codage Intra en utilisant les ondelettes par exemple. La partie audio reste à étudier et à développer conjointement avec une partie de synchronisation dans l'encodeur pour harmoniser les images décodées avec le son.

Au niveau de l'estimation de mouvement, les algorithmes de block matching y compris la méthode hybride peuvent être l'objet de parallélisation en vue de leur accélération [Mart 97], vue la similarité de traitement qui existe entre les blocs.

Au nivaux de la méthode hybride qui représente un pas vers l'apprentissage des images précédentes pour estimer le type de mouvement des blocs dans une séquence animée, les techniques les plus avancées de l'apprentissage tels que les réseaux de neurones peut être utilisées en prenant en compte les caractéristiques des séquences vidéo.

Bibliographie

[BARR 93] J.L Barron, D.J. Fleet, S.S. Beauchemin, T.A Burkitt, Performance of Optical Flow Techniques, Department of Computer Science, University of Western Ontario, July 1993.

[BOTT 01] Vincent Bottreau, Marion Benetière, Boris Felts, Béatrice Pesquet-Popescu, Full Scalable 3D SubBand Video Codec. Laboratoire d'électronique Philips, Video and communication Group, France 2001.

[BOUT 87] Patrick Bouthemy, Estimation et structuration d'indices Spatio-Temporels pour l'analyse du mouvement dans une séquence d'images, Traitement du signal volume 4 n° 3, IRISA 1987.

[CHAN 95] M.K.Mandal, E.Chan, X.Wang and S.Panshanathan, Multiresolution Motion Estimation Techniques for Video Compression, University of Ottawa Canada 1995.

[CHOK 98] Heung Chok Kwan, Fast Motion Estimation Techniques for Vidéo Compression, Master of Philosophy Thesis, City University of Hong Kong, July 1998.

[CRES 01] Sébastien Crespin, La compression des images animées, Université de Montpellier II, Dec 2001.

[EVAN 99] A.N.Evans, Y.Guo, D.M.Monro, Limited Motion Estimation Scheme for Multimedia Video, Compression, ICECS UK 1999.

[FANG 01] Ye Hui Wang, Guo Fang Tu, Fast Binary Block Matching Motion Estimation Using Efficient One-Bit Rate Transform, International Conference on Signal Processing, China 2001.

[GALP 02] Frank Galpin, Représentation 3D des séquences vidéo, Thèse PHD, Université Rennes I, Janv 2002.

[GIRO 97] P.Eisert and B.Girod, Model-Based 3D-Motion With illumination Compensation, IPA97, 15-17 July 1997 Conference Publication N° 443 P 194, IEEE 97.

[GOSS 97] A.Druet, N.Gosset, J.A.Robinson, Binary-Tree Recursive Motion Estimation For Video Coding, IPA97, 15-17 July 1997 Conference Publication N° 443 P 51, IEEE 97.

[H26x] H263 vidéo coding, and H261 video coding, at www.Mobile-ecs.soton.ac.uk

[HAIB 94] Haibo Li, Astrid Lundmark, and Robert Forchheimer, Image Sequence Coding at Very Low Bitrates: A Review, IEEE Transactions on Image Processing, Vol 3, N° 5, Sept 94.

[HANZ 94]L.Hanzo, D.Chandler, Performance of the MPAG2 Codec, University of Southampton, 1994.

[HERV 95] Hervé Guitter, *La compression des images numérique*, Edition Hermes 1995.

[ITU 93] ITU-T SG15, Video Codec for Audiovisual Service at Px64 Kbits/s, In ITU-T recommendation H.261 Version 3, Mars 1993.

[JAIN 02] Jain. E.G.Richardson, Introduction to Image and Video Coding, 2001-2002 at www.vcodex.com

[KEES 96] Georjan Keesman, Improvement of MPEG-2 Video Compression Based on Rate Distortion Concepts, Philips Research Labs, Feb 1996.

[KUHN 99] P.M.Kuhn and All, Complexity and PSNR-Comparaison os several Fast Motion Estimation Algorithms for MPEG-4, Technical University of Munich, Germany 99.

[LABI 93] Henri Nicolas, Claude Labit, Motion and Illumination Variation Estimation Using Hierarchy of Models, IRISA N° 742, Juin 1993.

[LAUR 98] Laurend Bonnaud, Schéma de suivi d'objets vidéo dans une Séquence Animée, Université de Rennes I, Octobre 1998.

[LIN 95] T.Lin, J.L.Barron, Image Reconstruction Error for Optical Flow, University of Western Ontario Canada 1995.

[MART 97] M.K.Sterialos, G.R.Martin, R.A Packwood, Parallelization of Block Matching Motion Estimation Algorithms, Research Review N° 320, Jan 1997.

[MEMI 96] E.Memin, P.Perez, Champs de Markov Multirésolution et Algorithmes Multigrille pour l'Estimation du Mouvement, INRIA 1996.

[MOSC 01] Fulvio Moschetti, A Statistical Approach to Motion Estimation, Thèse PHD, Ecole polytechnique Fédérale de Lausanne 2001.

[MPG4] Présentation de MPEG4, at www.membres.lycos.fr/benoitluttringer/

[NELS 93] Mark Nelson, *La compression des données textes, images et son*, Edition DUNOD, Paris 1993.

[PACK 97] R.A.Packwood, M.K.Steliaros and G.R.Martin, Variable Size of Block Matching Motion Compensation for Object-Based Video Coding, IPA97, 15-17 July 1997 Conference Publication N° 443 P 56, IEEE 97.

[PATR 02] I.Patrs, E.A.Hendriks, R.L.Lagendijk, Confidence Measure for Block Matching Motion Estimation, IEEE ICIP 2002.

[RICHT 01] H.Richter, A.Smolie, B.Sstabernack, E.Müler, Real Time Global Motion Estimation For an MPEG-4 Video Encoder, University of Rostock, Germany 2001.

[SEOK 01] Seok-Woo Jang, Mark Pomplun, and Hyung-II Choi, Adaptative Robust Estimation of Model Parameters from Motion Vectors, Proceedings of International Conference on Rought Sets and Current Trends in Computing, Banff, Canada, Vol 2005, December 2001 p 438-441.

[SHAN 97] Shankar, L.Regunathan, Kenneth Rose, Robust Video Compression for Time-Varying Wireless Channels, University of California, Santa Barbara 1997.

[SHAO 97] Vivian Shao, Implementation the Color Space Transformation Algorithm Using the TMS320C2xx, Digital Signal Processing Solutions, Texas Instruments, March 1997.

[TOUR 00] Prof. Touradj Ebrahimi, Compression vidéo, Signal Processing Laboratory Swiss Federal Institute of Technology, Lausanne, 2000.

[VCMP] video compression, at www.cs.sfu.ca

[VLAN] Codage des Images Animées, at www.videolan.org

[VOLD 94] Espen Volden, Gerard Giraudon, Marc Berthod, Modeling Image Redundancy, INRIA N° 2440, Dec 1994.

[WENG 93] J.Weng, T.S.Huang, N.Ahuja, Motion and Structure from Image Sequences, Springer Series in Information Sciences, Berlin 1993.

[WING 02] Guy Coté, Lowell Winger, Progrès récents Dans le domaine de la compression vidéo, IEEE Canadian Review Printemps 2002.

[XAVI 98] Xavier Marichal, Motion Estimation and Compensation for Very Low Bit Rate Coding, Thèse PHD, Université Catholique de Louvain, Mai 1998.