Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

DER- Génie Electrique & Informatique
Département d'Electronique

Thèse

Présenté par

M^r ZEKRINI Benyahia, Ingénieur d' Etat en Electronique

En vue d'obtenir le Grade
de Magister en Electronique

Option: Acquisition & Traitement de l'Information

THEME

COMPRESSION D'UNE SEQUENCE D'IMAGES PAR LA CACHE VQ ET COMPENSATION DE MOUVEMENT

Soutenue le ..... devant le Jury composé de :

Président:	Mr	A.FARAH	Professeur à L'ENP
Rapporteur:	Mr	D. BERKANI	Maître de Conférences à L'ENP
Examinateurs:	Mr.	A. BELOUCHRANI	DR, Enseignant à L'ENP
	Mme.	L. HAMAMI	Chargée de Cours à L'ENP
	Mr. .	L. SAADAOUI	Chargé de Cours à L'ENP

Novembre -1998

ملخص

في عملنا هذا نقدم خوارزميات لتشفير و ضغط سلسلة من الصور. تشفير سلسلة من الصور يتطلب استغلال التشابه في نفس الصورة, أي التشابه الفضائي مثل الصور الثابتة, واستغلال التشابه بين الصور أي التشابه الزمني. في التشفير الفضائي استعملنا طريقة التكميم الشعاعي المختفي التي تتناسب مع سلسلة الصور والتي تمكننا من تقليص التدفق. نطبق بعد ذلك تشفيرا زمنيا بين الصور, حيث يتم استخراج شعاع الحركة بكاشف للحركة, حيث استعملنا النموذج التآلفي و طريقة الاشكال المتشابهة السريعة, بعد ذلك نستعمل تعويض الحركة للتقليل من التدفق.

الكلمات الرئيسية: التكميم الشعاعي المختفي, الضغط, التشفير داخل الصورة وبين الصور, تقدير الحركة, الاشكال المتشابهة, النموذج التآلفي, تعويض الحركة.

Abstract

In this work we present algorithms for coding and compression of a sequence of images. Sequences coding consists of exploiting the spatial correlation, like the case of still images, and tje temporal correlation witch is very high in this type of images. In the intraframe coding we have udes the cache VQ, witch is very convinient to the case of images sequence, and it offers a low bit rate. After, an interframe coding is applied, it consists of extracting the motion vector using a motion detection bloc, in our case we have used the affine model and the fast bloc matching algorithms, followed by a motion compensation to reduce the bit rate.

Keywords: Cache vectorial quantification, compression, intraframe/interframe coding, motion estimation, bloc matching, affine model, motion compensation.

Résumé

Dans ce travail nous présentons des algorithmes de codage et compression d'une séquence d'images. Le codage d'une séquence consiste à exploiter la corrélation spatiale, tout comme le cas des images fixes, et d'exploiter la corrélation temporelle qui est très élevée. Dans le codage intraframe ou spatial on a utilisé la quantification vectorielle cache qui est très adaptée au cas des séquences, elle se caractérise par un débit faible. Un codage interframe est appliqué ensuite, il consiste à extraire le vecteur mouvement par un détecteur de mouvement, dans notre cas on a utilisé le modèle affine et la correspondance de bloc rapide, ensuite une compensation de mouvement est appliquée pour diminuer le débit.

Mots clés: quantification vectorielle cache, compression, codage intraframe/interframe, estimation de mouvement, correspondance de bloc, modèle affine, compensation de mouvement.

…..je dédie ce modeste travail à la mémoire de ma mère.

Nacereddine

Avant - Propos

LE travail présenté dans ce mémoire a été effectué au laboratoire "Signal & Communications Laboratory" du Département d'Electronique, DER-GEI de l'Ecole Nationale Polytechnique (ENP), sous la direction de Monsieur D.BERKANI, Maître de conférence à l'ENP, qu'il trouve ici l'expression de toute ma gratitude pour ses conseils avisés et ses critiques qui m'ont été d'une aide précieuse, ses encouragements et ses suggestions qui ont beaucoup contribué à susciter, à enrichir et à mener à bien ce travail.

Il m'est particulièrement agréable de remercier, tous les Enseignants qui m'ont aidé, conseillé et qui, par l'environnement crée, m'ont facilité grandement la tâche.

Je remercie Monsieur le Professeur A.FARAH, pour l'honneur qu'il m'a fait d'accepter la présidence de mon jury de thèse.

Je voudrais exprimer mes remerciements les plus sincères à Monsieur le Docteur A.BELOUCHRANI, et Enseignant à l'ENP; et L.SAADAOUI, Chargé de cours à l'ENP, pour l'intérêt qu'ils ont manifesté pour notre travail, malgré leurs nombreuses tâches, et pour avoir voulu participer au jury.

Je suis très heureux de pouvoir remercier, ici, Madame L.HAMAMI, Chargée de cours à l'ENP; pour ses aides et encouragements continues, et de m'avoir fait le grand honneur d'assister au jury, malgré ses multiples occupations.

Mes remerciements vont aussi:

- A tous les membres du laboratoire qui, en toute circonstance, nous ont donné l'exemple de l'esprit d'équipe, en particulier : C.Boubakir, K.Laidi, H.Akroum, M.Damou, F.Terranti, F.Fliti,...
- A tous mes amis et collègues qui ont montré un grand intérêt à mon travail et qui m'ont encouragé à toujours aller de l'avant.
- Aux personnels du centre de documentation de l'ENP, et l'ensemble du corps administratif de la D.E.R Génie Electrique et Informatique, pour l'aide, le soutien et l'encouragement qu'ils m'ont offert, en particuliers : Krimou, Saleh, Mahmoud, Fouial, Naima , Zahia. Sans oublier le personnel du centre de calcul pour l'aide qu'ils m'ont données aux moments les plus difficiles.

SOMMAIRE

La quantification vectorielle

Le codage interframe

Applications

Liste des Notations et Abréviations Utilisées

MSE	Mean Square Error
PSNR	Peak Signal to Noise Ratio
Kbits/s	Kilo bits par seconde
Mbits/s	Mega bits pas seconde
N_{dGmax}	Niveau de gris maximum (255)
Bpp (bps)	Bits per pixel (bits per sample)
σ	Le taux de compression
RDF	Rate-Distortion Function
LBG	Linde-Buzo-Gray
PNN	Pairewise Nearest Neighbour
NNQ	Nearest Neighbour Quantizer
QMF	Quadrature Mirror Filter
RLE(RLC)	Run Length Encoding(Coding)
LZW	Lempel-Ziv-Welch
DCT	Discrete Cosine Transform
DFT	Discrete Fourrier Transform
KLT	Karlhunen Loev Transform
SQ	Scalar Quantization
VQ	Vector Quantization
PL	Prediction Linear
SBIC	Sub-Band Image coding
M/SVQ	Mean/Shape Vector Quantization
MR/VQ	Mean Residual/Vector Quantization
CVQ	Classified Vector Quantization
TVQ	Transform Vector Quantization
PVQ	Predictive Vector Quantization
FSVQ	Finite State Vector Quantization
MSVQ	Multistage Vector Quantization
TVHD	Télévision Haute Définition
CCITT	Comité Consultatif International des Télécom et Télégraphe
JPEG	Joint Photographic Expert Group
MPEG	Motion Photographic Expert Group
HVS	Human Visual System
CODEC	Codeur Décodeur
ISDN	Integrated Service Digital Network

Liste des Figures et Tableaux

Figure Page

Chapitre I

Introduction

I.1 Problématique

UNE séquence d'images est une série d'images ordonnées dans le temps. Une séquence peut être acquise par une caméra ou par magnétoscope, ou peut être générée par un logiciel informatique en ordonnant un ensemble d'images fixes en vu d'une animation. L'utilisation des séquences d'images se trouve dans plusieurs domaines tels que la communication visuelle (visiophonie et vidéoconférence), la TVHD i.e. télévision haute définition, le multimédia, l'éducation, la médecine, la surveillance, le contrôle à distance, etc.[8][28]. Les travaux de recherche dans ces domaines d'un côté, et la chute des coûts des processeurs et mémoires d'un autre côté, ont rendu l'utilisation de la TV et la Vidéo de plus en plus courante.

Dans les applications citées ci-dessus, les séquences sont soit stockées, comme dans le multimédia et la médecine, soit transmises, comme la visiophonie, la vidéoconférence et la TVHD. Le stockage ou la transmission d'une séquence telle qu'elle est, est pratiquement impossible; parce qu'une séquence d'images représente une quantité d'information énorme. Soit l'exemple suivant où l'on considère une séquence de deux minutes, avec une fréquence de 25 images par seconde, et tel que chaque point est représenté par 8 bits, i.e. 256 niveaux de gris, les dimensions de l'image sont 640 x 480. La taille de cette séquence est :

$$640 \times 480 \times 8 \times 25 \times 120 = 7.37 \text{ Gbits.}$$

Avec des images couleur de qualité et avec des séquences plus longues, la taille d'une séquence devient gigantesque, et son stockage sur les supports d'information actuels, ou sa transmission à travers les réseaux qui existent de nos jours sont impossibles (Sachant qu'une disquette peut stocker 1.44 Mo, un CDROM 650 Mo et un disque dur jusqu'à 4 Go).

Ces problèmes ont fait l'objet de travaux de recherche pendant plus de 20 ans; ils se sont concentrés sur la façon de réduire la quantité d'information que contient une séquence d'images, en adoptant des techniques de codage et de compression adéquates, qui permettent de réduire au maximum le flux de données en conservant une qualité visuelle des images acceptable ou même excellente (suivant l'application).

Les particularités de codage des séquences se différencient de celles des images fixes par :

1) La variation dans le temps et l'information de mouvement ne peuvent pas être exploitées à partir d'une image fixe ,

2) Des algorithmes très puissants et efficaces utilisent l'information temporelle, tel que le codage interframe, ou la technique de corrélation temporelle [24].

La redondance temporelle ou interframe dans les séquences est souvent plus élevée que la redondance spatiale, et elle est aussi variable parce qu'il y a des moments où le mouvement dans la séquence est très faible , même nul, ce qui amplifie beaucoup la redondance temporelle. Cela ne minimise en aucun cas l'importance du codage intraframe qui est utilisé pour avoir l'image initiale ou la frame de référence pour le codage de la séquence entière.

Dans la conception d'un codeur pour des séquences d'images on doit toujours prendre en compte les contraintes et les compromis exigés par le type d'image et l'application à envisager. Le premier compromis qu'il faut prendre en compte est le rapport taux de compression – prix du codeur, qui doit être le plus élevé possible. Est ce que le codeur fonctionne en temps réel ? Est ce que la scène change ? Est ce qu'il y a une activité importante dans la séquence ? Est ce que le débit est variable ou fixe ? La longueur de la scène, etc.

Le codage intraframe est utilisé dans une seule image où c'est la redondance spatiale qui est exploitée. On trouve plusieurs techniques de codage intraframe qui se divisent en deux classes : Les techniques réversibles, et les techniques irréversibles[24][29].
Les techniques réversibles, dites aussi techniques entropiques, n'introduisent pas une dégradation à l'image, elles permettent une restitution complète de l'image. Parmi ces techniques on trouve les algorithmes de Shannon-Fano, Huffman, LZW, RLE etc. Ces techniques, bien qu'elles permettent une restitution fidèle des images, elles ne permettent pas d'atteindre des débits faibles, pour cela elles restent appliquées aux images délicates qui contiennent des informations très importantes.

Le deuxième type de techniques constitue les techniques irréversibles, qui permettent de restituer l'image, mais avec une certaine distorsion, cette perte d'information est remplacée par un débit très faible, qui semble très intéressant à la compression des images, où la qualité est plutôt subjective.

Parmi ces techniques citons les techniques prédictives qui opèrent dans le domaine spatial de l'image[4] [5] [29]. Dans ce type de méthode un symbole est généré par prédiction à partir des symboles déjà codés, une différence ou erreur est calculée en soustrayant la valeur prédite de la valeur vraie, puis elle est quantifiée et transmise. Ces techniques sont dites à mémoire, parce qu'il faut toujours garder un certain nombre de symboles qui serviront pour estimer le symbole à coder. Ces méthodes étaient les premières à être implémentées dans des applications temps réel mettant en œuvre des images animées de type télévision [43].

Un autre type de méthodes concerne les méthodes par transformée [5] [41], ces techniques permettent de passer du domaine spatial au domaine transformé, où la distribution des coefficients permet un meilleur codage. L'application d'une transformée sur une image, ou plutôt sur les blocs d'une image permet de décorréler ces coefficients en compactant l'énergie de chaque bloc en quelques coefficients seulement, ceux proches de l'origine. Les autres coefficients non significatifs sont négligés et par la suite écartés de la transmission [24].

La transformée KLT représente la transformée optimale, mais elle n'est pas utilisée dans le domaine de l'image parce qu'elle pose de sérieux problèmes pratiques [49]. Par contre la DCT, qui se dérive de la DFT est très utilisée dans la compression de l'image vu ses caractéristiques qui sont très proches de la transformée optimale. Elle est utilisée dans le standard JPEG(voir Annexe B).

Un troisième type de méthodes concerne la quantification vectorielle [32], qui figure parmi les méthodes qui font l'objet de recherches toujours plus élaborées [1] [5]. C'est une technique vectorielle, i.e. qu'elle manipule des vecteurs au lieu de scalaires, ces vecteurs sont constitués à partir d'un rassemblage d'un ensemble d'échantillons dans la parole, ou à partir des blocs de l'image. La quantification vectorielle consiste à concevoir un dictionnaire à partir d'un ensemble de vecteurs dit d'apprentissage, ce dictionnaire sera utilisé pour le codage et le décodage des vecteurs [13] [14].

Les problèmes majeurs de la quantification vectorielle sont bien la taille du vecteur, la taille du dictionnaire et le temps de calcul énorme qu'elle nécessite, surtout au niveau du codeur, où il faut calculer la distance entre le vecteur à coder et tous les vecteurs du dictionnaire. La réduction des tailles des vecteurs et du dictionnaire peut réduire la complexité de calcul, mais cela limite l'exploitation des propriétés statistiques des échantillons. D'autres solutions plus efficaces ont été proposées, au prix d'une dégradation infime des images. Il y a des techniques qui se sont consacrées à l'organisation du dictionnaire de façon à éviter la recherche exhaustive [5] [21] [22], elles consistent à le subdiviser en plusieurs sous-dictionnaires suivant certains critères. D'autres techniques essayaient de réduire le taux excessif de calcul par un codage rapide, où des simplifications sont effectuées sur le calcul des distances. Une technique très adaptée au cas de codage des séquences d'images est la VQ adaptative, qui consiste à modifier soit le dictionnaire soit la règle de codage en fonction des caractéristiques statistiques des images .

Le codage interframe, utilisé principalement au codage des séquences d'images, consiste à exploiter la redondance temporelle qui existe entre deux frames ou plus. Cela est très profitant voir même inévitable dans certaines applications [8] [28]. Entre deux frames successives souvent un nombre limité d'objets sibissant un mouvement, et tout le reste de l'image demeure inchangé, sa transmission représente une redondance.

Parmi les méthodes qui ont été utilisées pour réduire la corrélation temporelle on cite les méthodes interpolatives qui consistent à interpoler des lignes à partir d'autres, ou même d'interpoler des frames à partir d'autres, comme le cas du standard MPEG [32] (voir Annexe B). La compensation de mouvement est une autre technique qui est très utilisée dans le domaine de codage des séquences [24] [26], elle consiste à estimer les paramètres de mouvement des objets, puis reconstituer l'image courante en utilisant les paramètres du mouvement et l'image précédente. Les paramètres de mouvement sont regroupés dans un vecteur dit vecteur de mouvement qui est lui aussi codé et transmis. Bien évidemment cette opération est appliquée seulement aux régions qui ont subit un mouvement, sinon on se retrouve avec une image plus large que l'originale. Le module clé d'un schéma de compensation de mouvement est bien le module de détection de mouvement [7] [30].

I.2 Contribution de la Thèse

Dans notre travail on s'est intéressé à l'étude de quelques algorithmes de compression et de codage de l'image, en particulier la Quantification Vectorielle. On a appliqué la technique dite Cache VQ avec plusieurs stratégies de mise à jour, comme on a fait des comparaisons entre les différents algorithmes et techniques de conception des codebooks en essayant plusieurs sources d'informations. D'un autre côté dans le codage interframe on a utilisé la technique de correspondance de blocs avec une version rapide ainsi que l'algorithme du modèle affine qui permet une détection plus fine du mouvement. Pendant la réalisation de notre travail, il nous a été nécessaire d'élaborer d'autres travaux, qu'on considère comme contributions supplémentaires de notre thèse.

I.3 Organisation de la Thèse

La suite de la thèse est organisée comme suit :

Le chapitre II expose les différentes techniques de codage et compression de données, en particulier les images, où on a fait le tour d'horizon sur les techniques entropiques et les techniques irréversibles.

Dans le chapitre III on a détaillé la notion de la VQ en donnant tous les concepts de base et les définitions, ainsi que les algorithmes nécessaires pour la conception des codebooks, et enfin on a donné quelques exemples de schémas de la VQ, en exposant la technique cache VQ utilisée.

Le chapitre IV est consacré au codage interframe, où plusieurs techniques sont présentée , avec l'exposition des différents algorithmes de détection et estimation de mouvement.

Le dernier chapitre contient les résultats de l'application des algorithmes de codage intraframe et ceux du codage interframe.

Enfin on termine par une conclusion générale, où on a interprété et discuté les résultats obtenus.

Chapitre II

Techniques de compression des images

II.1 Introduction

Le codage, dans son sens le plus large, signifie la transformation d'un message exprimé en langage clair, suivant des équivalences convenues dans un code. Même si cette définition est générale, elle s'applique aux images qui constituent un support d'information très important. L'intérêt voulu par le codage de l'image est de tirer profit des avantages qu'autorisent les technologies modernes quand il s'agit de transmettre, stocker ou traiter les images. Dans la réalisation de cet objectif une première étape est nécessaire, c'est la numérisation de l'image. La numérisation de l'image se décompose en deux opérations :

* L'échantillonnage qui consiste à transformer le signal continu en une suite d'échantillons ou points élémentaires appelés pixels.
* La quantification qui consiste à mesurer les échantillons selon un choix limité de valeurs appelées niveaux de quantification, l'ensemble de ces niveaux est appelé alphabet.

Les niveaux de quantification des échantillons d'une image sont représentés par une suite de mots binaires; pour faire une transmission rapide ou un stockage économique il faut réduire ces mots binaires à ce qui est juste nécessaire, cette opération est dite *compression.* Donc on peut dire que le codage d'image consiste en une numérisation et une compression.

II.2 Quantité d'information

L'image est un support d'information très riche (couleurs, formes, objets , déplacements...), comparée à la parole, le signal image nécessite une bande passante importante. Dans une image fixe respectant la Recommandation 601 du CCITT 4:2:2 (voir Annexe A)(qui est la norme de télévision numérique, le signal vidéo est décomposé en une composante de luminance Y de 720x576, et deux composantes de chrominance $C_b \cdot C_r$ de 360x576 chacune(C_b et C_r sont sous-échantillonnées, voir Annexe A) [24], le nombre de niveaux de chaque composante est 256), cette image nécessite un nombre de bits :

$$720 \times 576 \times 8 + 2 \times (360 \times 576 \times 8) = 6.63.\text{Mbits/pixel.}$$

Les images couleur animées sont une succession d'images fixes à une fréquence donnée, en général 25Hz ou 30Hz. Si cette fréquence est égale à 25Hz alors le débit d'une séquence d'images animées de la TV est de 166Mbits/s, en ajoutant les signaux de services ou synchronisation, le débit atteint 216 Mbits/s. De tels résultats conduisent à de sérieuses difficultés d'ordre technique et économique dans le cas de stockage ou de transmission d'images. Dans la transmission , par exemple, l'utilisation du réseau téléphonique numérique qui a un débit de 64Kb/s est impossible. On conclue qu'une réduction de débit, ou compression, est nécessaire pour exploiter les réseaux publiques (téléphonique et TVHD) pour transmettre les images.

II.3 Entropie- Théorème de codage de source sans bruit

On considère une source discrète ergodique, générant une séquence $x(n)$, où $x(n)$ prend sa valeur d'un alphabet finie A_x, pour chaque $x(k)$ est associée une probabilité d'apparence :

$$P(x(k)) = P_k \tag{2.1}$$

On appelle le terme

$$I(x(k)) = -\log_2(P_k) \text{ (bits)} \tag{2.2}$$

auto-information de $x(k)$(selfinformation). $I(x(k))$ mesure l'information apportée par la réalisation $x(k)$ de $X(n)$. L'entropie $H(x)$ d'une source est la moyenne de I :

$$H(x) = E(I(x)) = -\sum_{k=1}^{n} P_k * \log_2(P_k) \text{ bits/symbole} \tag{2.3}$$

$H(x)$ est limitée par :

$$H_0 = \log_2(N)$$

c'est à dire :

$$0 \leq H(x) \leq log_2(N) \tag{2.4}$$

H_0 est appelée aussi capacité de l'alphabet, on défini $R(x)$ la redondance de la source par

$$R(x) = H_0 - H(x) = log_2(N) - H(x) \tag{2.5}$$

II.4 La fonction Débit Distorsion (*RD*)

Il n'est pas rare qu'une transmission ou un stockage de données soit soumis à des erreurs, faibles mais réelles. Mais heureusement dans la pratique, il n'est pas toujours nécessaire de conserver la qualité initiale d'une image, parce qu'en effet le système visuel humain accepte qu'il y ait une distorsion, c'est à dire une différence entre l'image originale et l'image décodée. Il est démontré en théorie de l'information qu'il est possible de tenir compte de ces distorsions (bruits ou erreurs) pour coder l'image avec un débit inférieur à l'entropie. Donc on peut dire qu'il est possible de coder une image avec un nombre de bits inférieur à l'entropie, mais avec des distorsions, et pour avoir une image décodée acceptable il faut contrôler ces distorsions.

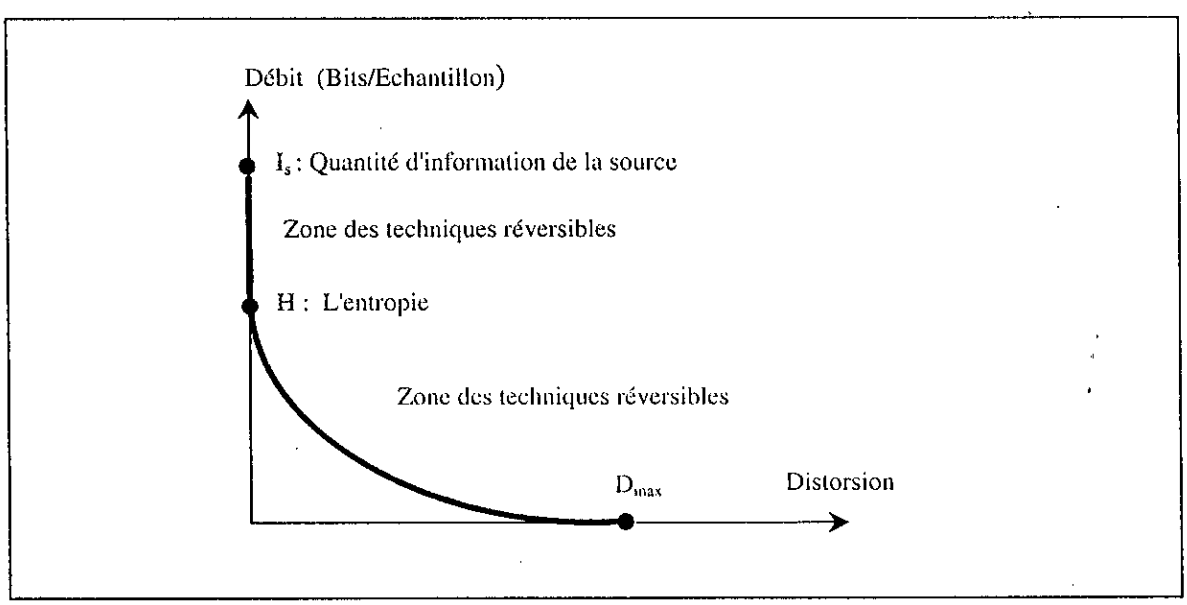


Figure II.1 *La fonction Débit/Distorsion RD* [29]

On remarque d'après la courbe qu'au-dessous de l'entropie H plus le débit diminue plus la distorsion augmente. D'après cette courbe on distingue deux zones :

- La zone sans distorsion entre I_s et H

- La zone avec distorsion entre H et 0

La zone sans distorsion : La courbe est confondue avec l'axe des ordonnées c'est à dire D=0, le décodage de l'image originale sans distorsion est possible dans cette zone. Ici l'entropie représente une limite inférieure qu'on ne peut atteindre dans pratique. Le codage entropique des images de scènes naturelles par exemple ne dépasse pas un taux σ de 2.5.

La zone avec distorsion : Dans cette zone, où D≠0, les images sont codées avec un débit inférieur à l'entropie. Les techniques qui permettent de coder une image avec un débit inférieur à H, tout en contrôlant la distorsion utilisent des méthodes avec distorsion ou des méthodes réversibles.

II.5 La distorsion

On peut dire que le seul outil capable de juger de la qualité d'une image est l'œil humain, qui est lui même intimement lié à l'individu qui le porte, donc il ne peut s'affranchir de tout aspect subjectif, tel que notre acuité personnelle, notre expérience à apprécier les images, notre éducation etc. . Des mesures de qualité d'image ont été standardisées et font intervenir des observations humaines dans des conditions rigoureusement définies validées par la suite par des algorithmes statistiques. Mais ces mesures restent coûteuses pour cela on leurs préfère les critères objectifs.

Il y a plusieurs critères objectifs, parmi eux on cite MSE (*Mean Square Error*) l'erreur quadratique moyenne, et le PSNR (*Peak Signal to Noise Ratio*) rapport signal à bruit crête, ce sont les plus utilisés dans le domaine de l'image.

a) Mean Square Error (MSE)

Soient deux images, en général l'image originale et l'image décodée, les éléments de la première image sont notés n_i et ceux de la deuxième image $\hat{n}_i$, et N le nombre de points de chaque images, l'erreur MSE est donnée par :

$$MSE = \frac{1}{N}\sum_{i=0}^{N-1}(\hat{n}_i - n_i)^2 \qquad (2.6)$$

Une image décodée est jugée de bonne qualité si MSE≤0.25, où cette valeur est déterminée par expériences.

b) Peak Signal to Noise Ratio (PSNR)

Il se déduit de la MSE :

$$PSNR = 10\log_{10}\left(\frac{N^2_{dG\,max}}{MSE}\right) \text{ en dB} \qquad (2.7)$$

avec $N_{dG\,max}$ est le niveau de gris maximum, en général vaut 255 .

II.6 Techniques de codage

Toutes les techniques de compression d'images exploitent la redondance spatiale et temporelle qui existent dans une image, fixe ou animée. Ces techniques se divisent en deux catégories :

Les techniques réversibles ou sans distorsion et les techniques irréversibles ou avec distorsions.

II.6.1 Techniques réversibles

Le théorème du codage de source sans bruit est basé sur l'utilisation de mots binaires à longueur variable; ces codes sont appelés VLC (*Variable Length Codes*). C'est un codage entropique parce qu'il nous permet de s'approcher aussi prés que l'on veut de l'entropie. Le principe de ces techniques est d'accorder aux symboles les plus probables les mots binaires les plus courts.

II.6.1.1 Algorithme de Shannon-Fano

C'est un algorithme statistique, il permet la détermination d'un VLC préfixé [31]. Son principe est le suivant :

i) Classer les symboles en fonction de leur occurrence, en plaçant les symboles les plus fréquents en premier et les plus rares en dernier.

ii) Diviser la liste en deux parties, la somme des fréquences de la partie haute doit être la plus proche à celle de lá partie basse de la liste.

iii) Affecter 0 à la partie haute de la liste, et 1 à la partie basse. Donc les codes de la partie haute commenceront par 0, et ceux de la deuxième par 1.

iv) Refaire les étapes ii) et iii) pour chaque partie résultante.

Soit l_i la longueur en bits du mot binaire M_i associé au niveau du point n_i on a :

$$\bar{l} = E(l_i) = \sum_{i=1}^{m} p(n_i) * l_i$$

d'après le théorème de codage de source sans bruit on a :

$$H(p) \leq \bar{l} \leq H(p) + \delta \qquad (2.8)$$

si on considère que les niveaux voisins sont indépendants on aura :

$$-\sum_{i=1}^{m}(p(n_i)*log_2(p(n_i)) \le \sum_{i=1}^{m} p(n_i)*l_i < -\sum_{i=1}^{m} p(n_i)log_2(p(n_i)) + \delta \qquad (2.9)$$

D'où la méthode de Shannon-Fano [29] nous donne :

$$l_i = \lceil -log_2(p(n_i)) \rceil \qquad (2.10)$$

où $\lceil\ \rceil$ dénote la partie entière.

(Démonstration Voir Jayant p146 [29]).

Sauf dans les cas extrêmement simples, cet algorithme ne permet pas d'approcher efficacement l'entropie.

II.6.1.2 Codage de Huffmann

Le code de Huffman est à longueur variable, les symboles à haute probabilité ont les codes les plus courts [30]. Cet algorithme consiste à disposer les symboles individuellement sous forme d'une chaîne aux nœuds qui seront connectés par un arbre binaire. Chaque nœud a un poids qui est la fréquence d'occurrence du symbole, ensuite l'arbre est créé par les étapes suivantes :

i) Les deux nœuds de poids le plus faible sont réunis pour donner un nœud parent qui a un poids égal à la somme des poids des deux nœuds fils.

ii) Le nœud parent est ajouté à la liste des nœuds, et les nœuds fils sont enlevés.

iii) Un des deux nœuds fils est désigné comme le chemin pris à partir du nœud parent pour décoder un bit 0, l'autre nœud étant pris pour décoder un bit 1.

iv) Les étapes précédentes sont répétées tant qu'il reste plus d'un nœud libre.

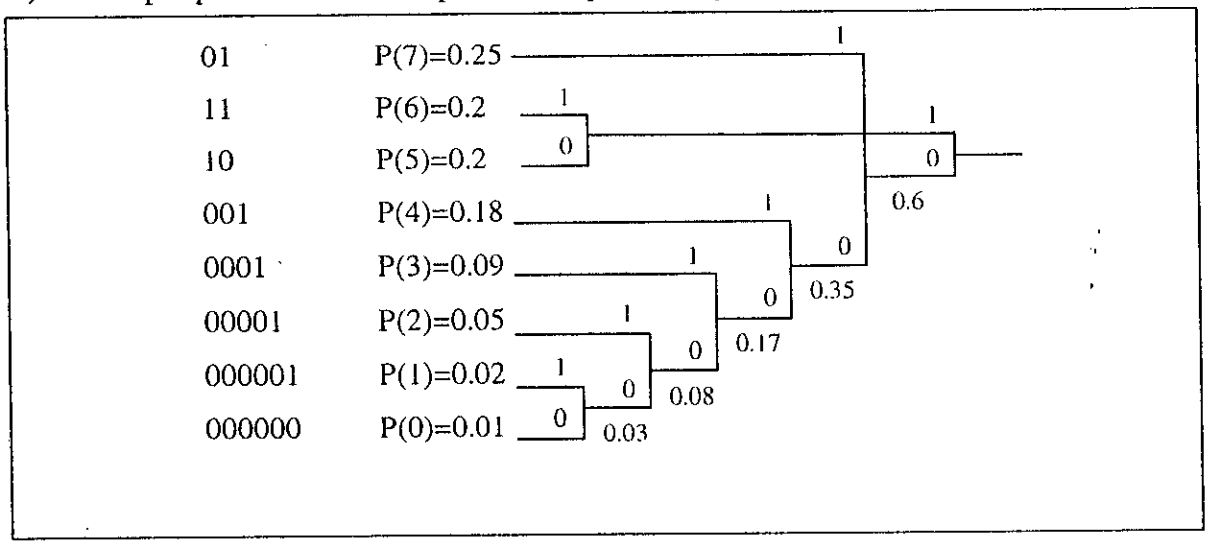


Figure II.2 *Codage de Huffman*

II.6.1.3 Codage par plage

Appelé le *Run Length Coding* (ou *Encoding*) RLC (ou RLE) [30], son principe est de regrouper les points voisins qui ont le même niveau, un groupement est défini par une paire de nombre (plage ,niveau), où plage est le nombre de points ayant le même niveau. Cet algorithme est d'autant performant que les plages des points sont étendues. L'application la plus profitante est le codage des images binaires.

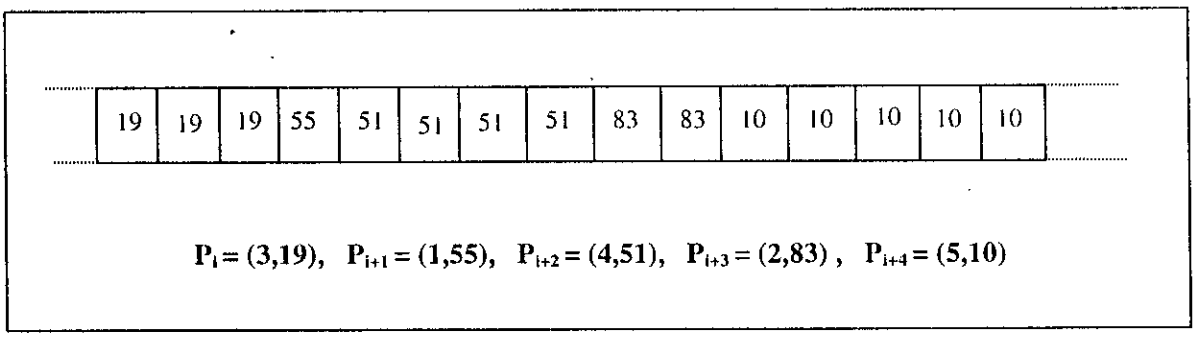


Figure II.3 *Codage RLE d'une ligne d'une image*

La figure II.3 montre comment les paires sont construites à partir d'un exemple de niveaux de gris prélevés d'une image.

II.6.1.4 L'algorithme LZW *(Lempel-Ziv-Welch)*

Cet algorithme est très utilisé dans la compression des fichiers informatiques [24][31], et surtout les fichiers d'images (formats GIF, TIF). Il est plus efficace qu'un simple RLC, il concurrence même l'algorithme de Huffmann et d'autre d'autant que la décompression est moins pénalisante. Le principe de cet algorithme est de coder des suites de mots ou chaînes de caractères par un seul numéro(code) qui sera transmit ou stocké. LZW est un algorithme monodimensionnel pour cela on traite l'image suivant le balayage télévision.

II.6.1.5 Le codage arithmétique

Ce codage consiste à remplacer un ensemble de symboles par un seul nombre en virgule flottante [31], ce concept est connu depuis longtemps mais ce n'est que récemment qu'on a pu l'implémenter sur des machines. La sortie d'un codeur arithmétique est un nombre < 1 et ≥ 0, ce nombre peut être décodé d'une manière unique pour créer le flot exact de symboles qui a conduit à sa génération.

II.6.2 Techniques irréversibles

Le deuxième type de méthodes de codage concerne les méthodes irréversibles qui ne permettent pas de retrouver l'image originale mais une image de qualité acceptable avec un taux de compression très intéressant. Ce sont les techniques les plus implémentées dans les systèmes pratiques, parmi ces méthodes citons la prédiction linéaire, la transformation, le codage par sous-bandes, la quantification vectorielle, les ondelettes, fractale et d'autres.

II.6.2.1 La prédiction linéaire

La technique de la PL est très utilisée dans la modélisation, le codage et la reconnaissance de la parole, où elle a donné de très bons résultats. Appliquée dans l'imagerie [4][5][12], la PL a donné des résultats très appréciables et comparables à ceux obtenus dans la parole. La PL se distingue par sa simplicité d'implémentation dans les applications temps réel, mettant en œuvre des images animées (TV) avec un faible coût. Pour ces raisons on la trouve dans plusieurs CODECS où elle a été adoptée par les comités de standardisation [32].

Principe

La prédiction linéaire exploite la forte corrélation entre les niveaux des points d'une même image. C'est à dire que le niveau d'un point est très lié aux niveaux de ses voisins. Ceci est mesuré objectivement par la fonction d'autocorrélation d'un signal image. Donc on calcule une valeur approchée du niveau d'un point dit valeur prédite, puis on calcule la différence entre la vraie valeur et la valeur prédite, si on arrive à avoir une valeur prédite très proche de la vraie valeur, alors l'erreur sera très faible, ce qui nécessite un nombre de bits très faible pour qu'elle soit codée.

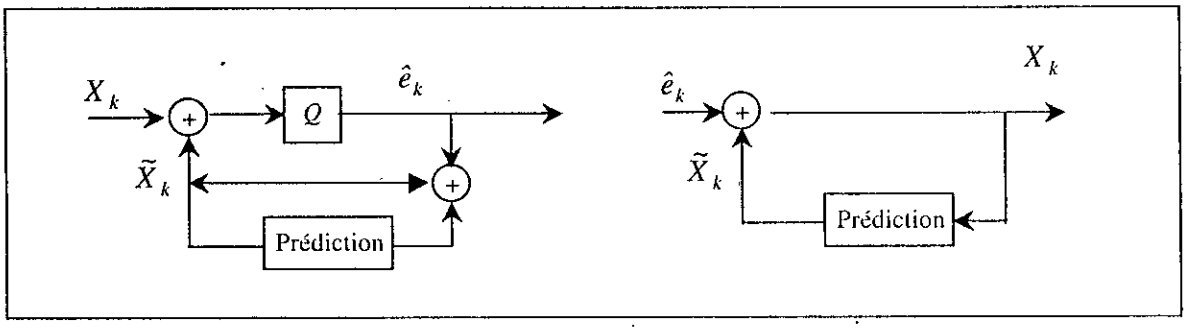


Figure II.4 *Les Blocs de Prédictions Cod/Dec*

La méthode la plus utilisée pour calculer la valeur prédite $\hat{x}$ est celle qui utilise une combinaison des points voisins du point à traiter [42]. Comme la plupart des applications utilisent le balayage vidéo (*rasetr scan*) donc les points qui peuvent être utilisés pour prédire un point sont ceux à gauche et au dessus de ce point, figure II.5, c'est à dire des points qui étaient déjà traités. Soit à prédire *x(i,j)*:

$$\hat{x}(i,j) = a_1 * x(i,j-1) + a_2 * x(i-1,j-1) + a_3 * x(i-1,j) + a_4 * x(i-1,j+1) \qquad (2.11)$$

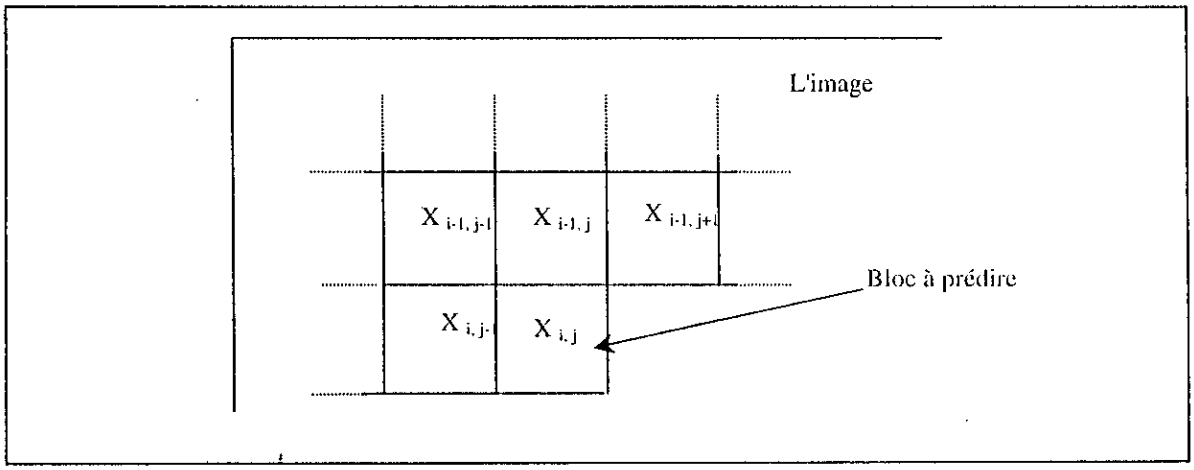


Figure II.5 *Les blocs de l'image utilisés pour la prédiction*

où les a_i sont dits *coefficients de prédiction*. Les coefficients de prédiction doivent être disponibles au codeur et au décodeur, il y a plusieurs algorithmes pour calculer ces coefficients, le plus connu est celui de Levinson-Durbin.

II.6.2.1.1 Calcul des coefficients de prédiction

L'idée de base est de déterminer les coefficients de prédiction qui minimisent l'erreur entre la vraie valeur et la valeur prédite. Pour une image complète, on minimise la moyenne de l'erreur quadratique. Cette erreur est donnée par :

$$e = x(i,j) - \hat{x}(i,j) \qquad (2.12)$$

on minimise l'espérance de e^2 :

$$E(e^2) = E[(x(i,j) - \hat{x}(i,j))^2] \qquad (2.13)$$

on a :

$$\hat{x}(i,j) = \sum_k \sum_l a_{kl}\, x(i-k, j-l) \qquad (2.14)$$

donc:

$$E(e^2) = E[(x(i,j) - \sum_k \sum_l a_{kl}\, x(i-k, j-l))^2] \qquad (2.15)$$

cette fonction à plusieurs variables, a_{kl}, passe par un minimum(démontré dans la littérature), ce minimum est atteint pour les valeurs de a_{kl} qui annulent les dérivées $\partial E(e^2)/\partial a_{kl}$

$$\partial E(e^2)/\partial a_{k'l'} = E\{[2*(x(i,j)-\sum_k \sum_l a_{kl}x(i-k,j-l)]*[-x(i-k',j-l')]\} \quad (2.16)$$

en annulant cette dérivée on obtient :

$$E(x(i,j)*x(i-k,j-l))=\sum_k \sum_l a_k * E(x(i-k,j-l)*x(i-k',j-l')) \quad (2.17)$$

$$R(k,l)=\sum_{k'} \sum_{l'} a_{kl} * R(k-k',l-l') \quad (2.18)$$

avec $R(k,l)$ la fonction d'autocorrélation.

Comme le signal image n'a pas des propriétés statistiques modélisables (non stationnaire non ergodique), donc les coefficients calculés dépendent grandement de l'image à partir de laquelle ils ont été calculés.

Le bruit dans la prédiction linéaire : pour pouvoir retrouver les valeurs de $x(i,j)$, il faut que $\hat{x}(i,j)$ du codeur soit identique à celui du décodeur, mais en pratique cela n'est pas possible.

Un codeur se basant sur le principe de la PL est soumis à plusieurs sources de bruit, tel que le bruit du au canal, où l'erreur $\hat{e}_k$ transmise sera infectée par un bruit du canal et par la suite X_k reconstruit au décodeur sera différente de celui du codeur. Il y a le bruit de quantification de X_k et de e_k, et enfin le bruit de différence de précision entre le codeur et le décodeur.

II.6.2.2 Les transformées

La transformation d'un signal permet de passer d'un espace donné à un autre espace pour effectuer certaines opérations. Les transformées nous permettent de passer du domaine temporel ou spatial, au domaine spectral ou transformé [45][49].

Principe

La forte corrélation entre les points ou les blocs d'une image, conduit à des composantes spectrales à fréquence élevée très faibles, donc on peut les coder avec un nombre de bits faible, on peut même les négliger carrément. Cette opération tend à décorreler les coefficients de l'image en diminuant la redondance, ce qui permet un codage efficace, mais tout de même les données restent fragiles envers les erreurs.

Une transformée est la multiplication successive des éléments des données par les éléments des vecteurs de base de la transformée; pour les images les données sont segmentées en blocs de forme et de taille définie. Une transformée bidimensionnelle est représentée comme suit :

$$X(k,l)=\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}x(i,j)A(k,l,i,j) \tag{2.19}$$

$A(k,l,i,j)$ est dit *noyau* de la transformée (*Kernel*). Pour calculer $x(i,j)$ on applique la transformée inverse sur $X(k,l)$ comme suit :

$$x(i,j)=\sum_{k=0}^{N-1}\sum_{l=0}^{M-1}X(k,l)B(k,l,i,j) \tag{2.20}$$

Une transformée est séparable si :

$$A(k,l,i,j)=A_v(k,i)A_h(l,j) \tag{2.21}$$

cette propriété, très intéressante, permet de calculer une transformée bidimensionnelle en calculant deux transformées monodimensionnelles, donc :

$$X(k,l)=\sum_{i=0}^{N-1}A_v(k,i)\sum_{j=0}^{M-1}A_h(l,j)x(i,j) \tag{2.22}$$

$$\Rightarrow \quad X=A_v x A_h^T \tag{2.23}$$

pour un noyau symétrique on aura :

$X=AxA^*$ et $x=A^*XA$

Intérêt: En observant x et X on peut retenir que :

- Les valeurs des éléments du bloc X sont faibles, et aussi on trouve des éléments qui sont favorisés par rapport à d'autres.
- Et aussi, ils sont signés (+ ou -), donc X n'a pas un sens visuel.

D'après la première observation on peut dire que la compression est effectuée en négligeant les éléments transformés très faibles, les négliger veut dire ne pas les transmettre ou les stocker.

Il y a quelques transformées dites singulières, elles ne possèdent pas une transformée inverse. Parmi les transformées inversibles il y a celles qui vérifient :

$$A^{-1}=(A^*)^T$$

elles sont dites transformées orthogonales. Leurs avantages est la simplicité de calculer la transformée inverse à partir de la transformée directe et cela par un simple changement de signes et un changement d'ordre des éléments de A.

II.6.2.2.1 Transformations particulières

Il y a toujours un compromis à faire entre l'efficacité de la transformation et sa simplicité d'implémentation, donc on cherche à trouver une transformée optimale qui permet de décorreler complètement les données et ramener le maximum d 'énergie aux coefficients de faible fréquence.

1)La KLT

La transformée de Karlhunen-Loev est une transformée optimale parce qu'elle répond aux critères d'une transformée optimale, mais sa mise en œuvre en pratique pose de sérieux problèmes [49]. Le calcul des coefficients transformés par la KLT nécessite le calcul de la matrice corrélation, comme le signal image n'est pas stationnaire cette fonction doit être calculée à chaque fois. La matrice de la transformée peut être singulière donc il sera difficile de calculer les vecteurs propres de C_x la matrice de covariance de l'image, et même le calcul des valeurs propres λ_i est très coûteux, et comme sa matrice est non séparable il n'y a pas un algorithme rapide pour la calculer. A cause de ces inconvénients la KLT n'est pas utilisée dans les systèmes pratiques, car son implémentation pratique est très coûteuse.

2)La TFD

Elle permet de compacter grandement l'énergie moyenne sur un petit nombre de composantes, sans atteindre bien entendu les performances de la KLT [24]. Le noyau de la TFD bidimensionnelle est donné par :

$$a(u,v,m,n)=\frac{1}{\sqrt{MN}}exp\left(-2j\pi\left(\frac{um}{M}+\frac{vn}{N}\right)\right) \qquad (2.24)$$

La matrice de la transformée TFD est séparable et unitaire donc inversible, il suffit d'inverser le signe de l'exposant. Sans oublier que les coefficients transformés sont complexes.

La compression par la TFD: Les coefficients transformés sont issus d'une image périodisée, par simple recopie, ainsi cette image sera discontinue, ce qui entraîne l'accroissement des modules des fréquences élevées. Si on prend un bloc d'une image, son bloc transformé correspondant est caractérisé par : une symétrie autour du coefficient central, une décroissance monotone du module vers le centre du bloc. D'après ces constatations, les coefficients transformés à abandonner seront ceux du centre du bloc.

Ici se pose le problème de contradiction entre le critère objectif (*MSE*) et le critère subjectif (*HVS*)(*Human Visual System*), tel que pour satisfaire le critère MSE il faut supprimer des coefficients auxquels le HVS accorde une importance, pour cela on préfère la DCT.

3)La DCT

La DCT, *Discret Cosine Transform*, est une transformée issue de la TFD [41], depuis son introduction par Ahmed et al en 1974 [25], elle a trouvé de larges applications dans l'imagerie, elle représente le noyau du standard JPEG I. Elle permet d'obtenir des coefficients plus faibles que ceux obtenus par la TFD, réels, et qui sont localisés dans une zone d'acuité visuelle minimale. La transformée 2D-DCT est donnée par :

$$X(u,v)=\frac{2}{N}C(u)C(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}x(i,j)\cos\left(\frac{\pi}{N}u\left(i+\frac{1}{2}\right)\right)\cos\left(\frac{\pi}{N}v\left(j+\frac{1}{2}\right)\right) \qquad (2.26)$$

avec,

$$C(k)=\begin{cases}\frac{1}{\sqrt{2}} & si\ \ k=0\\ 1 & si\ \ k\neq 0\end{cases}$$

La transformation inverse de la DCT, i.e. la IDCT est donnée par :

$$x(i,j)=\frac{2}{N}\sum_{u=0}^{N}\sum_{v=0}^{N}C(u)C(v)\left(X(u,v)\cos\left(\frac{\pi}{N}u\left(i+\frac{1}{2}\right)\cos\left(\frac{\pi}{N}v\left(j+\frac{1}{2}\right)\right)\right)\right) \qquad (2.27)$$

Les performances de la DCT sont supérieures à celles de la TFD. De toutes les transformées rapides, la DCT possède les performances les plus proches de celles de la KLT. D'après la formule de calcul des coefficients de la DCT on voit bien que le temps requis pour calculer un coefficient est proportionnel à N^2, donc lorsque N augmente ce temps augmente d'une façon dramatique. Pour cela la DCT n'est jamais appliquée sur l'image entière, pour cela on procède à découper l'image en blocs plus petits (par exemple la norme JPEG 8x8).

Il y a plusieurs algorithmes rapides de calcul de la DCT [46], qui ont été mise au point, ces algorithmes sont classés suivant leurs méthodes d'approche comme suit :

- Calcul indirect où la FFT est utilisée [25]
- Factorisation directe, ce type d'algorithme très difficile à implémenter
- Calcul récursif, ce sont les algorithmes les plus recommandés à cause de leurs simplicité d'implémentation.

II.6.2.2.2 Quantification des coefficients

Après transformation, les coefficients transformés seront répartie sur un intervalle plus important, occupant ainsi un nombre de bits plus élevé, ce qui donne l'impression qu'au lieu d'appliquer une compression on a appliqué une expansion. Mais le problème est résolu par la quantification des coefficients. Cela est effectué en divisant les coefficients transformés par des valeurs dites *quantum* .

A chaque coefficient de la matrice transformée correspond une valeur dans la matrice de quantification, ainsi on aura les coefficients quantifiés donnés par la formule :

$$Coefficient\ Quantifié\ (i,\ j) = \frac{Coefficient\ Tranformés\ (i,\ j)}{Quantum(\ i,\ j\ )} \qquad (2.28)$$

au décodage l'opération inverse est appliquée.

II.6.2.3 Technique de codage par sous-bandes [8] [48]

Le codage d'image par sous-bandes (SBIC); est devenue très intéressant surtout pour le codage d'image de haute qualité, parce que l'image décodée avec cette technique ne souffre pas du problème de discontinuité de blocs.

Dans le codage par sous-bandes , un ensemble de filtres est utilisés pour décomposer le signal d'entrée en plusieurs signaux à bande étroite, chaque signal représente une partie du spectre du signal original. Ces filtres sont dits filtres d'analyse.. Comme chaque signal a une bande très réduite, un sous-échantillonnage est appliqué sur chaque signal, ce qui permet une réduction considérable dans le débit. A la réception une interpolation est appliquée sur l'ensemble des signaux , pour régénérer le signal original. Cette opération est effectuée par un ensemble de filtres dits filtres de synthèses figure II.6.

Les filtres utilisés dans cette technique sont les filtres à miroir quadratique, QMF [48], développés par J.B.Johnston, à cause de leur propriétés de constitution des signaux parfaite, et caractéristiques aux basses fréquences.

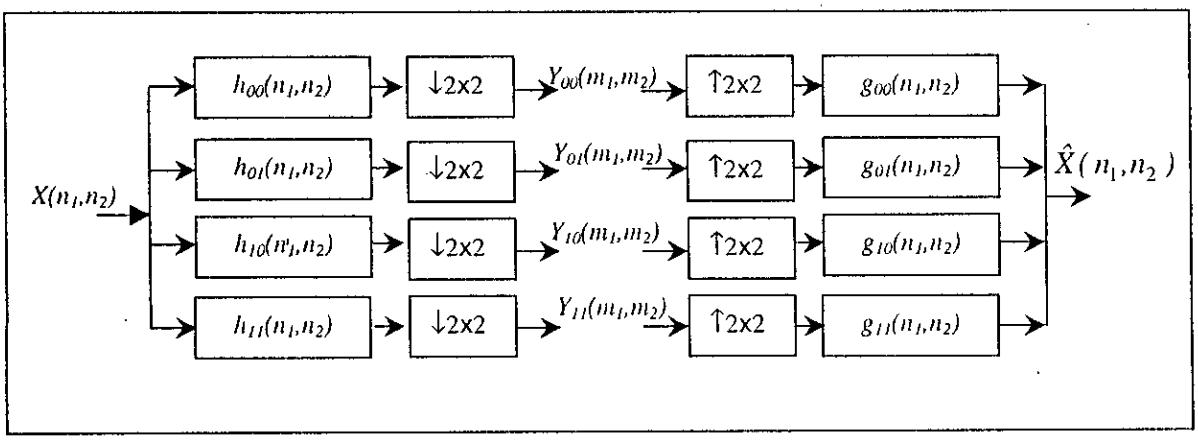


Figure II.6 *Codeur/Décodeur Sous-bandes*

II.6.2.4 Quantification vectorielle

La VQ est une généralisation de la quantification scalaire, en effet plutôt que de quantifier les niveaux de gris des points les uns après les autres, la VQ quantifie un groupe de points ou un bloc à la fois [1] [5] [34]. Ces blocs sont obtenus par un découpage fictif de l'image, ensuite la VQ consiste à remplacer chaque bloc par un autre bloc d'une liste préétablie dite *codebook* ou dictionnaire. Le choix du bloc remplaçant s'effectue en comparant le bloc original à tous les blocs du codebook et en retenant le bloc qui lui est le plus ressemblant ou proche au sens d'une certaine distance (MSE ou MAE). Ensuite on transmet l'indice du vecteur choisi, au décodeur, on adresse le codebook avec l'indice pour retrouver le vecteur-code pour reconstituer l'image.

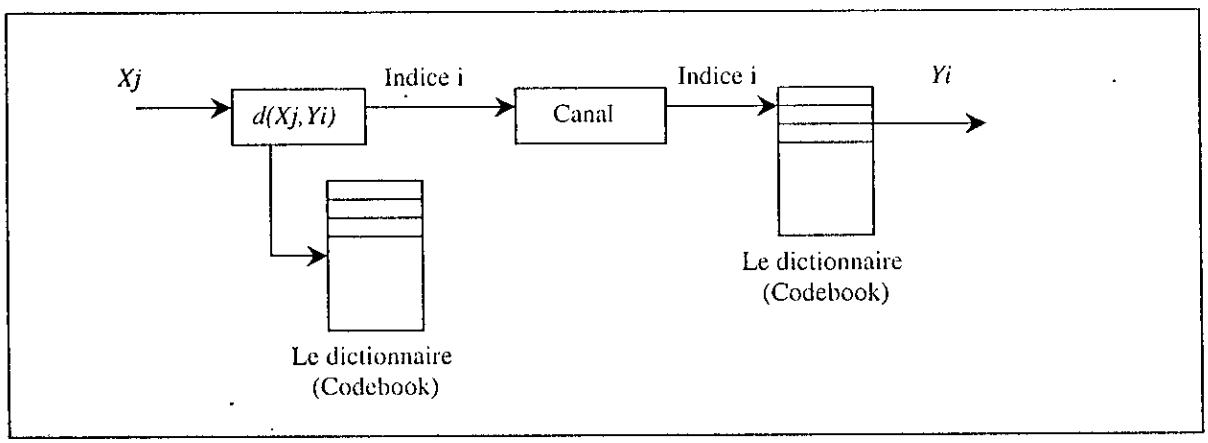


Figure 7. *Quantificateur vectoriel*

Le codeur et le décodeur doivent disposer du même codebook et le même critère de ressemblance (distance). Il y a plusieurs algorithmes pour concevoir un codebook, parmi eux

on cite l'algorithme LBG (*Linde-Buzo-Gray* [14]), et l'algorithme PNN (*Pairewise Nearest Neihbghour* [13][15]), tous ces algorithmes se basent sur un ensemble d'apprentissage très large pour extraire un nombre très limité de vecteurs-codes qui constitueront le codebook.

La VQ permet d'avoir des taux de compression très intéressants, tel que si N est la taille du codebook, la longueur des indices est $\log_2(N)$, et k la dimension des vecteurs ou blocs, alors le débit *r* est donné par :

$$r = \frac{1}{k} log_2(N) \quad \text{bpp} \tag{2.29}$$

et si *n* représente le nombre de bits par pixel de l'image originale alors le taux de compression sera:

$$\sigma = \frac{n}{r} \tag{2.30}$$

II.7 Conclusion

Les techniques de compression sans distorsion sont utilisées surtout avec des données délicates comme les fichiers informatiques, les données des banques, etc. Elles permettent de restaurer des images fidèlement avec un taux de compression moyen.

Les techniques de compression avec distorsion permettent d'atteindre un taux de compression très élevé par rapport à celui généré par les techniques sans distorsion, et cela au prix d'une réduction de la qualité de l'image décodée. Mais souvent cette dégradation de qualité n'est pas tellement perceptible au point où parfois on n'arrive pas à distinguer l'image originale de l'image décodée.

Ces techniques avec distorsion sont combinées pour avoir un codeur hybride qui a les avantages des techniques combinées. Pour avoir des performances plus élevées, souvent les techniques avec distorsion sont utilisées conjointement avec des techniques entropiques.

Chapitre III

La Quantification Vectorielle

III.1 Introduction

La quantification vectorielle est une généralisation de la quantification scalaire, cette généralisation qui est le passage d'un espace monodimensionnel à un espace multidimensionnel a ouvert plusieurs horizons de travail et de recherche. La quantification scalaire est utilisée principalement dans les conversions Analogique – Digital, par contre la quantification vectorielle utilise des données numériques. Dans la plupart des cas, l'entrée est une représentation digitale d'une information, et la sortie sera la version compressée de cette entrée [1].

La quantification vectorielle manipule des vecteurs formés par regroupement des échantillons du signal en forme d'onde (parole ou image), par la suite la quantification vectorielle peut être considérée comme une reconnaissance de forme, parce qu'elle procède en comparant le vecteur d'entrée avec les vecteurs de références et cela en calculant des distances.

Les problèmes rencontrés avec la quantification vectorielle sont :

- L ' énorme quantité de calculs à effectuer pour la recherche du plus proche vecteur parmi les vecteurs de références.
- L'effet de bloc visible dans les images décodées. Ainsi de nombreux travaux ont été effectuées dans ces deux sens.

III.2 La quantification scalaire

La quantification est l'opération qui permet de passer d'un espace infini à un espace fini [29], elle consiste à remplacer une valeur exacte par une valeur choisie à partir d'un ensemble prédéterminé suivant un critère de rapprochement. La quantification scalaire est à la base des convertisseurs AN, tel que l'entrée peut prendre sa valeur dans un intervalle continu, et la sortie est numérique ou digitale.

III.2.1 Définitions

D'une manière plus précise on définie un quantificateur scalaire à N points par l'application Q de $\Re$ vers C, comme suit :

$$Q : \Re \longrightarrow C$$

avec $\Re$ l'ensemble des nombres réels, et :

$$C = \{ y_1, y_2, ..., y_N \} \subset \Re$$

l'ensemble de sortie ou *codebook (dictionnaire)*,

$|C| = N$ (cardinal de C) y_i est appelé souvent niveau de sortie ou valeur de reproduction, ils sont pris en général :

$$y_1 < y_2 < ..... < y_N$$

On définie la résolution ou le débit du code d'un quantificateur scalaire comme :

$$r = log_2(N) \tag{3.1}$$

r donne le nombre de bits nécessaire pour représenter une valeur quantifiée.

Un quantificateur scalaire à N niveaux est une partition de l'ensemble $\Re$ en N *régions* ou *atomes* R_i, tel que :

$R_i = \{ x \in \Re / Q(x) = y_i \} = Q^{-1}(y_i)$,

on conclue que :

$\bigcup_i R_i = \Re$ et $R_i \cap R_j = \phi \quad \forall i \neq j$.

Une région non limitée est dite surchargée, et toute région limitée est dite granulaire.

Un quantificateur scalaire est dit régulier si chaque région R_i est un intervalle, et si $y_i \in (x_{i-1}, x_i)$ où x_i est appelé point limite ou point de décision, tel que :

si : $Q(a)=w$ et $Q(b)=w$ et si $a<c<b$ alors $Q(c)=w$

Un quantificateur scalaire est dit symétrique si :

$$Q(x)=-Q(-x).$$

Un graphe d'une fonction $Q(x)$ symétrique d'un quantificateur scalaire régulier est donné par la figure suivante.

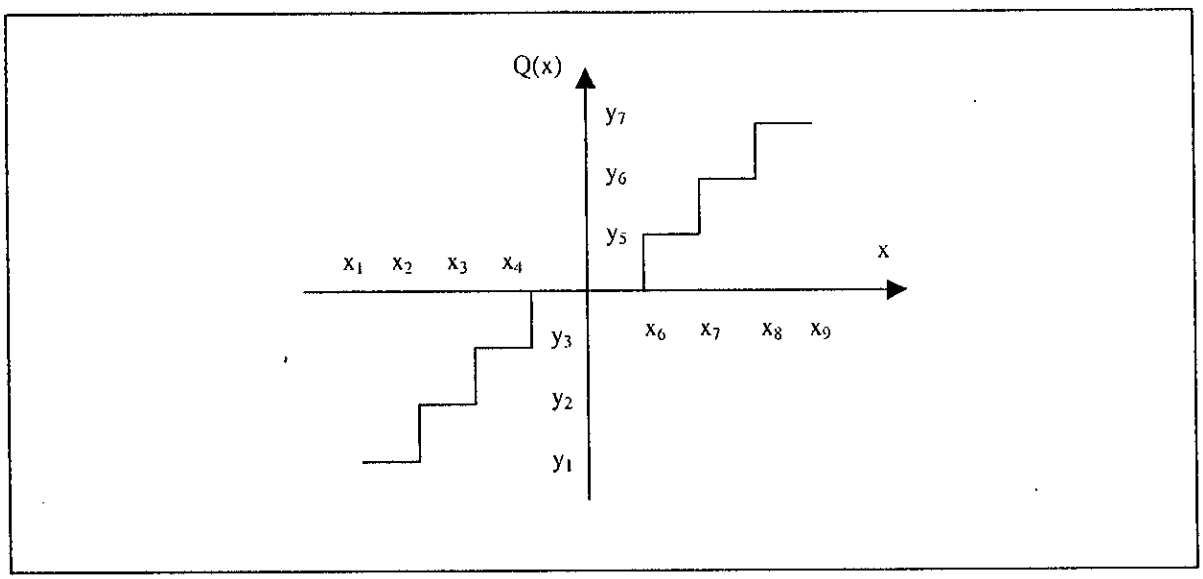


Figure III.1 *La fonction escalier*

Un quantificateur scalaire est vu comme une combinaison de deux opération : l'opération de codage et l'opération de décodage :

Le codage : $E:\Re \longrightarrow I$ avec $I=\{1,2,...,N\}$

Le décodage : $D:I \longrightarrow C$

on peut écrire : $Q(x)=y_i$ et $E(x)=i$ et $D(i)=y_i$

d'où : $Q(x)=D(E(x))$

III.2.2 Performances

L'évaluation des performances d'un quantificateur s'effectue en calculant la dégradation en qualité apportée par le fait de remplacer x par $Q(x)$ [24]. L'erreur $Q(x)-x$ est dite distorsion, elle est notée $d(Q(x),x)$, l'erreur la plus utilisée est l'erreur MSE :

$$d(Q(x),x)=d(y_i,x)=|x-y_i|^2 \tag{3.2}$$

L'erreur moyenne statistique :

$$D=E(d(y_i,x))=\int_{-\infty}^{+\infty} d(y_i,x)f_X(x)dx \tag{3.3}$$

où : $f_X(x)$ est la fonction de densité de probabilité

Un codeur est dit optimal si D est minimale.

III.2.3 Allocation de bits

En général un système contient plusieurs quantificateurs, chacun se charge de la quantification d'un paramètre du signal. Chaque quantificateur diffère de l'autre suivant le paramètre à quantifier et la précision qu'il nécessite pour transmettre le signal de la façon la plus fidèle que possible. Le nombre total de bits pour représenter tout le système est en général limité. Donc un concepteur d'un système de codage doit prendre en compte l'allocation de bits, qui est la distribution des bits sur les différents quantificateurs du système.

III.3 La quantification vectorielle VQ

La quantification vectorielle est une généralisation de la quantification scalaire, tel que la VQ manipule des blocs ou des vecteurs de données. La SQ est utilisée principalement pour faire la conversion AN, par contre la QV utilise des données déjà numérisées pour un but d'avoir une version compressée du signal. Les vecteurs sont formés à partir des échantillons du signal, dans la parole un vecteur est un ensemble d'échantillons successifs appelés segments, dans l'image un vecteur est constitué d'un bloc rectangulaire et souvent carré.

III.3.1 Définitions [1]

Un quantificateur vectoriel de dimension k, où k est la dimension du vecteur, et de taille N est une application Q de l'espace $\Re^k$ dans un ensemble C ($C \subset \Re^k$) fini ayant N vecteurs dits *vecteurs code* ou *mots code*.

Donc :

$$Q : \Re^k \longrightarrow C$$

avec ;

$C=\{y_1, y_2, ..., y_N\}$;

$y_i \in \Re^k$ pour tout $i \in I=\{1,2,....,N\}$.

L'ensemble C est appelé *dictionnaire* ou *codebook*. Le débit du quantificateur est :

$$r = \frac{1}{k} log_2(N) \tag{3.4}$$

r représente le nombre de bits par échantillons .

Un quantificateur vectoriel à N points effectue une partition dans l'espace $\Re^k$ en N *régions* ou *cellules* R_i, $i \in I$, figure III.2, et avec :

$$R_i = \{X \in \Re^k : Q(X) = Y_i\},$$

R_i est appelée parfois image inverse de Y_i :

$$R_i = Q^{-1}(Y_i)$$

Les régions R_i vérifient :

$$\bigcup_i R_i = \Re^k \text{ et } R_i \cap R_j = \phi \quad \forall i \neq j$$

Une propriété importante est la convexité, un sous-ensemble $S \in \Re^k$ est dit convexe si : $a, b \in S \Rightarrow \alpha a + (1-\alpha) b \in S \quad \forall 0 < \alpha < 1$, d'où on dit qu'un quantificateur vectoriel est régulier si toute région R_i est convexe et $\forall i \in I$ alors $Y_i \in R_i$.

Les R_i sont dites régions de *Voronoï*, elles sont convexes et ont des frontières particulières satisfaisants les critères de Direcklet.

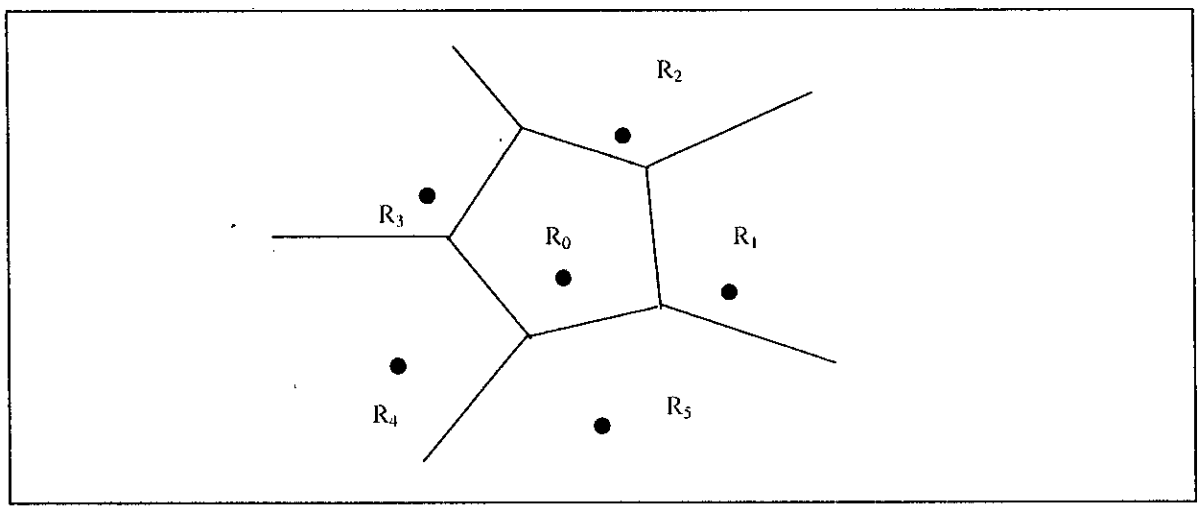


Figure III.2 *Régions de Voronoï*

Un quantificateur vectoriel peut être décomposé en deux opérations : le codage et le décodage, figure III.3. Le codage est une application E de $\Re^k$ dans l'ensemble des indices I, le décodage est une application D de I dans C :

$E : \Re^k \longrightarrow I$

$D : I \longrightarrow C$

$Q(X) = D(E(X))$

Un quantificateur vectoriel génère l'indice et le vecteur code correspondant Y_i (voir Figure)

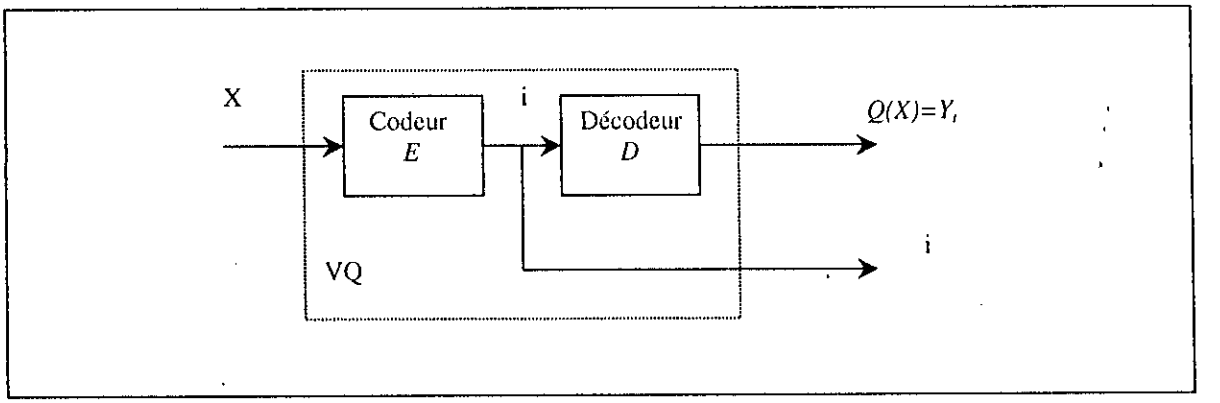


Figure III.3 *Quantificateur Vectoriel*

III.3.2 Performances d'un quantificateur vectoriel

La détermination de la performance d'un quantificateur vectoriel revient à calculer la distorsion $d(X,Q(X))$ qui est un coût non négatif de la quantification du vecteur X. Donc la performance du système est déterminée par le calcul de l'erreur moyenne D :

$$D = E(d(X, Q(X))) \tag{3.5}$$

Une mesure de distance doit être la plus significative et la plus simple à évaluer pour avoir des systèmes travaillant en temps réel. La mesure de distorsion la plus utilisée est l'erreur quadratique ou la distance Euclidienne ; tel que la distance entre un vecteur X et sa reproduction $Q(X)$ soit donnée par :

$$d(X, Q(X)) = \|X - Q(X)\|^2 = (X - Q(X))^T (X - Q(X)) = \sum_{i=1}^{k} |X_i - Y_i|^2 \tag{3.6}$$

d'où :

$$D = E(\|X - Q(X)\|^2) \tag{3.7}$$

Une autre distorsion utilisée dans l'évaluation de la performance d'un quantificateur vectoriel est l'erreur quadratique pondérée (*WSE : Weighted Squared Error*) :

$$d(X, Q(X)) = (X - Y)^T W (X - Y) \tag{3.8}$$

où W est la matrice des poids, elle est symétrique et positive définie. On remarque que si : $W=I$ on tombe sur l'erreur quadratique. Si W est une matrice diagonale avec :
W = diag (w_{ij}), alors :

$$d(X,Y)=\sum_{i=1}^{k} w_{ij}(x_i - y_i)^2 \qquad (3.9)$$

suivant w_{ij} cette erreur permet de favoriser quelques composantes par rapport à d'autres.
Par ailleurs si l'on connaît la distance d(x, y), et N le nombre de vecteur et la distribution conjointe du vecteur d'entrée X $f(X)$ on peut reconnaître le quantificateur optimal. C'est de tous les quantificateurs celui qui minimise l'espérance mathématique de la distance :

$$E(d)=\int d(X,Q(X))f(X)dX \qquad (3.10)$$

III.3.3 Quantificateur de Voronoï

Une classe importante de quantificateur vectoriel dite quantificateur de Voronoï ou quantificateur du plus proche voisin(*Nearest Neighbor quantizer*) elle est caractérisée par le fait que la partition de l'espace est complètement définie par le dictionnaire et la distance utilisée [1]. L'avantage de ce type de quantificateur est que le processus de codage ne nécessite pas le stockage de la géométrie des régions. Un quantificateur de Voronoï est défini comme un quantificateur qui a des régions respectant :

$$R_i = \{X, d(X,Y_i) \le d(X,Y_j), j \in I\}$$

Un quantificateur de Voronoï (ou NNQ) suit l'algorithme suivant pour coder les vecteurs d'entrées :

1) $d = d_0$; $j=1$; $i=1$
2) Calcul $D_j = d(X,Y_i)$
3) Si $D_j < d$ $D_j \rightarrow d$, $j \rightarrow i$
4) Si $j<N$ $j+1 \rightarrow j$. *Aller à 2)*
5) Stop . Résultat dans i

La valeur de d_0 doit être la plus grande possible. Après l'exécution, l'algorithme détermine le vecteur de sortie Y_i, et d la distance entre X et Y_i.

III.3.4 La conception d'un quantificateur vectoriel

La phase clé dans la conception d'un VQ est sans doute la détermination du codebook. Les algorithmes qui permettent de définir un codebook sont en majorité itératifs, i.e. ayant à une itération *i* un codebook et une partition optimale, on calcule un nouveau codebook qui est optimal à cette partition, et ainsi de suite. Le problème de ces algorithmes est bien la détermination du codebook initial, parce que c'est ce codebook qui détermine la façon de convergence de l'algorithme et la qualité du codebook final.

III.3.4.1 L'algorithme LBG

C'est l'algorithme le plus utilisé dans le design des codebook, il est basé sur l'approche de Lloyd. Le LBG a été mis au point par Y.Linde, A.Buzo et R.M.Gray et publié en 1980 [14] . Les propriétés de l'algorithme ont été prouvées par des exemples de sources Gaussienne et sur des signaux de la parole. L'algorithme consiste à concevoir un quantificateur à partir d'un ensemble d'apprentissage. L'algorithme est [5]:

0) *N la taille du codebook, M la taille de l'ensemble d'apprentissage*

$X = \{x_i, i = 0, N-1\}$ *l'ensemble d'apprentissage,*

$C_0 = \{y_i, i = 0, N-1\}$ *le codebook initial,*

$P(C_m) = \{S_i, i = 0, N-1\}$ *une partition de l'ensemble d'apprentissage,*

et soit $\varepsilon \geq 0$ *un seuil de distorsion,* $D_{-1} = \infty$ *et* $m=0$

1) *Ayant* $C_m = \{y_i, i = 0, N-1\}$, *déterminons la partition* $P(C_m) = \{S_i, i = 0, N-1\}$ *qui vérifie :* $x_j \in S_i$ *si* $d(x_j, y_i) \leq d(x_j, y_l)$ *pour tout l. Calculons l'erreur moyenne :*

$$D_m = D(C_m, P(C_m)) = \frac{1}{M} \sum_{j=0}^{M-1} \min_{y \in C_m} d(x_i, y)$$

2) Si $(D_{m-1} - D_m) / D_m \leq \varepsilon$ *alors* C_m *est le codebook final sinon on continue.*

3) Calcul du nouveau codebook par le calcul des centroïdes des partitions comme suit :

$$\hat{x}_i = \frac{1}{card(S_i)} \sum_{j: x_j \in S_i} x_j$$

$C_{m+1} = \{\hat{x}_i, i = 0, N-1\}$

m=m+1, aller à 1).

Il est prouvé que cet algorithme diminue ou laisse inchangée la distorsion *D*, donc on a toujours C_{m+1} meilleur que C_m. Une fois le codebook final déterminé, il est utilisé sur de nouvelles données autres que l'ensemble d'apprentissage par la règle du plus proche voisin (*nearest neighbor NN*).

Les deux problèmes rencontrés dans cet algorithme sont la détermination du dictionnaire initial C_0 et le problème du minimum local.

III.3.4.2 L'algorithme PNN (*Pairewise Nearest Neighbour*)

C'est un algorithme similaire à LBG, il permet la conception d'un codebook en se basant sur la fusion des groupes les plus proches au sens d'une certaine distance [13] .

Principe

Initialement l'algorithme démarre avec M groupes, où M est le nombre de vecteurs de l'ensemble d'apprentissage, tel que chaque groupe contient un seul vecteur. On cherche les deux vecteurs les plus proches l'un de l'autre, et on les fusionne dans le même groupe qui sera représenté par le centroïde des deux, et on aura par la suite M-1 groupes. Cette opération est répétée en utilisant les centroïdes des groupes résultants par fusion jusqu'à atteindre la taille du codebook voulu et l'erreur voulue.

Soient les notation suivantes :

C_i le i^{ème} groupe de vecteurs,

C_{ij} le groupe formé par la fusion du groupe i et le groupe j,

n_i le nombre de vecteur de C_i ,

n_{ij} le nombre de vecteurs de C_{ij},

$\bar{x}_i$ centroïde ou moyenne du groupe C_i ,

$\bar{x}_{ij}$ centroïde du groupe C_{ij},

S_i^2 MSE entre $\bar{x}_i$ et les vecteurs du groupe C_i ,

S_{ij}^2 MSE entre $\bar{x}_{ij}$ et les vecteurs du groupe C_i,

notons par $< x, y >$ le produit scalaire du vecteur x et y, donc on aura :

$$n_{ij} = n_i + n_j \tag{3.11}$$

$$\bar{x}_{ij} = \frac{n_i \bar{x}_i + n_j \bar{x}_j}{n_i + n_j} \tag{3.12}$$

$$S_{ij}^2 n_{ij} = \sum_{x \in C_{ij}} \left| x - \bar{x}_{ij} \right|^2 = \sum_{x \in C_i} \left| x - \bar{x}_{ij} \right|^2 + \sum_{x \in C_j} \left| x - \bar{x}_{ij} \right|^2 \tag{3.13}$$

$$\sum_{x \in C_i} \left| x - \bar{x}_{ij} \right|^2 = \sum_{x \in C_i} \left(|x|^2 - 2 < x, \bar{x}_{ij} > + \left| \bar{x}_{ij} \right|^2 \right)$$

$$= n_i (S_i^2 + |\bar{x}_i|^2) - 2 n_i < \bar{x}_i, \bar{x}_{ij} > + n_i \left| \bar{x}_{ij} \right|^2$$

$$= n_i S_i^2 + n_i \left| \bar{x}_i - \bar{x}_{ij} \right|^2$$

$$= n_i S_i^2 + n_i \left| \frac{n_i \bar{x}_i + n_j \bar{x}_i - n_i \bar{x}_i - n_j \bar{x}_j}{n_i + n_j} \right|^2$$

$$= n_i S_i^2 + n_i \left| \frac{n_j \bar{x}_i - n_j \bar{x}_j}{n_i + n_j} \right|$$

$$= n_i S_i^2 + \frac{n_i n_j^2}{(n_i + n_j)^2} \left| \bar{x}_i - \bar{x}_j \right|^2$$

d'où :

$$n_{ij} S_{ij}^2 = n_i S_i^2 + n_j S_j^2 + \frac{n_i n_j^2}{(n_i + n_j)^2} \left| \bar{x}_i - \bar{x}_j \right|^2 + \frac{n_i^2 n_j}{(n_i + n_j)^2} \left| \bar{x}_j - \bar{x}_i \right|^2$$

$$= n_i S_i^2 + n_j S_j^2 + \frac{n_i n_j (n_i + n_j)}{(n_i + n_j)^2} \left| \bar{x}_i - \bar{x}_j \right|^2 \qquad (3.14)$$

On déduit de la dernière expression l'erreur commise lors de la fusion de deux groupes C_i et C_j, donc il faut choisir deux groupes qui minimisent cette erreur.

III.3.4.3 Détermination du dictionnaire initial C_0 [1]

La convergence de l'algorithme de conception du codebook dépend de plusieurs paramètres, le plus important est le dictionnaire initial. Dans ce domaine on trouve plusieurs techniques dont on va citer quelques-unes.

1) Génération aléatoire de C_0

C'est la façon la plus simple, elle consiste à prendre des vecteurs de l'ensemble d'apprentissage d'une manière aléatoire suivant la loi de distribution de la source, elle est dite technique de Monte Carlo. On prend les N premiers vecteurs de la source, et si la corrélation est très élevée entre les vecteurs alors on prend un vecteur chaque K vecteurs. Une technique plus compliquée et intéressante consiste à générer un codebook initial suivant une distribution qui optimise la fonction distorsion débit RDF de Shannon.

2) Par élimination (*Pruning*)

Cette technique aussi utilise les vecteurs de la source pour générer C0 tel que : on prend le premier vecteur on le met dans le dictionnaire ensuite on calcule la distorsion entre ce vecteur et le deuxième vecteur de l'ensemble d'apprentissage, si cette distorsion est inférieure à un certain seuil alors on passe au vecteur suivant, sinon on ajoute ce vecteur au codebook, jusqu'à ce que le codebook atteint sa taille finale. Si tous les vecteurs de l'ensemble d'apprentissage sont épuisés et le codebook n'a pas atteint sa taille alors on réduit le seuil de distorsion et on refait l'opération.

3) Codes produit

Dans cette technique on utilise plusieurs quantificateurs scalaires pour avoir un quantificateur vectoriel et cela par un produit scalaire. Si on veut avoir un dictionnaire de taille N, et de dimension k, alors on effectue le produit scalaire de k quantificateurs scalaires avec chacun contenant 2^R mots code, tel que :

$Q(x_0, x_1, ..., x_{k-1}) = (q(x_0), q(x_1), ..., q(x_{k-1}))$

où R doit être entier.

4) Spliting

Introduite par Linde et al, cette technique permet d'avoir des codebook de plus en plus larges (à chaque itération sa taille double). Le codebook initial contient un seul vecteur qui est le centroïde de l'ensemble d'apprentissage. Ce vecteur code Y_0 est splité en deux, en ajoutant et soustrayant une valeur aléatoire $\varepsilon > 0$, ainsi de suite(voir l'algorithme). L'algorithme du spliting est donné par:

0) Initialisation: $M = 1,\ C_0(M) = \hat{x} = \frac{1}{L}\sum_{i=0}^{L-1} x_i$, *L la taille de l'ensemble d'apprentissage*

avec $\hat{x}$ *le barycentre de l'ensemble d'apprentissage.*

1) Ayant le codebook $C_0(M)$*, contenant M vecteurs* $\{y_i, i = 0,2,.....,M-1\}$ *à partir de chaque vecteur on crée deux autres sous-ensembles de vecteurs comme suit* $y_i + \varepsilon$ *et* $y_i - \varepsilon$ *où* ε *est un vecteur de perturbation, ainsi on obtient une collection de 2M vecteurs, on pose M=2M*

2) Si M=N, alors $C_0 = C_0(M)$ *est le codebook initial, fin*

sinon

3) On exécute l'algorithme LBG pour trouver un quantificateur optimal à M niveaux, et on revient à 1)

III.3.5 Comparaison entre la VQ et la SQ

L'avantage le plus intéressant de la VQ par rapport à la SQ est que rien ne limite son taux de compression. En effet, la SQ ne peut avoir un taux de compression supérieur à *n* dans la mesure où il n'est pas possible de coder binairement un niveau de gris avec moins de un bit. Bien évidemment, la VQ ne peut pas construire des mots binaires avec moins de un bit mais ces mots binaires remplacent tout un ensemble de niveaux de gris de l'image.

En effet si à partir du nombre de bits utilisés pour un bloc complet on calcule le nombre de bits utilisés pour un niveau de gris seulement en divisant la longueur du mot binaire par le nombre de points dans le bloc on aura *R/k* qui peut être inférieur à 1.

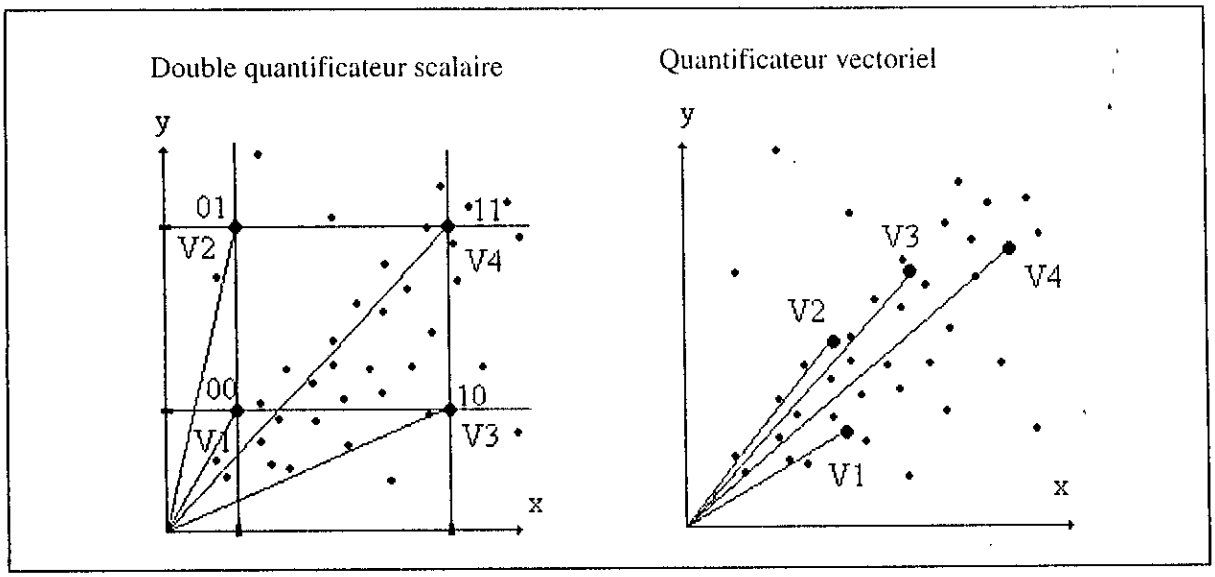


Figure III.4 *Comparaison entre la VQ et la SQ*

D'un autre côté, pour des taux de compression identiques, la VQ restitue des images avec une meilleure qualité que la SQ. Ceci est confirmé d'une manière générale par la théorie de l'information et vérifié en pratique par les travaux qui ont été effectués dans ce domaine.

La figure III.4 nous montre une comparaison entre la VQ et la SQ d'un nuage de points. Pour quantifier ces vecteurs on doit les remplacer par des vecteurs représentants en nombre plus réduit. Supposons qu'ils sont au nombre de quatre. Pour les déterminer deux méthodes sont utilisées :

1- Une double SQ opérée successivement sur les abscisses puis sur les ordonnées. Plus précisément on quantifie scalairement et successivement les projections des points sur l'axe des abscisses puis les projections de ces même points sur l'axe des ordonnées.
2- Une quantification vectorielle appliquée directement dans l'espace .

Dans les deux cas on obtient 4 représentants pouvant être numérotés binairement par 2 bits. D'après les figures on voit bien :

* Les vecteurs V_2 et V_3 de la SQ sont nettement écartés du centre du nuage de points. Ces deux représentants ne pourront pas remplacer plus de 2 ou 3 points . Tous les autres seront remplacés par V1 et V4, le choix et limité.
* Par contre pour la VQ les représentants sont à l'intérieur du nuage et peuvent remplacer autant de points. Plus rigoureusement les 4 représentants ont des efficacités équivalentes.

III.3.5.1 Taille et forme des blocs

On trouve dans [24] les résultats d'une étude sur les taille et les formes optimales. Pour des taux de compression supérieurs à quatre, les conclusions qui s'en dégagent sont : la forme

carrée est l'optimale, et pour un taux de compression donné il est préférable de choisir les plus grandes tailles possibles.

Le fait que la forme optimale soit la forme carrée était prévisible, parce que la quantification vectorielle tient compte des corrélations spatiales entre les niveaux de gris des points. Pour ce qui est de la taille des blocs, l'argumentation est plus délicate, parce qu'elle dépend du type des images utilisées et le taux de compression, autrement dit il n'est pas évident qu'il faille systématiquement prendre les plus grands blocs carrés possibles, car il y a toujours une limite pratique supérieure imposée.

Les dimensions des blocs sont toujours des puissances entières de 2, cela s'explique par le fait qu'il est plus efficace d'organiser une mémoire avec des puissances entières de 2. Les dimensions possibles sont 2x2, 4x4, 8x8, 16x16 etc. Comme le temps de calcul est une fonction croissante de la dimension des blocs, alors le meilleur compromis est obtenu pour une dimension 4x4 .

III.3.6 Les différentes techniques de la quantifications vectorielle

III.3.6.1 La Mean Shape VQ (M/S VQ) et la Mean Residual VQ (M/R VQ)

C'est une technique proposée par Baker et Gray [5]. La technique M/S VQ procède en calculant la moyenne du vecteur et la quantifié par un quantificateur scalaire. Puis on soustrait la moyenne de toutes les composantes du vecteur d'entrée, le vecteur différence est ensuite vectoriellement quantifié figure III.5.

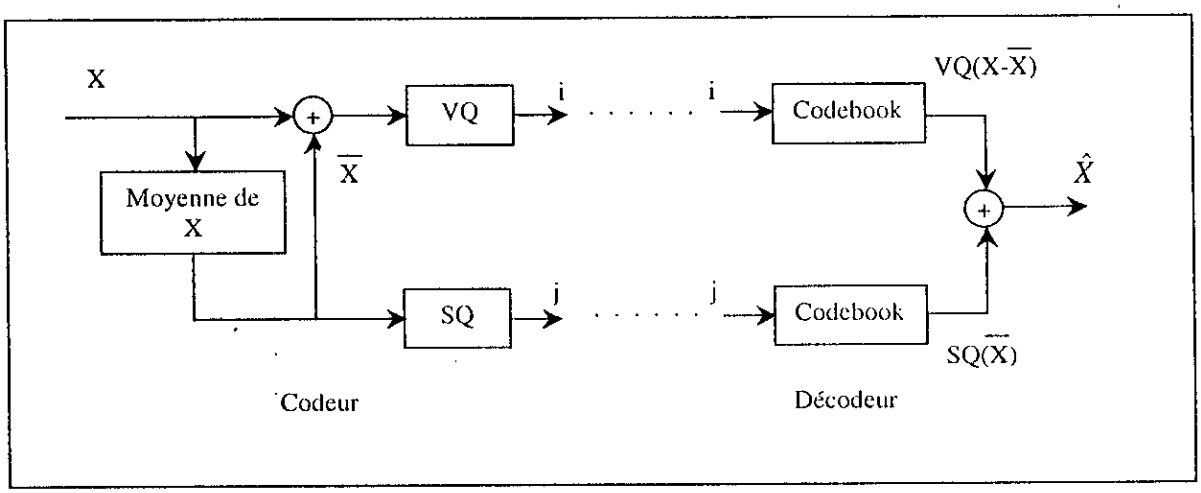


Figure III.5 *Codeur/Décodeur M/S VQ*

Une variante de la MS/ VQ est la M/ R VQ figure III.6, dans cette technique la moyenne est d'abord quantifiée scalairement puis elle est soustraite du vecteur d'entrée , cela permet d'incorporer l'erreur de quantification de la moyenne dans l'erreur de quantification du

vecteur erreur. Cette technique a permis la réduction de l'effet de bloc par la quantification de la moyenne dans MS/VQ.

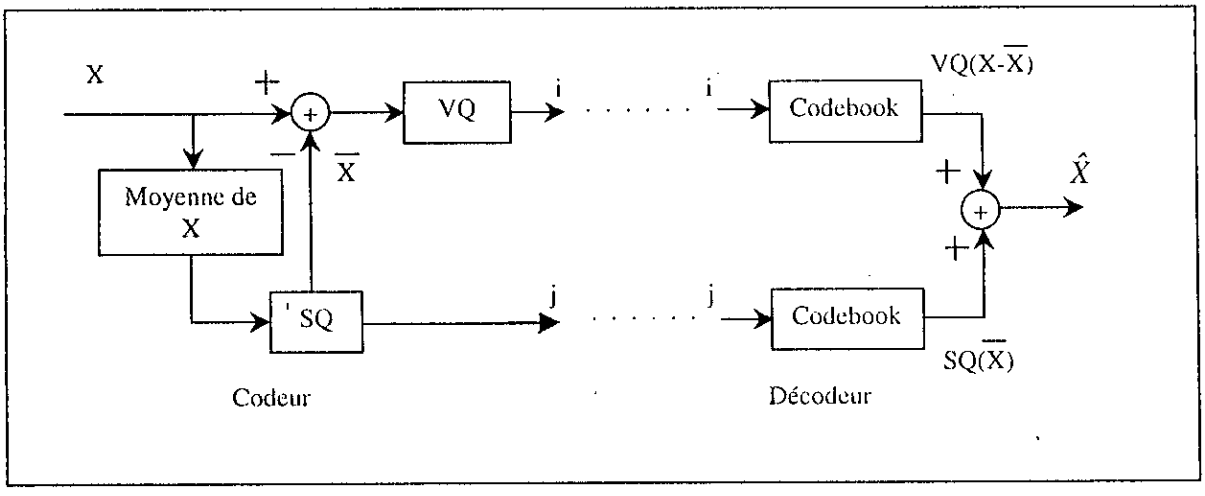


Figure III.6 *Codeur/Décodeur MR/VQ*

III.3.6.2 Quantification vectorielle par classification

Les images codées à de faibles taux souffrent souvent de la distorsion du contour et l'effet de blocs, et avec un codebook de faible taille la reproduction du contour n'est pas parfaite. Gersho et Ramamurthi [1] [22] ont remédié à ça en classifiant les vecteurs suivant des catégories. Une classification des vecteurs de l'ensemble d'apprentissage suivant des blocs d'ombre et des blocs de contour (*shade bloc and edge bloc*). Ensuite pour chaque type de vecteurs on construit un codebook qui servira par la suite à coder les vecteurs de cette classe . Comme à chaque fois un seul codebook est consulté, cela réduit énormément la complexité de calcul.

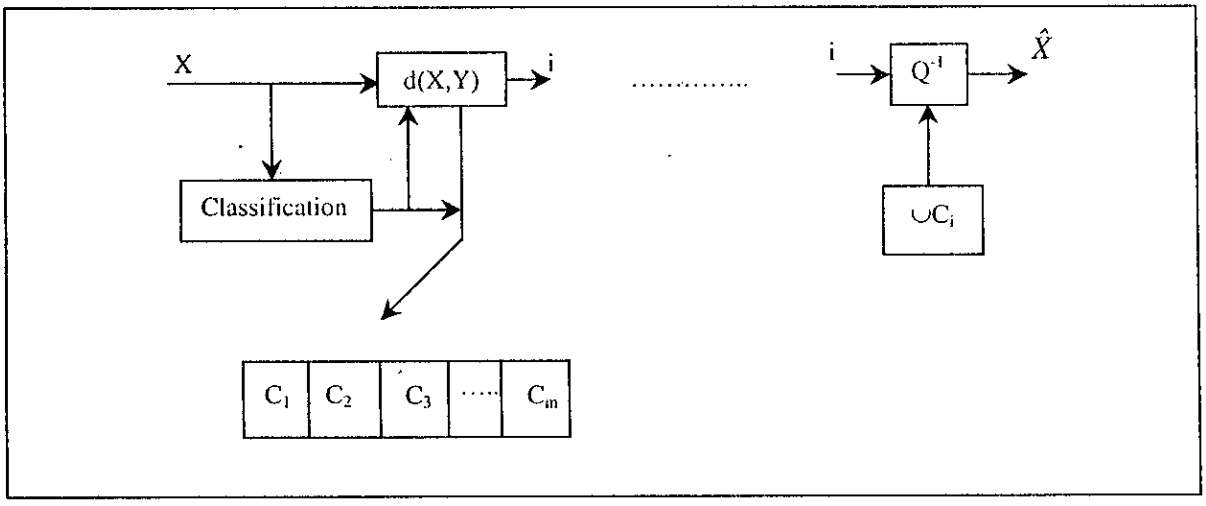


Figure III.7 *Codeur/Décodeur CVQ*

Pour améliorer les performances et surtout la qualité des images, les auteurs de la technique ont élargi le nombre de classes, par exemple la classe des blocs de contours est subdivisée en plusieurs sous-classes, suivant l'orientation et la position des contours figure III.8.

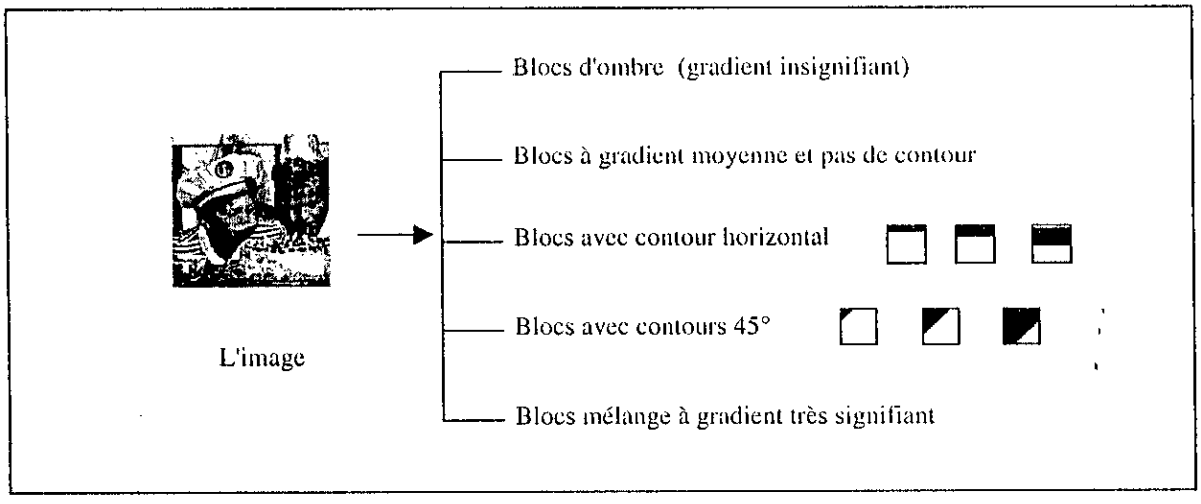


Figure III.8 *La classification des blocs de l'image* [5]

III.3.6.3 Quantification vectorielle par transformée

Dans cette technique au lieu de quantifier le vecteur d'entrée X directement, on quantifie sa transformée [1][5][41]. L'utilisation d'une transformée permet de décorréler les coefficients du bloc et de compacter toute l'énergie autour de l'origine. Les coefficients lois de l'origine seront très faibles, quelques uns peuvent même être négligés, ce qui diminue le nombre de données et par la suite diminue la complexité de calcul figure III.9.

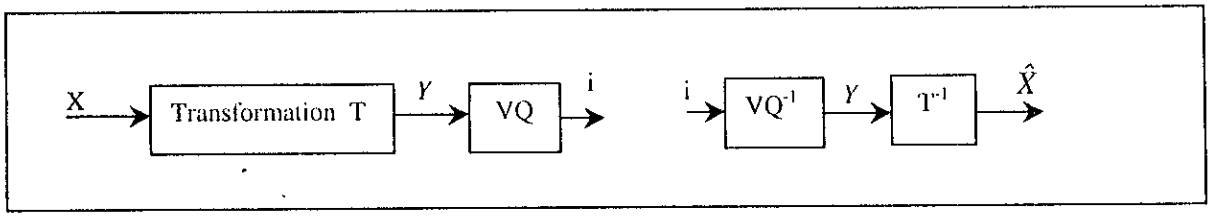


Figure III.9 *Codeur/Décodeur de la VQ transformée*

Le codebook conçu dans le domaine transformé est plus optimal que celui conçu dans le domaine spatial parce que la distribution des coefficients transformés est mieux définie que celle des coefficients spatiaux.

III.3.6.4 Quantification vectorielle prédictive

Toutes les techniques qu'on vient de voir sont sans mémoire, i.e. que chaque vecteur est codé indépendamment des vecteurs passés (ou futurs) [1][42]. La VQ prédictive utilise une mémoire pour pouvoir exploiter la corrélation qui existe entre les blocs adjacents d'une même image. La PVQ est une simple généralisation de la prédiction scalaire. D'après la figure III.10, la valeur précédente estimée (ou prédite) $\tilde{X}_k$ de X_k est calculée à partir des valeurs précédentes de X_n (n<k). Si X_k dépend de m valeurs on dit que la prédiction est d'ordre m, et :

$$\tilde{X}_k = P\left(\hat{X}_{k-1}, \hat{X}_{k-1} \ldots\ldots, \hat{X}_{k-m}\right) \tag{3.15}$$

C'est le vecteur erreur $e_k = X_k - \tilde{X}_k$ qui est vectoriellement quantifié, et à la réception le vecteur $\hat{X}_k$, qui est une approximation de X_k, est calculé par :

$$\hat{X}_k = \tilde{X}_k + e_k \tag{3.16}$$

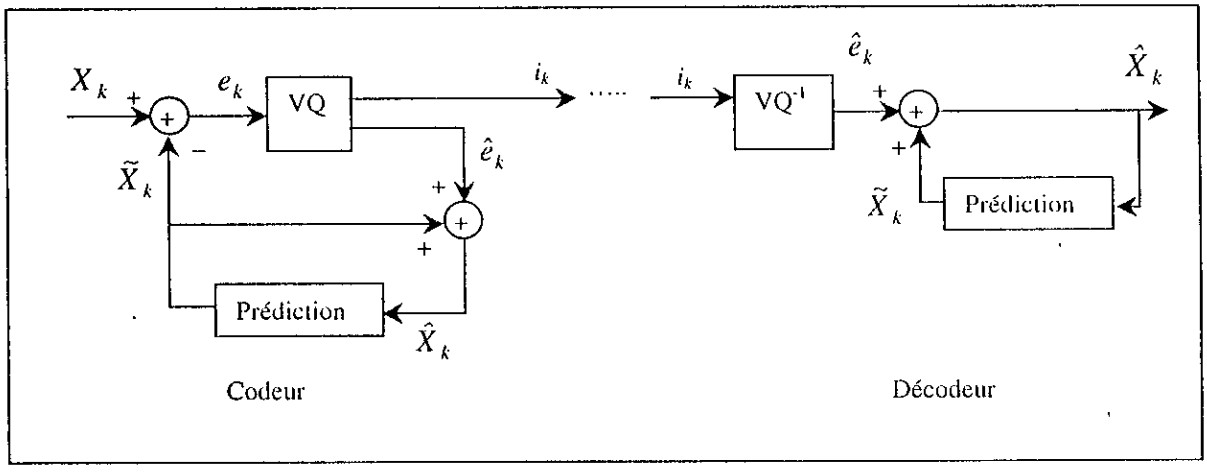


Figure III.10 *Codeur/Décodeur de la VQ prédictive*

III.3.6.5 La VQ multiétage (*Multitage VQ MSVQ*)

La technique VQ à multiétage, dite aussi VQ cascadée ou VQ résiduelle, a donné de très bons résultats dans des applications de la parole et de l'image [1][8]. La propriété de la MSVQ est que le processus de codage d'un vecteur passe par plusieurs étages. Le premier étage quantifie le vecteur d'entrée X en utilisant un codebook de faible taille, ensuite l'erreur entre le vecteur original X et la sortie du 1^{ier} étage est quantifiée par le second étage. Le figure III.11 représente un quantificateur vectoriel multiétage à deux étages, le premier étage VQ_1 quantifie X et génère $\hat{X}_1$ et i_1, le deuxième étage VQ_2 quantifie la différence $X - \hat{X}_1 = E_2$, en générant $\hat{E}_2$ et i_2. L'approximation finale de $\hat{X}$ est calculée par la somme de $\hat{X}_1$ et $\hat{E}_2$:

$$\hat{X} = \hat{X}_1 + \hat{E}_2 \tag{3.17}$$

Ensuite le codeur transmet les indices i_1 et i_2.

Si on considère un quantificateur MSVQ à p étages, avec chaque étage englobant un codebook à N vecteurs, ce quantificateur est capable de représenter M vecteurs avec :

$$M = \prod_{i=1}^{p} N_i \ , \tag{3.18}$$

en stockant seulement $\sum_{i=1}^{p} N_i$ vecteurs.

On conclue que la MSVQ nous permet d'économiser un espace mémoire important, et aussi, comme la comparaison s'effectue sur $\sum_{i=1}^{p} N_i$ vecteurs , alors la complexité de calcul sera grandement diminuée, cela permet l'utilisation des blocs de taille très importante (16x16 et 32x32).

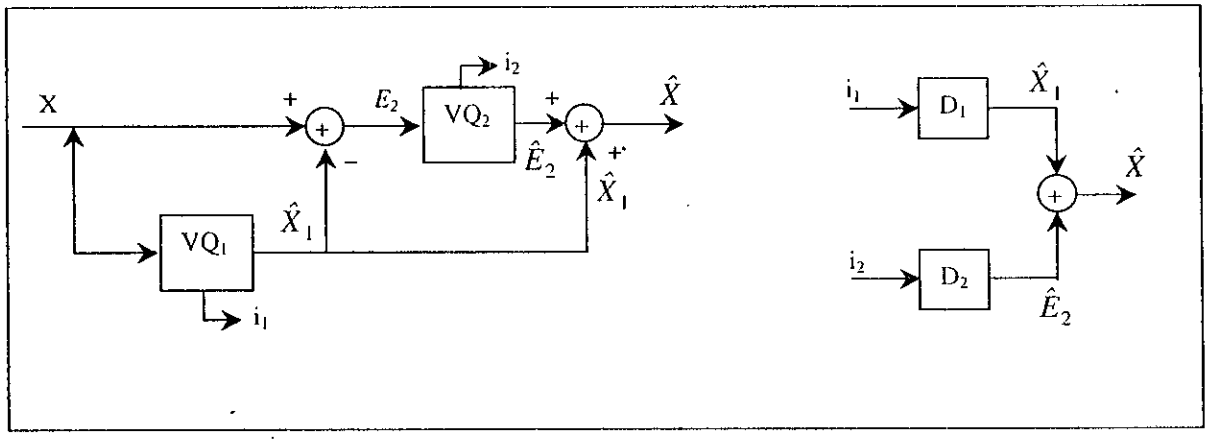


Figure III.11 *Codeur/Décodeur de la VQ multiétage (deux étages)*

III.3.6.6 La FSVQ (*Finite State VQ*)

Comme la PVQ, la FSVQ est une quantification avec mémoire, i.e. que chaque vecteur dépend des vecteurs précédents. La FSVQ et la PVQ font toutes les deux partie de ce qu'on appelle *quantification vectorielle récursive* (ou *feedback vector quantization*) [1][21][27].
La FSVQ est une collection de quantificateurs vectoriels ordinaires, tel que le vecteur d'entrée est codé par un quantificateur qui est déterminé par l'état du codeur. L'espace des états S est défini comme étant un ensemble de symboles :

$S = \{\sigma_i, i = 0,1,2...., K-1\}$, IN est l'espace des indices $IN = \{0,1,..., N-1\}$.

Le codeur FSVQ est une application α :

$$\alpha : R^k \times S \longrightarrow IN,$$

ayant un vecteur $X \in R^k$, et un état $s \in S$, alors le vecteur X sera codé par $\alpha(X,s) \in IN$.
Au codeur, est associé une fonction dite fonction de prochain état, elle est définie comme suit: $f : IN \times S \longrightarrow S$,
Cette fonction permet de définir le prochain état du codeur. Si l'état actuel du codeur est $s \in S$, X le vecteur à coder alors l'état prochain du système est $f(u, s)$, avec $u = \alpha(X, s)$ (figure II.12)

Le décodeur FSVQ est une application β :

$$\beta : N \times S \longrightarrow R^k ,$$

ayant un indice $u \in IN$ et un état s le symbole u est décodé en $\hat{X}$:

$$\hat{X} = \beta(u, s) = \beta(\alpha(X, s), s) \qquad (3.19)$$

A chaque état s est associé un codebook C_s :

$$C_s = \{\beta(u, s), u \in IN\},$$

L'union de tout les codebooks représente le super codebook : $C = \bigcup_{s \in S} C_s$.

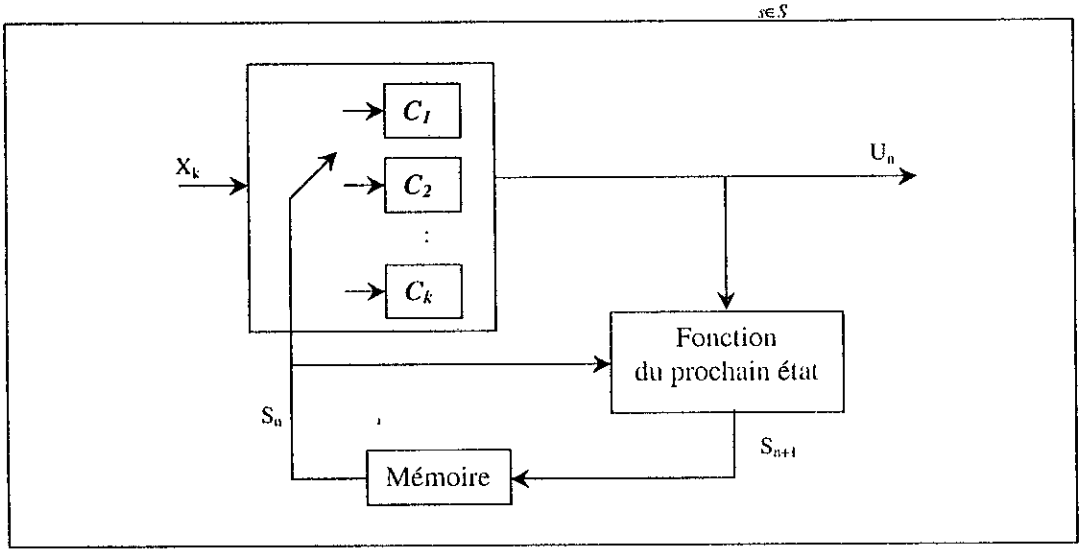


Figure III.12 *Codeur FSVQ*

La FSVQ était appliquée à l'image par Aravind et Gersho, elle exploite les dépendances linéaires et non-linéaires de l'image. L'état d'un vecteur est déterminé en fonction des états des vecteurs adjacents dans l'image.

III.3.6.7 Cache VQ

Un quantificateur vectoriel cache est un codeur sous-optimal qui permet de réduire la corrélation interbloc [3][8]. Cela est très profitant dans le cas d'une séquence d'images où il y a une très forte corrélation entre frames et même dans la même frame. Les propriétés statistiques des frames changent de l'une à l'autre, et avec un codeur à caractéristiques probabilistiques fixes on ne peut pas atteindre un résultat satisfaisant. Pour cela la Cache VQ est utilisée, elle permet de suivre les variations des statistiques de la source, et d'autre part elle permet un temps de codage plus faible. La cache VQ a un super-codebook à haute qualité et un débit élevé, ce codebook peut être conçu par un algorithme ordinaire de VQ. Un cache-codebook est associé à ce super-codebook de taille très réduite, il contient les vecteurs codes les plus utilisés.

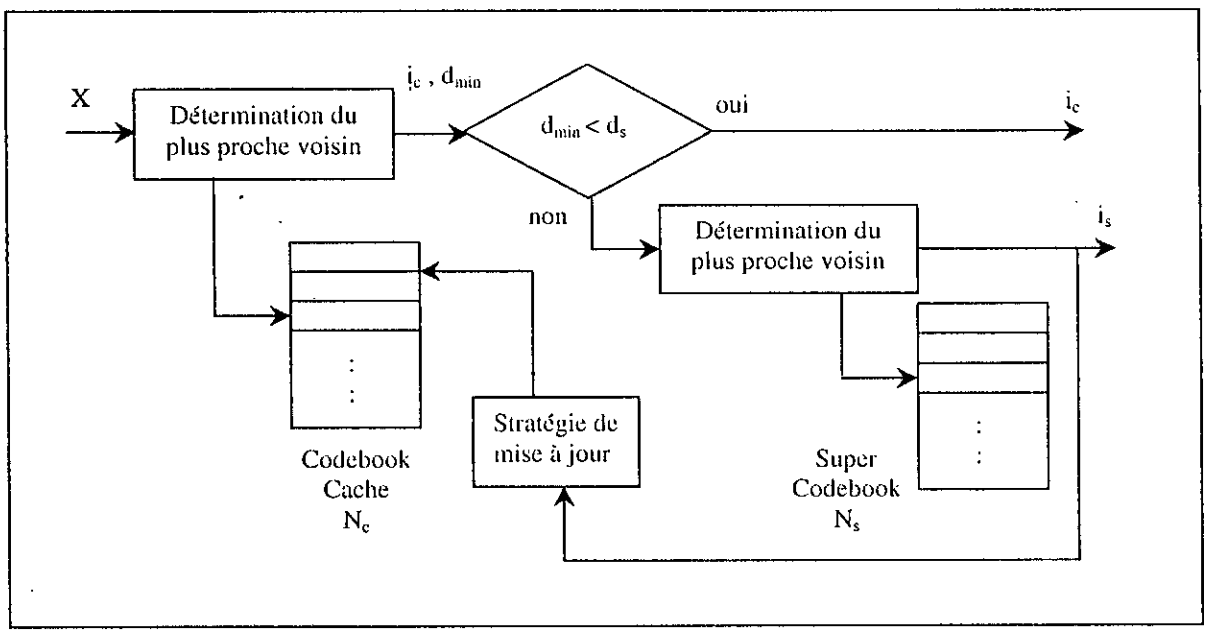


Figure III.13 *Codeur de la VQ Cache*

Pour coder un vecteur d'entrée on cherche d'abord dans le cache-codebook le vecteur code qui vérifie une distance seuil (si ça réussit c'est un *hit* ou succés), figure III.13, c'est l'adresse du vecteur code dans le cache est transmise. Dans le cas où on ne trouve pas un vecteur code convenable dans le cache alors on cherche dans le super-codebook (c'est un *miss* ou échec), donc c'est l'adresse du vecteur code dans le super-codebook qui est transmise.

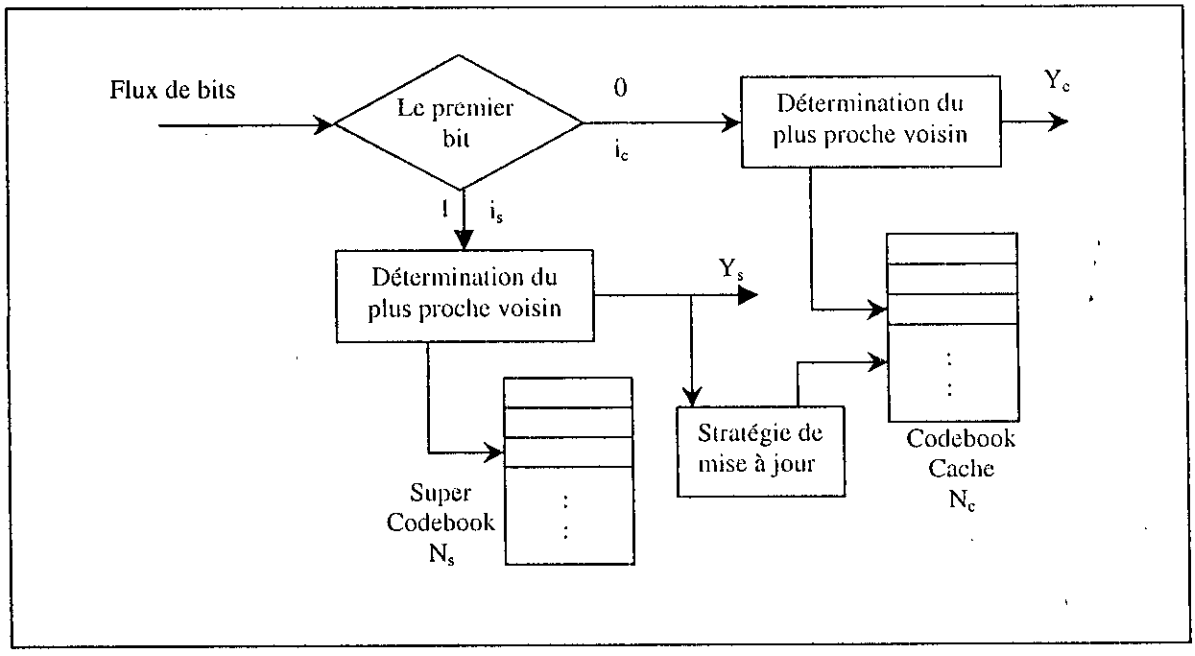


Figure III.14 *Décodeur de la VQ Cache*

Si M est la taille du super-codebook et N celle du cache-codebook alors le débit sera donné par :

$$r = \frac{1}{k}(1 + h * log_2(N) + (1-h) * log_2(M)) \qquad (3.20)$$

avec h la probabilité de coder un vecteur à partir du cache-codebook, cette probabilité dépend de la distance seuil d_s précisée et de la stratégie de mise à jour du cache-codebook. La mise à jour du cache doit être effectué au codeur et au décodeur (de préférence sans la transmission d'information), elle doit conduire à h le plus élevé possible pour réduire le débit effectif.

Une opération de mise à jour est effectuée quand il y a un miss i.e. échec d'utilisation du cache. Les stratégies de mise à jour du cache sont empruntées de celle utilisées dans les mémoires des ordinateurs, parmi elles on cite ;LRU pour *Least Recently Used*, LFU pour *Least Frequently Used*, FIFO pour *First In First Out*.

III.4 Conclusion

La quantification vectorielle constitue l'une des techniques les plus utilisées dans le codage d'images vu le taux de compression très intéressant qu'elle permet. Mais son utilisation directe pose des problèmes pratiques, en particulier aux systèmes travaillant en temps réel. Pour cela plusieurs solutions ont été proposées pour détourner les deux barrières principales de la VQ; la taille du dictionnaire et la recherche exhaustive du vecteur plus proche.

Toutes ces solutions se sont basées sur les caractéristiques statistiques des images, en exploitant les corrélations spatiale et temporelle. La VQ classifiée a permit de réduire grandement le problème de discontinuité de blocs, d'un autre côté la VQ transformée a permis de réduire la taille des vecteurs en négligeant les coefficients à haute fréquence. Des versions adaptatives de la VQ comme la FSVQ et la Cache VQ ont permis une exploitation meilleure des redondances tout en offrant un temps de codage nettement plus réduit.

Bien que la VQ ne soit pas universelle, l'intérêt qu'elle apporte la laisse toujours prévue dans les systèmes de compression.

Chapitre IV

Codage interframe

IV.1 Introduction

La caractéristique principale d'une séquence d'images est bien la redondance temporelle, qui est très élevée, où souvent on trouve dans la scène certains objets qui se déplacent sur un fond stable, et quand le déplacement devient plus faible la corrélation entre deux frames devient très élevée . Pour cela les codeurs des séquences prennent toujours en considération cette corrélation pour réaliser un codage interframe.

On trouve ici la méthode la plus utilisée qui est la compensation de mouvement, elle consiste à détecter ou estimer le mouvement des objets, et puis coder les paramètres du mouvement qui sont le déplacement, la rotation et l'homotétie. L'efficacité de la compensation de mouvement dépend fortement des caractéristiques statistiques de la séquence d'un côté, et du module de détection de mouvement d'un autre côté.

IV.2 Codage interframe par interpolation

Les techniques interpolatives sont utilisées dans plusieurs codeurs, tel que le standard MPEG et la télévision numérique [28]. Elles consistent à déduire les lignes non transmises à partir des lignes déjà transmises par interpolation. Un algorithme dit up-conversion, a été utilisé dans les récepteurs TV, où une transmission entrelacée est utilisée. Une image est transmise en deux trames, la première contient les lignes paires et la deuxième les lignes impaires .

La trame des lignes paires est notée T_e, et celle des lignes impaires est notée T_o, $X(T,l,c)$ représente un pixel appartenant à la trame T, à la ligne l et la colonne c. Avec ces notations une séquence d'images peut être représentée par une fonction à 3 variables comme suit :

$$X=X(T,l,c) \tag{4.1}$$

La séquence des trames paires et impaires sera donnée par :

$$\{\ldots\ldots\ldots\ldots, (T-1)_o, (T-1)_e, T_o, T_e, (T+1)_o, (T+1)_e, \ldots\ldots\ldots\ldots\}$$

L'algorithme de up-conversion consiste à doubler le débit des trames, en ajoutant 2 trames entre T_o et T_e comme suit

$$\{\ldots\ldots\ldots\ldots, (T-1)_e, T_o, T_e^*, T_o^*, T_e, (T+1)_o, \ldots\ldots\ldots\ldots\}$$

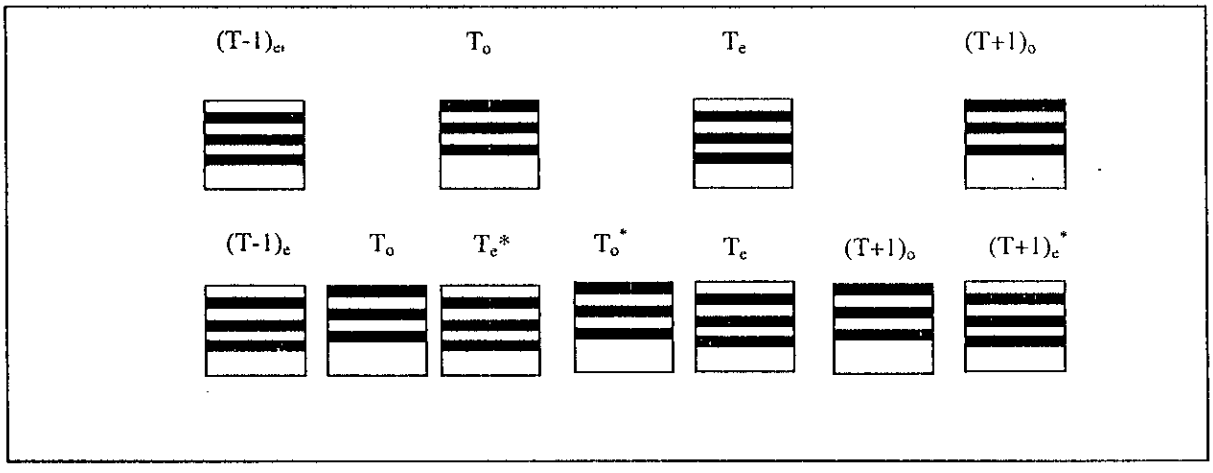


Figure IV.1 *Différents types de trames utilisées dans l'algorithme up-conversion*

Cet algorithme consiste à calculer T_e^* , T_o^* , $(T+1)_o^*$....., et cela par interpolation. Le plus simple serait de mettre : $T_e^* = T_o$ et $T_o^* = T_e$, c'est très simple à implémenter, d'ailleurs ça a été adopté par plusieurs producteurs de récepteurs de TV fonctionnant à 100Hz.

Pour améliorer les performances, une interpolation est utilisée comme suit :

$$X(T_e^*, l, c) = 1/2 [X(T_o, l-1, c) + X(T_o, l+1, c)] \quad (4.2)$$

$$X(T_o^*, l, c) = 1/2 [X(T_e, l-1, c) + X(T_e, l+1, c)] \quad (4.3)$$

C'est une interpolation purement spatiale, elle engendre une perte dans la résolution des scènes sans mouvement. Pour remédier à ce problème, un filtrage médian est utilisé, il permet de préserver les contours, parmi les 3 points verticaux, on prend celui qui a l'amplitude médiane comme suit :

$$X(T_e^*, l, c) = Med [X(T_o, l-1, c), X(T_o, l+1, c), X(T_e, l, c)] \quad (4.4)$$

$$X(T_o^*, l, c) = Med [X(T_e, l-1, c), X(T_e, l+1, c), X((T+1)_o, l, c)] \quad (4.5)$$

Med[ ], signifie la valeur médiane de toutes les valeurs citées en argument.

Ce filtre utilise l'information spatiale et l'information temporelle, il s'adapte aux mouvements de la scène. Cette technique d'interpolation par filtrage médian a donné de très bon résultats au codage interframe, dans certaines applications un bloc de détection de mouvement est combiné pour remédier au problème de déformation de contour, et pour profiter de l'information temporelle.

IV.3 La Quantification vectorielle des séquences

La quantification vectorielle des images fixes consiste à concevoir un codebook à partir de l'image et le stocker au niveau du codeur et du décodeur [18]. Ensuite la quantification cherche pour chaque vecteur d'entrée le plus proche vecteur code du codebook et transmet son indice. Le décodeur utilise l'indice comme adresse dans le codebook pour extraire le vecteur code correspondant . Le codebook dans ce cas est fixe le long de l'opération de codage et de décodage. Dans le cas d'une séquence d'images, les statistiques varient d'une frame à une autre, et d'une scène à une autre, avec une VQ ordinaire ou statique la qualité des images se dégrade avec le temps.

Une solution peut être envisagée c'est la détermination pour chaque frame de son propre codebook, mais cela pose le problème de stockage (mémoire) des codebooks. Une technique meilleure a été proposée elle consiste à la mis à jour continue du codbook (*codebook update*), pour qu'il puisse suivre les statistiques des images constituant la séquence,

la mise à jour doit être effectuée dans le codeur et le décodeur en même temps, et cela sans transmettre une information entre les deux, pour ne pas augmenter le débit.

IV.3.1 Remplacement d'indice

Le principe de cette méthode est illustré dans la figure IV.2, deux phases sont nécessaires; la première consiste à générer le codebook, elle est représentée par le trait en pointillé, en utilisant un apprentissage sur la première image de la séquence. Cette image est quantifiée puis stockée dans la mémoire du codeur après elle est transmise. La deuxième phase, représentée par un trait continu, consiste à coder les frames suivantes. L'indice généré par la quantification de chaque bloc est comparé avec l'indice correspondant dans la mémoire, s'ils sont différents, l'indice dans la mémoire est remplacé par l'indice généré. Un bit est envoyé au décodeur pour signaler le remplacement de l'indice.

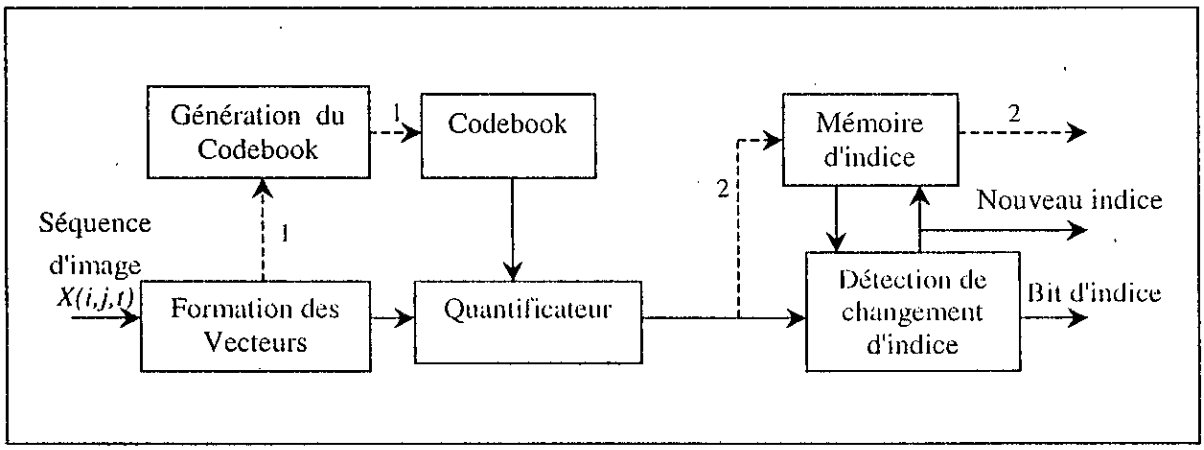


Figure IV.2 *Schéma d'un codeur de séquence par remplacement d'indice*

L'inconvénient de cette approche est qu'il faut concevoir un dictionnaire à partir de toute la séquence, pour pouvoir suivre ses statistiques, mais cela conduit à un codebook très large et par la suite un débit élevé.

IV.3.2 Remplacement du codebook

L'idée de cette technique consiste à mettre à jour ou remplacer les vecteurs du codebook continuellement suivant les changements des caractéristiques statistiques de la séquence.

IV.4 La compensation du mouvement

Elle est très utilisée dans le codage des séquences d'images, où les frames sont liées par un simple mouvement de blocs. Cette technique consiste à utiliser l'information temporelle pour réduire la quantité de données à transmettre. Le schéma le plus simple et le plus général de la compensation de mouvement est donné par la figure IV.3

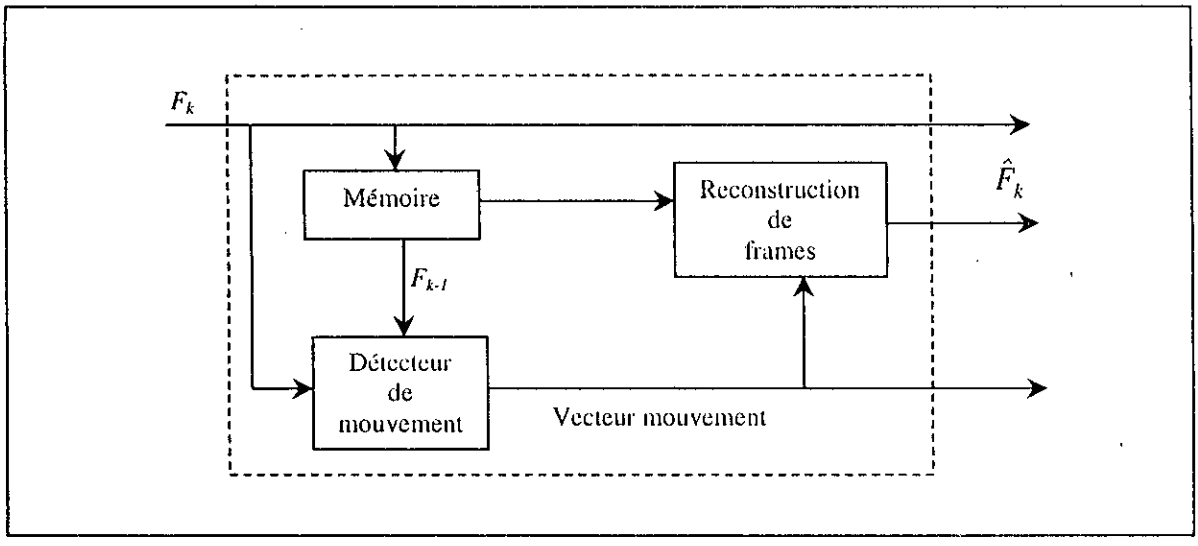


Figure IV.3 *Synoptique d'un schéma de compensation de mouvement*

Dans ce synoptique le module le plus important est le détecteur de mouvement qui opère sur deux frames successives pour extraire le vecteur mouvement qui a comme composantes le déplacement, l'angle de rotation et le facteur d'homotétie.

IV. 4.1 Détection et estimation de mouvement

La prédiction du mouvement est une technique utilisée pour réduire la redondance temporelle, ou la corrélation, dans les séquences . L'idéal serait de reconnaître les différents objets de la scène, puis de les suivre de la frame F_t à la frame F_{t+1}, et suivant les déplacements et les mouvements on peut déduire, ou prédire, F_{t+1}.

Donc l'estimation du mouvement est la partie essentielle dans un système de compensation de mouvement, pour cela il y a plusieurs techniques et méthodes qui ont été proposées tel que la technique de correspondance de blocs(*Blocks Matching*), le modèle affine, détection par la méthode de gradients, détection par les descripteurs de Fourrier ...

IV.4.1.1 Correspondance de bloc (*Blocks Matching*)

C'est la première technique qui a été utilisée dans les premiers systèmes, elle était destinée aux images de télévision [24]. Elle est basée sur des hypothèses simplificatrices pour la reconnaissance des objets et le calcul des vecteurs mouvement, ce qui lui permet d'être

implémentée dans beaucoup d'applications pratiques. En effet toute technique de codage des mouvements (détection, estimation) doit satisfaire l'incontournable contrainte de temps réel des images animées.

La technique de BM permet la détection des mouvements translationnels seulement (horizontal et vertical)

Principe

La technique BM prend le bloc B_j de la frame courante F_k, et cherche un bloc lui ressemblant ou identique dans la frame précédente F_{k-1},figure IV.4.

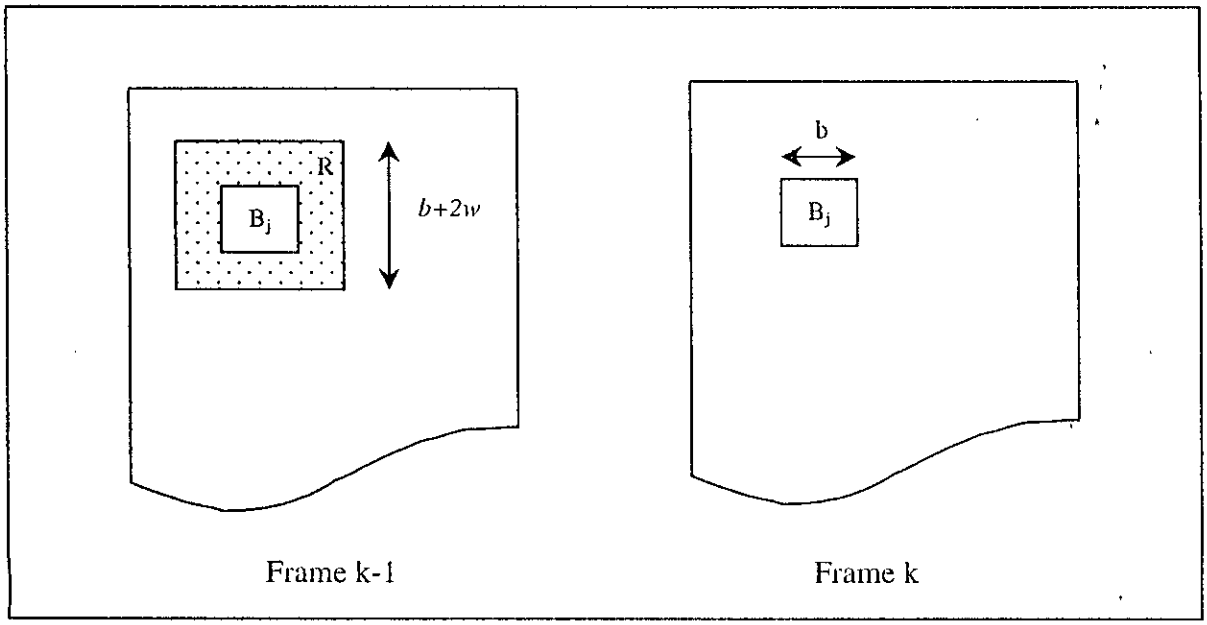


Figure IV.4 *Les deux frames utilisées dans la détection du vecteur mouvement*

La ressemblance est basée sur le principe de calcul d'une distance. La frame F_k est divisée en blocs de taille bXb, et chaque bloc est comparé avec tous les blocs appartenant à une région R_j centrée sur le bloc B_j^{k-1}, cette région est de taille $(b+2w)^2$ où w est le décalage maximal autour de B_j^{k-1}. Donc la position du bloc le plus ressemblant à B_j^k dans la région R_j permet de calculer le vecteur mouvement ou déplacement.

La technique BM nécessite $(2n+1)^2$ décalage, et si on utilise l'erreur MSE comme critère de ressemblance, alors chaque décalage nécessite b^2 multiplications et b^2 additions, donc $b^2(2w+1)^2$ multiplications et $b^2(2w+1)^2$ additions pour une recherche complète. Pour cela on trouve des versions rapides de la technique BM.

IV.4.1.1.1 Correspondance de bloc rapide (*Fast BM*)

La version rapide de la BM consiste à chercher le bloc B_j dans la région de recherche en 9 positions éloignées notés S_1 dans la figure IV.5 (4 coins, 4 milieux et le point central), dans la deuxième étape la taille du carré de recherche est réduite, et le centre du carré de recherche sera confondu avec le centre du carré qui a donné la meilleure correspondance dans la première étape. Dans cette étape on aura 8 positons de recherche(S_2). On refait la même opération dans la troisième étape, enfin on se retrouve avec 25 recherches au lieu de 289 pour une recherche exhaustive(*b=4,w=8*), figure IV.5.

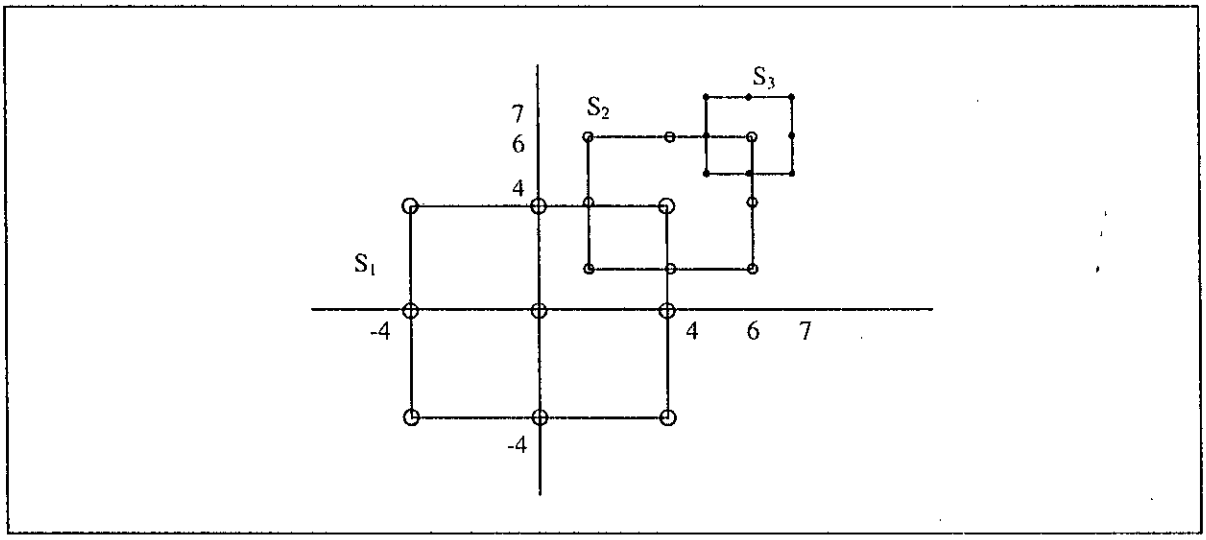


Figure IV.5 *Correspondance de blocs rapide*

IV.4.1.2 La technique de corrélation [30]

Cette technique permet de calculer la corrélation entre le bloc B_j^k et la région de recherche R_j^{k-1} et chercher la position où la corrélation est plus forte (maximale). La corrélation entre B_j^k et R_j^{k-1} est donnée par :

$$r(m,n)=\sum_p\sum_q R_j(p,q)B_j(p-m,q-n) \tag{4.6}$$

La fonction *r(m,n)* nous donne le taux de ressemblance entre B_j^k et avec tous les blocs de même taille contenus dans R_j . La valeur maximale de *r(m,n)* nous donne les coordonnées *d(m,n)* du bloc le plus ressemblant à B_j^k dans R_j .Pour éviter des valeurs élevées de *r(m,n)* on utilise une version normalisée de la corrélation donnée par :

$$\rho(m,n)=\frac{\sum_p\sum_q\left(R(p,q)-\overline{R}(p,q)\right)\left(B(p-m,q-n)-\overline{B}\right)}{\sqrt{\sum_p\sum_q\left(R(p,q)-\overline{R}\right)^2\sum_p\sum_q\left(B(p-m,q-n)-\overline{B}\right)^2}} \quad (4.7)$$

où $\overline{B}$ est la moyenne du bloc B et $\overline{R}$ est la moyenne des coefficients coïncidant avec B, donc $\rho(m,n)$ est comprise entre –1 et +1. L'intérêt de cette technique est de pouvoir utiliser la FFT pour les calculs.

IV.4.1.3 Le modèle affine pour la détection de mouvement :

Cette technique est une généralisation de la technique BM [7]. En plus du fait qu'elle permet la détection des translations comme la BM, elle permet la détection des rotations et les hométeties et elle prend en compte les variations de l'éclairage. Soit R' une région (à $t = t_2$) cette région résulte de la région R (à $t = t_1$) suivant la relation de déformation :

$$p \longrightarrow Mp + d$$

avec :

$$M=\begin{bmatrix} s_x\cos(\theta_x) & -s_y\sin(\theta_y) \\ s_x\sin(\theta_x) & s_y\cos(\theta_y) \end{bmatrix} \quad (4.8)$$

et $d=\begin{bmatrix} d_x & d_y \end{bmatrix}$ (4.9), et $p=\begin{bmatrix} x & y \end{bmatrix}$ (4.10)

Avec d le vecteur de translation et la matrice M représente les rotations et les homotéties. Dans cette matrice s_x et s_y représentent les paramètres de l'homotétie (*scaling*) suivant x et y (agrandissement ou réduction des objets), θ_x et θ_y correspondent aux angles de rotation des objets. Dans ce modèle l'intensité est prise en compte avec deux paramètres :

$$I_2 = rI_1 + c \quad (4.11)$$

avec r le coefficient d'amplitude et c le décalage de luminance. Pour détecter tous les mouvements produits entre deux frames, il faut déterminer tous les paramètres déjà cités ($s_x, s_y, \theta_x, \theta_y, d_x, d_y, r$ et c), en minimisant l'erreur :

$$E(M,d,r,c)=\sum_{p\in R}\left|I(p,t_1)-rI(Mp+d,t_2)-c\right|^2 \quad (4.12)$$

c'est une optimisation nonlinéaire à 8 variables, qui peut être effectuée par un algorithme itératif, mais la convergence vers un minimum absolu n'est pas garantie. Pour déterminer r et c on procède ainsi :

$$\partial E/\partial r = 0 \quad (4.13)$$

$$\partial E/\partial c = 0 \quad (4.14)$$

et remplacer leurs valeurs dans E, on obtient un problème à 6 variables seulement :

$$r^* = \frac{A\sum I_1 I_2 - \sum I_1 \sum I_2}{A\sum I_2^2 - \left(\sum I_2\right)^2} \tag{4.15}$$

$$c^* = \frac{\sum I_1 \sum I_2^2 - \sum I_1 \sum I_1 I_2}{A\sum I_2^2 - \left(\sum I_2\right)^2} \tag{4.16}$$

avec : $I_1 = I(p, t_1), I_2 = I(Mp + d, t_2), A = \sum_{p \in R} 1$

on aura :

$$E^*(M,d) = E(M,d,r^*,c^*) = \sum I_1^2 - r^* \sum I_1 I_2 - c^* \sum I_1 \tag{4.17}$$

$$= \sum I_2^2 - \frac{A\left(\sum I_1 I_2\right)^2 + \sum I_2^2 \left(\sum I_1\right)^2 - 2\sum I_1 \sum I_2 \sum I_1 I_2}{A\sum I_2^2 - \left(\sum I_2\right)^2} \tag{4.18}$$

$$E^*(M,d) = \sum I_2^2 - K(M,d) \tag{4.19}$$

donc minimiser E^* revient à maximiser K (6 variables).

Dans notre cas on suppose les rotations uniformes i.e. $\theta_x = \theta_y = \theta$ et des homotéties uniformes i.e. $s_x = s_y = s$, d'où on aura un système plus réduit, à 4 variables seulement(θ, s, d_x, d_y).

IV.5 Conclusion

Le codage interframe consiste à réduire la redondance temporelle. Souvent la technique utilisée est la compensation de mouvement, elle consiste à extraire l'information de mouvement, cette information se caractérise par un vecteur de mouvement, ensuite elle génére la frame courante en utilisant ce vecteur et la frame précédente. La robustesse d'un schéma de compensation de mouvement dépend fortement du module d'estimation de mouvement.

La technique de correspondance de bloc est très simple dans sa mise en forme, mais elle est gourmande en temps de calcul, elle permet de détecter les mouvements translationnels seulement. Une version rapide de cette technique est proposée . Un autre algorithme permet la

détection des trois types de mouvement ; la translation, la rotation et l'homotétie, c'est l'algorithme du modèle affine.

Chapitre V

Applications

V.1 Introduction

Dans ce chapitre on applique les algorithmes cités dans les chapitres III et IV sur des images standards. On a procédé à l'étude du codage intraframe où la quantification vectorielle cache est utilisée. On a fait varié plusieurs paramètres et à chaque fois on trace la courbe correspondante illustrant les variations d'un paramètre par rapport aux autres. On a montré l'effet de filtrage de la VQ et l'utilisation des images autres que celles de l'ensemble d'apprentissage pour voir l'universalité de la VQ dans le codage de l'image.

Ensuite on a abordé le codage interframe, où on a étudié trois algorithmes de détection de mouvement, à savoir la correspondance de bloc BM, la version rapide de cette technique FBM et le modèle affine, des résultats sont exposés sous forme d'images.

V.1 Codage intraframe

V.1.1 Quantificateur cache

Un quantificateur cache est caractérisé par les paramètres suivants :

N_s la taille du super codebook

N_c la taille du cache codebook ($Ns > Nc$)

d_s la distance seuil

La stratégie de mise à jour (voir figure III.13)

De ces paramètres se dégage un autre facteur qui est h la probabilité que le vecteur d'entrée X soit codé à partir du cache codebook. Il est clair que nous devons garder h la plus élevée possible pour atteindre des débits plus faibles, mais au prix d'une dégradation dans la qualité.

Si le vecteur d'entrée X est codé à partir du super codebook alors le nombre de bits utilisé pour coder un bloc est :

$$n_s=log_2(N_s) \tag{5.1}$$

quand il est codé à partir du cache alors le nombre de bits est :

$$n_c=log_2(N_c) \tag{5.2}$$

On pose :

$$r_s=\frac{1}{k}log_2(N_s) \tag{5.3}$$

$$r_c=\frac{1}{k}log_2(N_c) \tag{5.4}$$

Comme à chaque vecteur est associé un bit d'information, alors on pose :

$$r_i=\frac{1}{k} \tag{5.5}$$

Connaissant la probabilité h, on peut évaluer le débit effectif qui est donné par :

$$\begin{aligned} r_e &= r_i + h * r_c + (1-h) * r_s \\ r_e &= \frac{1}{k}(1 + h * log_2(N_c) + (1-h) * log_2(N_s)) \end{aligned} \tag{5.6}$$

V.1.2 Stratégies de mise à jour

La stratégie de mise à jour détermine la façon dont le cache codebook est modifié. Ce codebook varie avec le temps en fonction de la source. Le cache initial est déterminé soit :

- Par un choix aléatoire d'un ensemble de vecteurs du super-codebook.
- Soit, pendant la conception du super codebook on sélectionne les vecteurs les plus probables et on les stocke au début du codebook, ensuite on les transfert au cache.
- Soit on prend carrément un nombre de vecteurs égal à la taille du cache.

Toutes ces techniques donnent des résultats très proches.

V.1.2.1 Le codage

Toujours le vecteur d'entrée est codé à partir du cache, si la meilleure correspondance est vérifiée avec une distance :

$$d(X, Y_c) < d_s \tag{5.7}$$

Avec d_s une distance seuil prédéterminée. Quand le vecteur X est codé à partir du cache, bien entendu en utilisant n_c bits, au lieu de n_s bits, plus un bit mis à 0 pour signaler au décodeur que le vecteur est codé à partir du cache.
Dans ce cas le cache n'est pas mis à jour, ni au codeur ni au décodeur.

Dans le cas où la distance $d(X, Y_c) \geq d_s$, alors le vecteur code de reproduction est recherché dans le super codebook, cette fois ci le vecteur est codé sur n_s bits, en envoyant le bit d'information avec la valeur 1, pour que le décodage se fasse à partir du super codebook.
Dans ce cas une mise à jour est effectuée au cache codebook, tel que le vecteur trouvé dans le super codebook soit stocké dans le cache suivant une certaine stratégie.
La procédure de codage est donnée par l'algorithme suivant :

0) j=0, M la taille de la source en vecteurs

1) $d_{min} = d(X(j), Y_c(i_c))$

2) si $d_{min} < d_s$ *alors* *Code ='0' +'i_c' aller à 3).*

sinon $d_{min} = d(X(j), Y_s(i_s))$

Code ='1' + 'i_s'

Mise_à_jour_du_Cache() ;

finsi

3) j=j+1 , si j>M alors fin sinon aller à 1)

V.1.2.2 Le décodage

Le décodage s'effectue en procédant dans le sens inverse, tel qu'on reçoit un flux de bits au niveau du décodeur, on identifie le premier bit, si ce bit est à 0, alors les n_c bits suivants constitueront le code ou l'indice du vecteur de reproduction dans le cache codebook, ce dernier est adressé directement (*look up table*) sans mise à jour. Dans le cas où le premier

bit serait à un alors les n_s bits qui suivent constitueront l'indice du vecteur code dans le super codebook, ce vecteur est utilisé pour reconstituer l'image et il est stocké dans le cache suivant une certaine stratégie.
La procédure de décodage est donnée comme suit :

0) j=0, $n_c=log_2(N_c)$, $n_s=log_2(N_s)$, M la taille de la source en vecteurs
1) si premier bit = 0 alors Lire n_c bits dans i_c et extraire $Y_c(i_c)$
sinon Lire n_s bits dans i_s et extraire $Y_s(i_s)$
Mise_à_jour_du_Cache() ;
2) j=j+1 si j>M alors fin sinon aller à 1)

V.1.2.3 La stratégie du premier entré premier sortie (FIFO)

Elle consiste à remplacer toujours le plus ancien dans le cache.

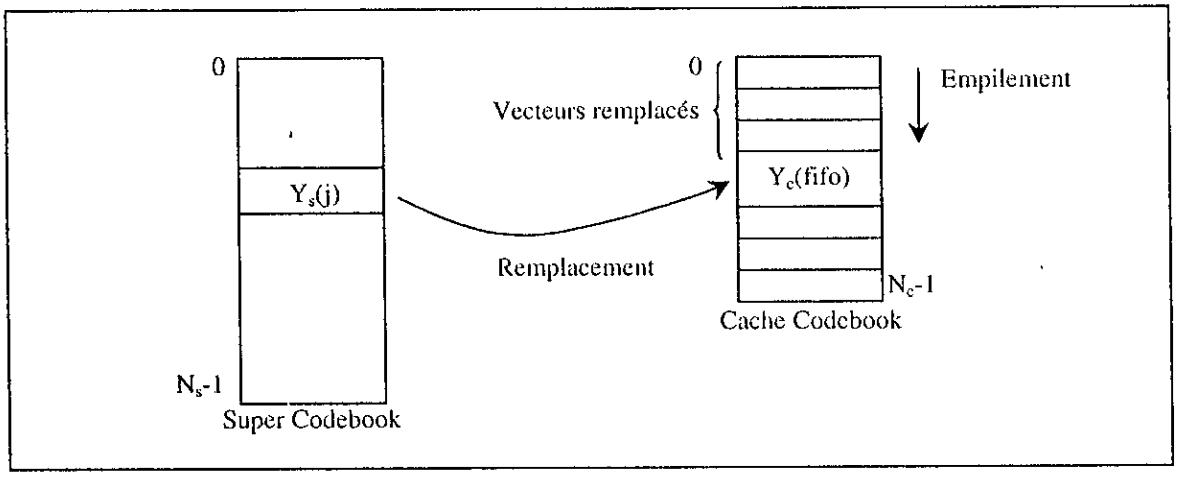


Figure V.1 *La stratégie de mise à jour FIFO*

V.1.2.4 La stratégie du moins fréquemment utilisé (LFU)

Elle consiste à remplacer le vecteur le moins fréquemment utilisé dans le cache par le vecteur trouvé dans le super codebook.

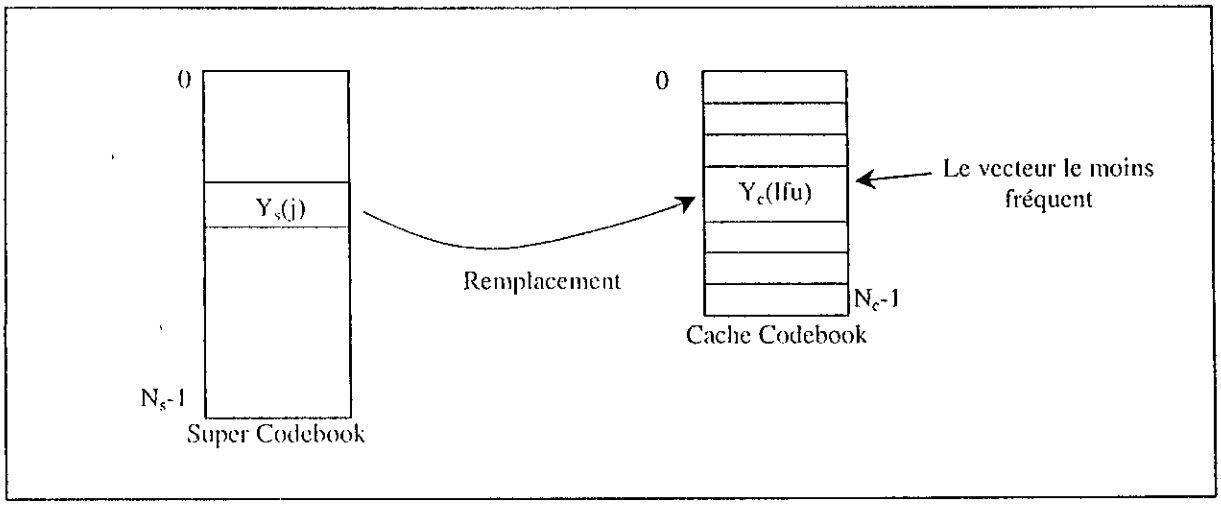


Figure V.2 *La stratégie de mise à jour LFU*

V.1.3 Résultats et analyses

Le super codebook a une taille de 512 vecteurs, chaque vecteur est formé de 16 composantes, parce qu'on a utilisé des blocs de 4x4

La figure V.3.a montré l'image originale utilisée dans les applications, et l'image V.3.b montre une image décodée par un quantificateur vectoriel ordinaire à 0.75 bpp. La figure V.9 montre la quantification cache d'une image hors l'ensemble d'apprentissage.

a *b*

Figure V.3 *a) L'image originale b)Image codée par une simple VQ a 0.75 bpp*

Image codée avec r =0.59 bpp
PSNR=31.31 dB
d_s=200

Image codée avec r =0.50 bpp
PSNR=30.56
d_s=1500

Figure V.4 *Images codées avec un codeur Cache VQ avec la stratégie FIFO Avec taille du cache =64*

Image codée avec r =0.59 bpp
PSNR=30.84 dB
d_s=200

Image codée avec r =0.53 bpp
PSNR=30.39
d_s=1500

Figure V.5 *Images codées avec un codeur Cache VQ avec la stratégie FIFO Avec taille du cache =128*

Image codée avec r =0.61 bpp
PSNR=30.28 dB
d_s=200

Image codée avec r =0.57 bpp
PSNR=30.06
d_s=1500

Figure V.6 *Images codées avec un codeur Cache VQ avec la stratégie FIFO Avec taille du cache =256*

Image codée avec r =0.62 bpp
PSNR=28.15 dB
d_s=50

Image codée avec r =0.49 bpp
PSNR=27.20
d_s=1500

Figure V.7 *Images codées avec un codeur Cache VQ avec la stratégie LFU Avec taille du cache =64*

Image codée avec r =0.62 bpp
PSNR=26.05 dB
d_s=50

Image codée avec r =0.53 bpp
PSNR=24.89
d_s=1500

Figure V.8 *Images codées avec un codeur Cache VQ avec la stratégie LFU Avec taille du cache =128*

Figure V.9 *a) Image originale b) Image codée par un codeur Cache VQ avec codebook conçu par l'image Lenna*

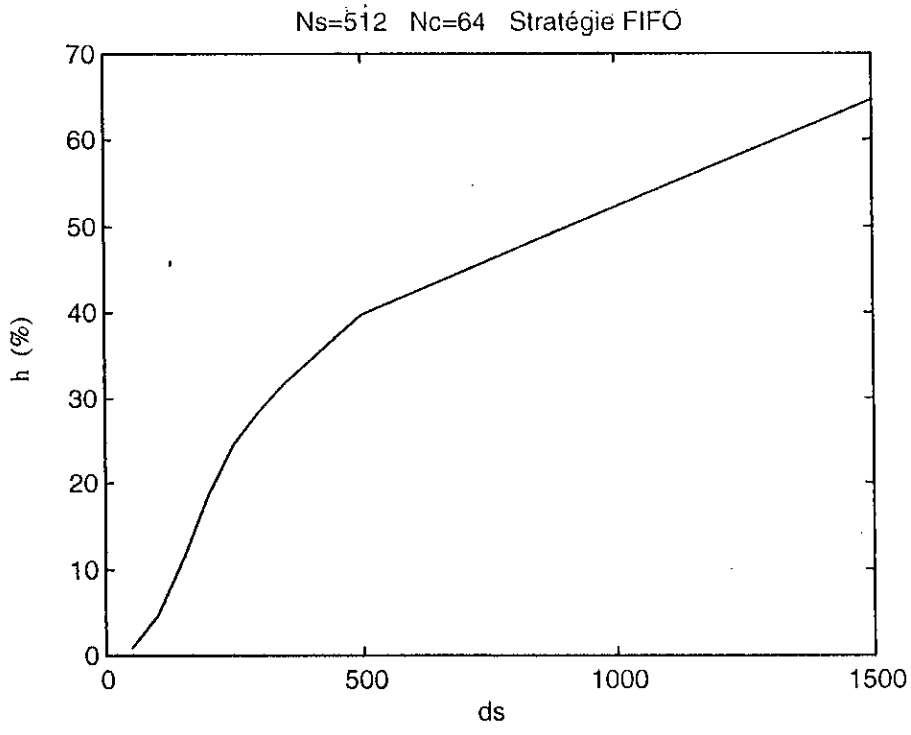


Figure V.10.a *h en fonction de d_s*
Stratégie FIFO - N_c=64

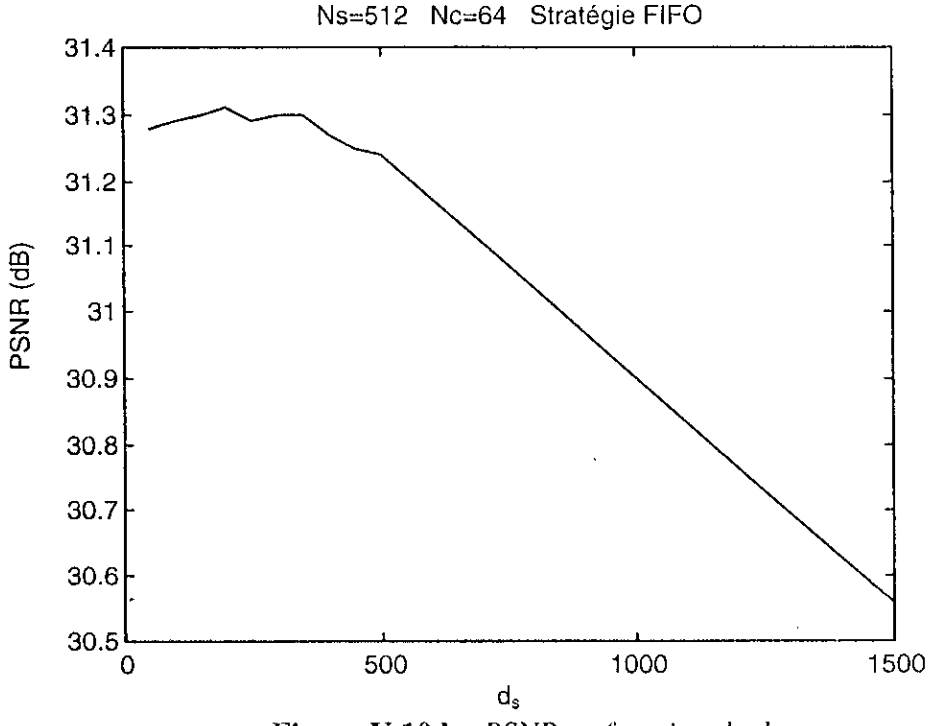


Figure V.10.b *PSNR en fonction de d_s*
Stratégie FIFO – Nc=64

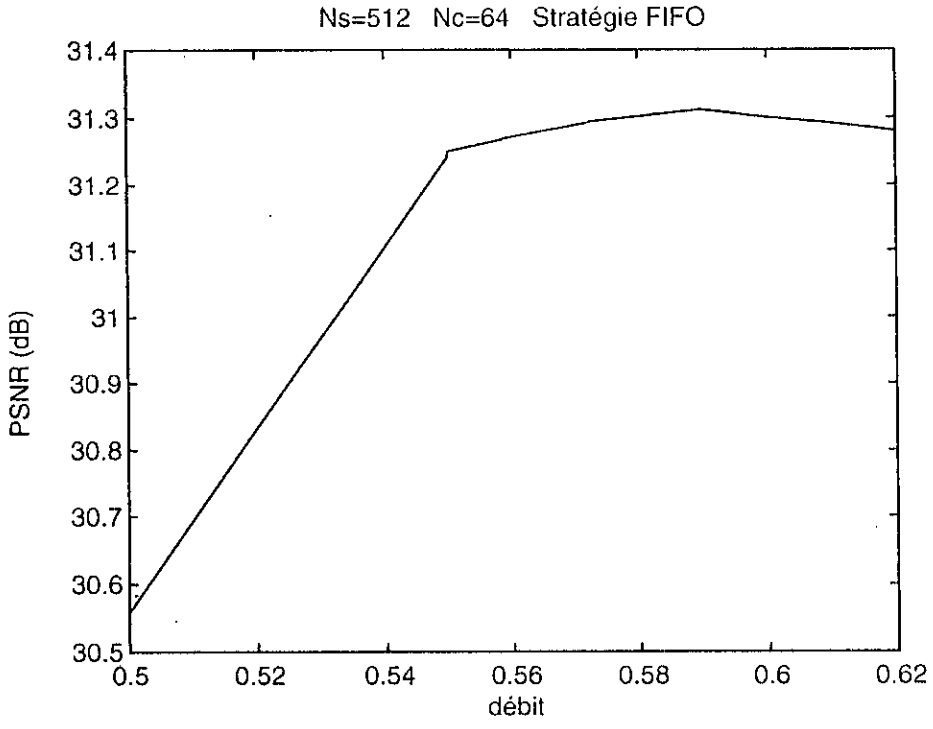


Figure V.10.c *PSNR en fonction du débit*
Stratégie FIFO- Nc=64

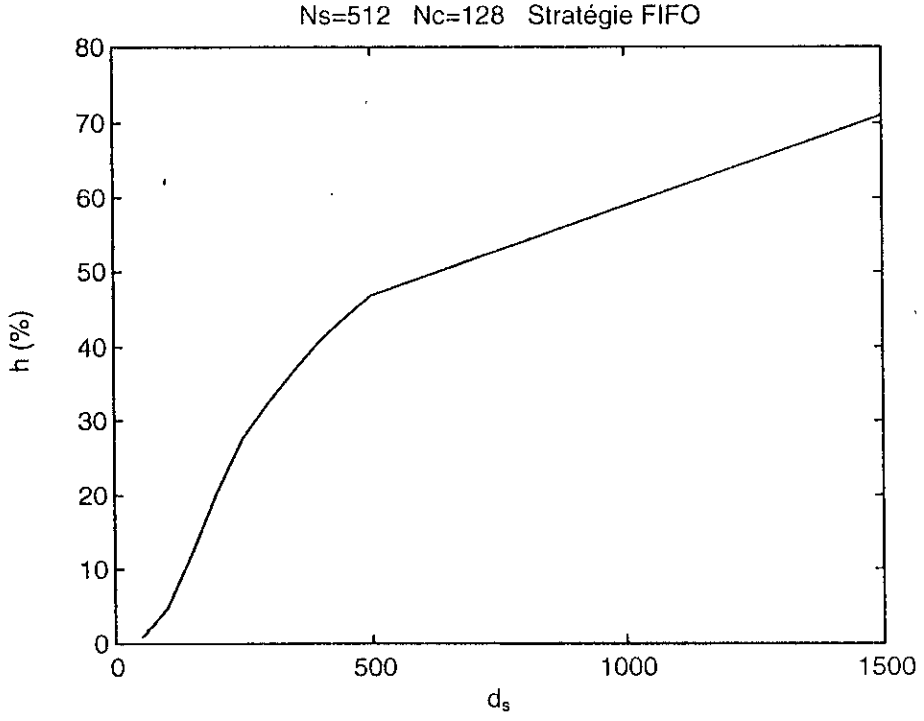


Figure V.10.d *h en fonction de d_s*
Stratégie FIFO - N_c=128

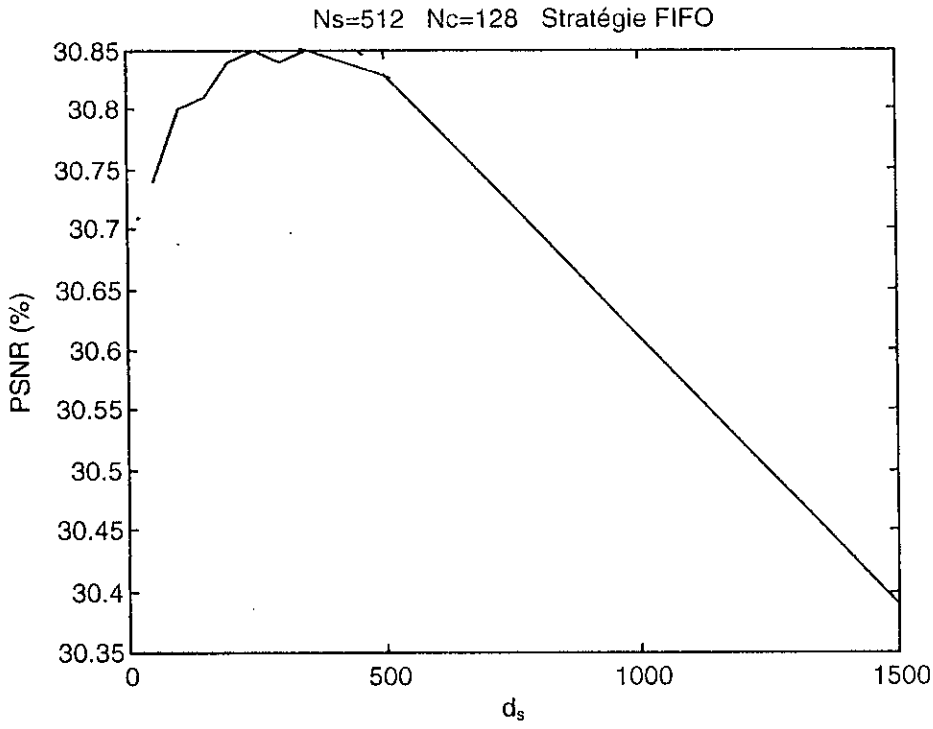


Figure V.10.e *PSNR en fonction de d_s*
Stratégie FIFO - N_c=128

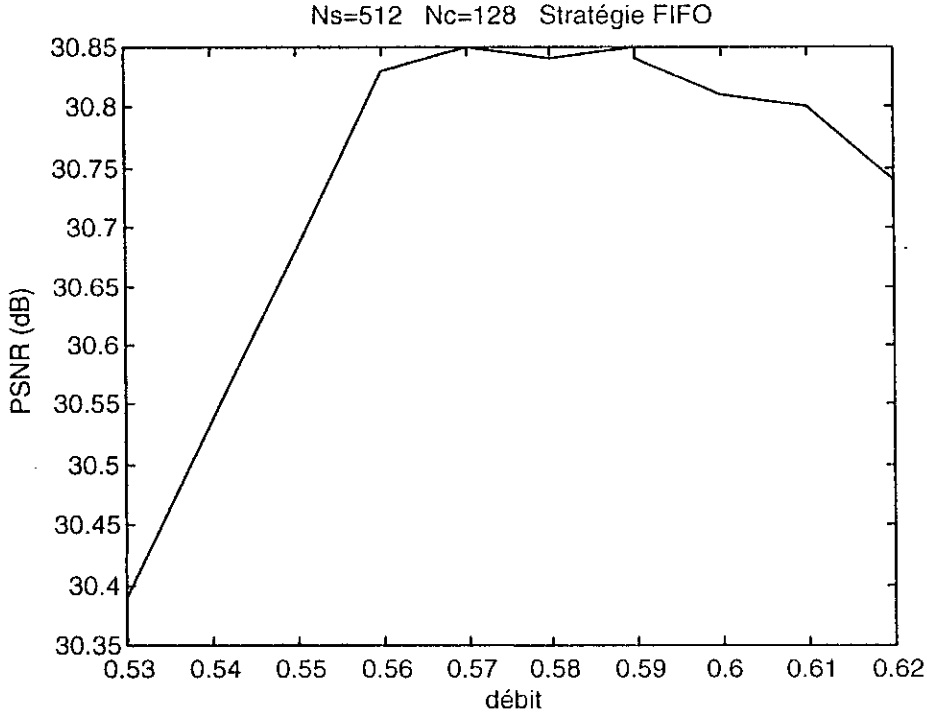


Figure V.10.f *PSNR en fonction du débit*
Stratégie FIFO - N_c=128

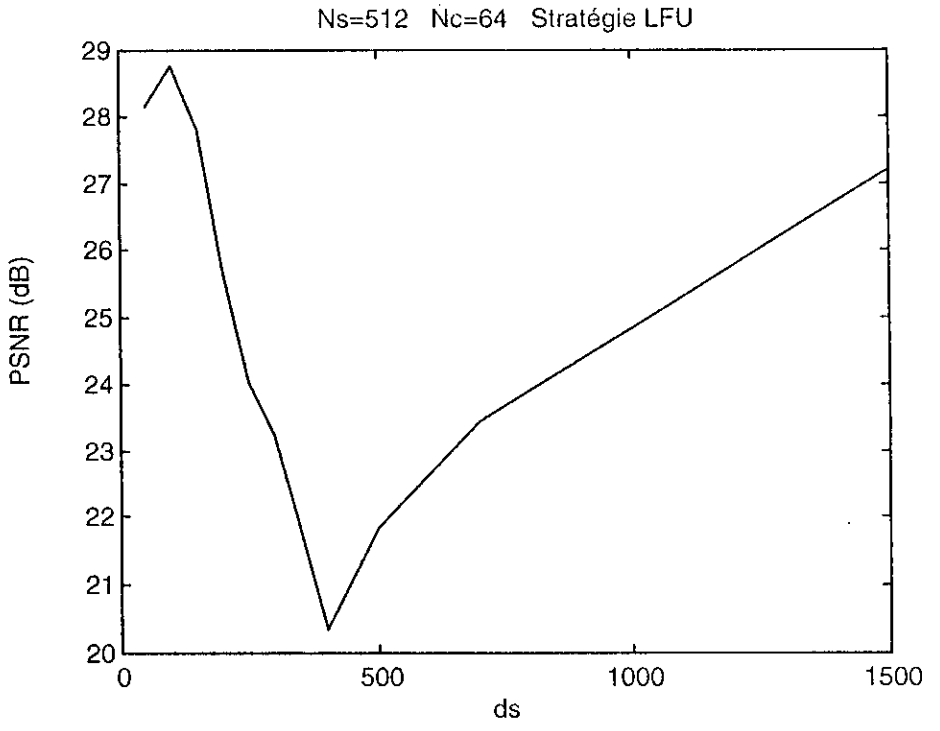


Figure V.11.a *PSNR en fonction de d_s*
Stratégie LFU - N_c=64

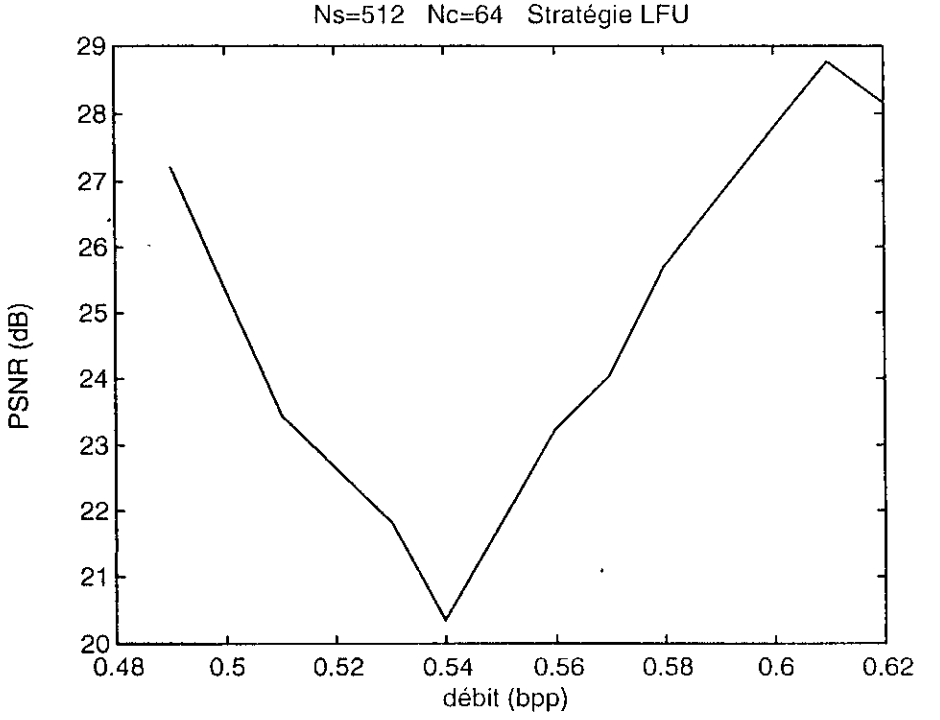


Figure V.11.b *h en fonction du débit*
Stratégie LFU - N_c=64

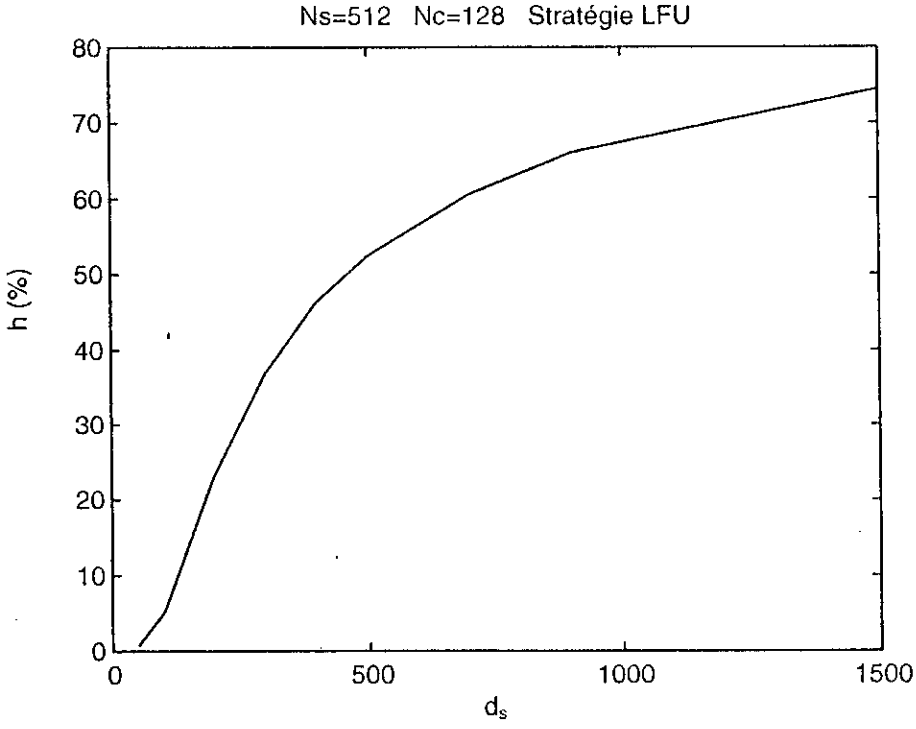


Figure V.11.c *h en fonction du d_s*
Stratégie LFU - N_c=128

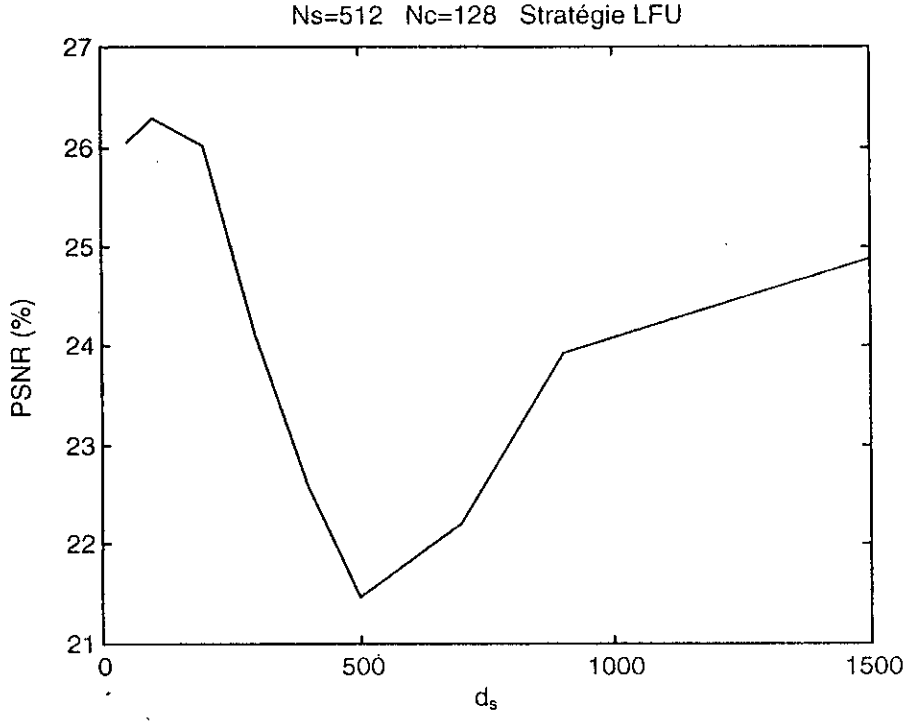


Figure V.11.d *PSNR en fonction de d_s*
Stratégie LFU - N_c=128

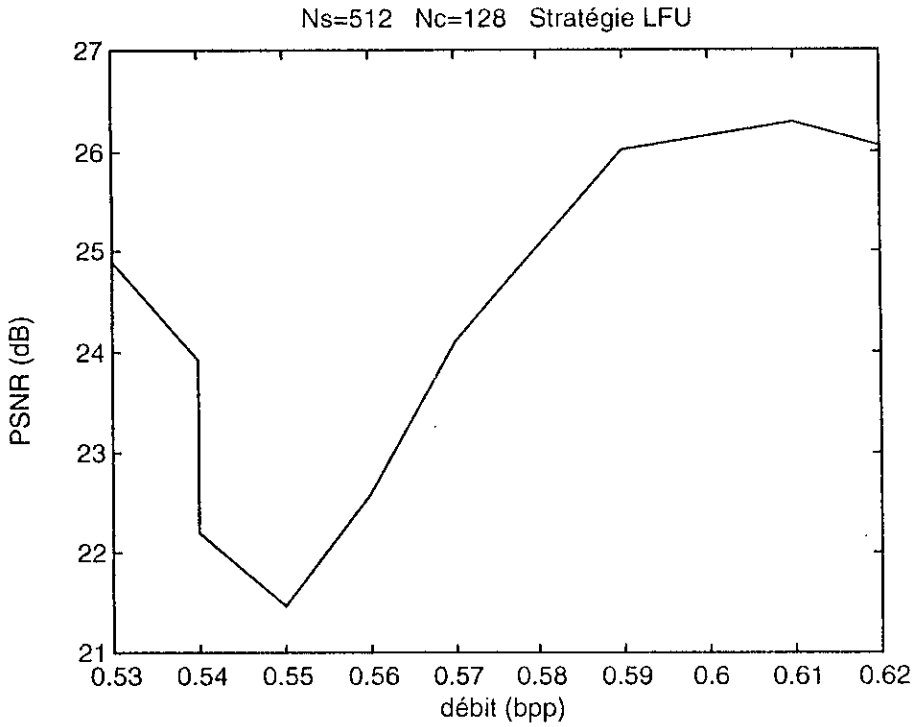


Figure V.11.e PSNR *en fonction du débit*
Stratégie LFU - N_c=128

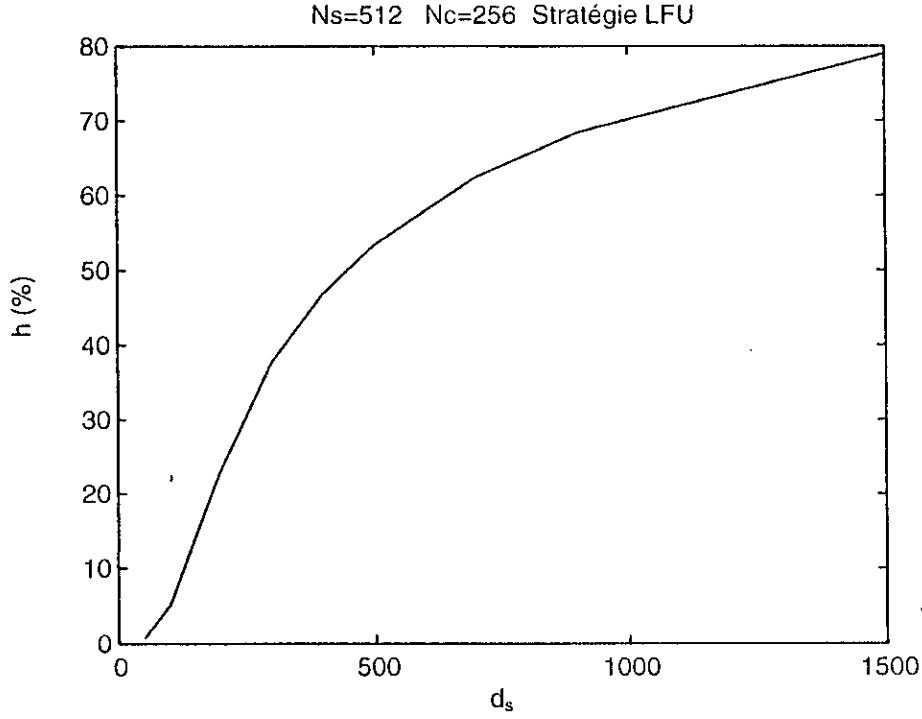


Figure V.11.f PSNR *en fonction de d_s*
Stratégie LFU - N_c=256

V.1.3.1 Analyse des courbes

On a illustré les variations des paramètres d'un quantificateur cache par les courbes de la figure V.10.a à la figureV.11.f. Deux catégories de courbes sont tracées ; les courbes où la stratégie FIFO est utilisée, et les courbes où la stratégie LFU est utilisée.

***h* en fonction de d_s**

Dans les courbes des figures V10.a, V10.d, V11.c et V.11.f, on remarque que plus d_s augmente plus *h* augmente, cela est prévisible parce que lorsque d_s augmente cela veut dire que la stratégie devient moins exigeante, et par la suite le cache devient plus utilisé, mais les valeurs de h pour la stratégie LFU son plus élevées que ceux de la stratégie FIFO. D'un autre côté on remarque un gain dans le temps de codage intéressant, parce que le temps de codage devient le temps de recherche dans le cache codebook plus le temps de mise à jour de celui ci, avec une stratégie de mise à jour efficace on arrive à un temps de codage meilleur.

PSNR en fonction du débit r

Le PSNR en fonction du débit est représenté dans les figures V.10.c, V.10.f, V.11.b et V.11.e. On remarque dans les deux premières figures que lorsque le débit augmente les qualités augmentent (PSNR), cela est connu par la fonction RDF, voir figure I.1, car le PSNR est l'inverse de la distorsion, équations 2.6 et 2.7.($PSNR \propto log_2(\frac{1}{D})$) . Pour les figures V.11.e et V.11.b on remarque qu'à partir d'une certaine valeur de r le comportement du PSNR est le même que dans le cas de la stratégie FIFO, mais au-dessous de cette valeur le PSNR est élevé pour des valeurs faibles du débit, mais il est limité. Cela est dû au fait que le PSNR et la distorsion D sont reliés par une relation logarithmique.
On voit aussi que le PSNR dans la stratégie FIFO est nettement plus élevé que celui de la stratégie LFU,voir les images des figures V.4, V.5, V.6, V.7, V.8.

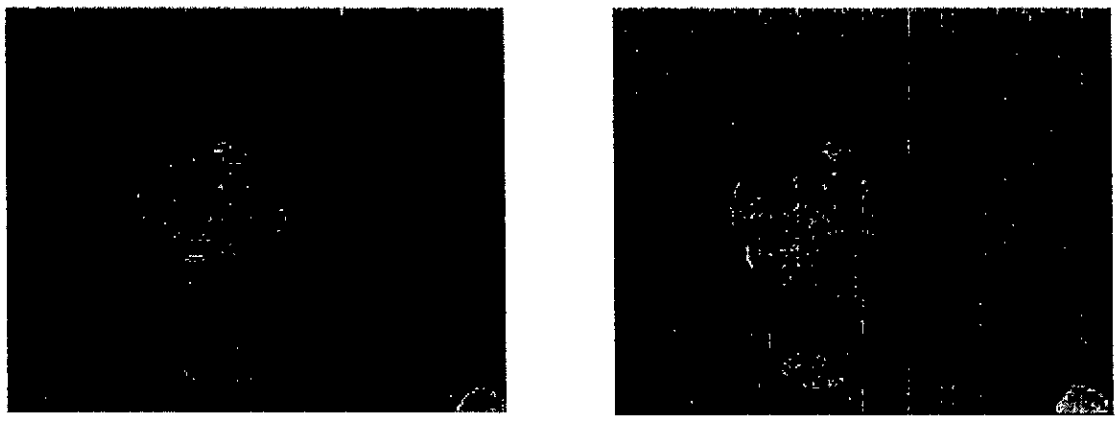
PSNR en fonction de d_s

Les figures V10.b, V10.e, V11.a et V.11.d, montrent les variations du PSNR en fonction de d_s. On remarque que pour les valeurs élevées de d_s le PSNR chute dans la stratégie FIFO, cela aussi est prévisible car le cache est très sollicité quand d_s augmente, d'où le choix très limité de vecteurs. Par contre pour la stratégie LFU on remarque que le PSNR diminue ensuite à partir d'une certaine valeur de d_s il commence à augmenter. Comme le débit est inversement proportionnel avec d_s, alors les variations du PSNR en fonction de ds sont l'inverse de celles du PSNR en fonction du débit. D'un autre côté on remarque une qualité meilleure (PSNR) avec des caches de faible taille (N_c).

V.2 Codage interframe

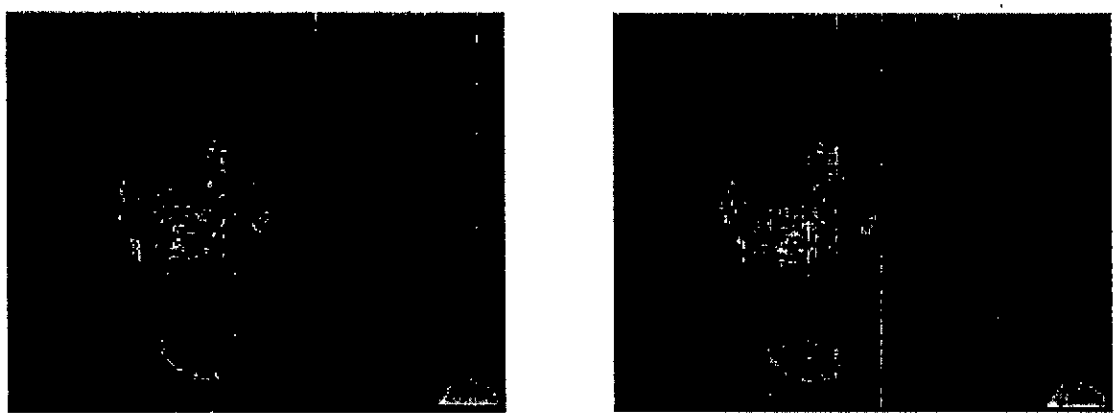
V.2.1 Détection de mouvement

Dans cette partie on s'est intéressé à étudier quelques détecteurs de mouvement tel que la technique de correspondance de blocs BM, sa version rapide la FBM et le modèle affine. La technique de BM consiste à diviser la frame F_k en blocs carrés de taille bxb, dans notre cas b = 4, i.e. 4x4, et chercher pour chaque bloc de F_k un bloc lui ressemblant dans une région R de F_{k-1}, cette région est de dimension (b+2w)x(b+2w). Dans la comparaison on a utilisé la distance MAE pour alléger les calculs. Les images suivantes sont obtenues pour w=7 et w=9 figures V.12.a à V.12.d.



a)Frame #29 b)Frame #30

Figure V.12.a *La séquence Miss America (originale)*
a)La Frame #29 b) La Frame #30



a) w=5 PSNR=30.41dB *b) w=7 PSNR=30.75 dB*

Figure V.12.b *La frame #30 reconstruite par détection de mouvement par l'algorithme BM*

a) w=5 PSNR=29.15 dB

b) w=7 PSNR=29.43 dB

Figure V.12.c *La frame #30 reconstruite par détection de mouvement par l'algorithme BM rapide*

Figure V.12.d *La frame #30 reconstruite par détection de mouvement par l'algorithme du modèle affine PSNR=31.03 dB*

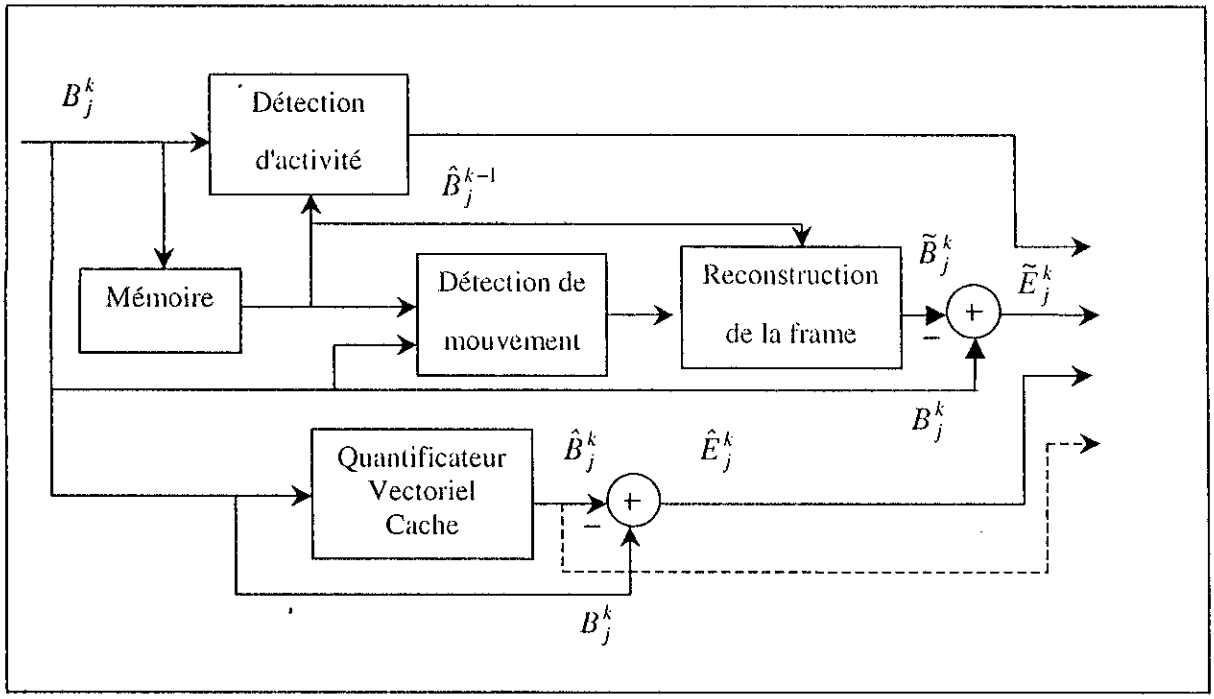


Figure V.13 *Codage Intraframe-Interframe*

V.2.2 Codage intraframe-interframe

Dans la figure V.12 les deux types de codage sont combinés, intraframe et interframe. La première frame est quantifiée et envoyée au décodeur, elle sert comme frame de référence, cela est représenté dans la figure par un trait en pointillé. Par la suite on procède à une détection d'activité dans chaque bloc des frames suivantes. La détection d'activité consiste à effectuer une soustraction entre le bloc B_j^k et le bloc B_j^{k-1} et la différence est comparée à un seuil, si la différence est inférieure au seuil on dit que le bloc est inchangé et par la suite un bit est envoyé au décodeur lui indiquant qu'il faut garder l'ancien bloc. Sinon, on procède à une détection de mouvement d'un côté et à une VQ d'un autre côté, et on compare les blocs reproduits par les deux modules et on envoie le code de la meilleure reproduction, en envoyant des bits d'information au décodeur pour qu'il puisse reproduire le vecteur ou le bloc. On a utilisé la technique de BM pour la détection du vecteur mouvement sur une région de 10x10, d'où d_x sera codé sur 3 bits , idem pour d_y, par la suite le vecteur de déplacement sera codé sur 6 bits.

L'image de la figure V.14 est la reproduction de la frame #30, par le codeur de la figure V.13. Elle est codée sur 0.29 bpp, tel que :

38 % de blocs est inchangé,

29 % de blocs compensés,

33% de blocs quantifiés.

Figure V.14 *La frame #30 codée avec 0.29 bpp à PSNR = 29.04*

Conclusion Générale

Le travail qui nous a été confié consiste à simuler un système de compression d'une séquence d'images en vue d'un stockage sur un support d'information, ou une transmission à travers un réseau de communication. Dans la réalisation de cet objectif, on a travaillé dans deux sens : Un codage intraframe et un codage interframe. Dans le codage intraframe, c'est la corrélation spatiale qui est exploitée pour réduire le débit. Dans ce type de codage on trouve une panoplie de techniques, on a opté pour la quantification vectorielle, d'un côté pour le taux de compression très intéressant qu'elle permet, et d'un autre côté vu le grand nombre des travaux qui ont été réalisés dans ce domaine et qui ont donné des résultats très satisfaisants.

Dans notre travail on a utilisé une technique dite cache VQ, cette technique est classée parmi les techniques adaptatives, elle utilise un codebook large qui se rapproche le plus possible des statistiques de l'image, associé avec un autre codebook dit cache codebook de taille faible par rapport au premier. Le cache codebook contient les vecteurs les plus utilisés, et il est tout le temps mis à jour suivant une certaine stratégie. Cette technique nous a permis d'atteindre des débits faibles atteignant 0.6 à 0.5 bpp. On a remarqué que lorsque d_s augmente le débit diminue et h augmente cela est prévisible, car lorsque d_s augmente le cache codebook sera très sollicité par la suite les vecteurs seront codés par des mots binaires plus courts. D'un autre côté on remarque que lorsque la taille du cache diminue le PSNR augmente, et le débit diminue légèrement. On remarque que le PSNR, i.e. la qualité des images est meilleure dans la stratégie FIFO que dans la stratégie LFU.

On a pu conclure d'après les résultats qu'avec la VQ les notions de codage, compression et filtrage sont confondues. La VQ effectue un codage car un vecteur est représenté par un certain nombre de bits, et elle effectue une compression parce qu'un ensemble de k pixels, qui nécessitaient k mots pour être représentés, seront représentés par un mot, par exemple 16 pixels seront représentés par un octet. Elle consiste en un filtrage car les vecteurs de reproduction sont issus d'un algorithme de conception de codebook, tel que LGB, qui calcule ces vecteurs en effectuant un moyennage sur les partitions (barycentre). Avec une stratégie de mise à jour efficace on peut réduire le débit et le temps de codage de plus en plus tout en gardant une qualité acceptable.

Dans le codage interframe on a utilisé une détection et compensation de mouvement. Dans la détection on a essayé l'algorithme BM, qui a donné des résultats très appréciables mais il est coûteux en temps de calcul. La version rapide de cet algorithme permet de contourner le problème de temps, mais une dégradation faible est remarquée dans la qualité

des images. En ce qui concerne le modèle affine, il donne des résultats intéressants mais le temps de calcul est très important, son utilisation directe dans un système fonctionnant en temps réel n'est pas possible. L'intégration d'un module de détection de mouvement nous a permet d'utiliser l'information temporelle, où le débit a chuté jusqu'à 0.29 bpp, et avec un choix convenable et moins exigeant des seuils, on peut atteindre un débit plus faible.

Comme perspectives dans le prolongement de notre travail nous proposons ; la recherche d'une stratégie plus efficace, qui permet de réaliser un compromis entre le débit et la qualité des images, et même l'utilisation de plusieurs caches permettra de palier au problème de discontinuité de blocs. D'un autre côté l'amélioration du modèle affine pour la détection du mouvement en élaborant une version rapide de cet algorithme donnera de meilleurs résultats.

Nous estimons qu'on a atteint des résultats acceptables, et que notre contribution servira de base pour d'autres travaux dans le même domaine.

Annexe A

La norme 4.2.2 de la télévision numérique

La norme 4.2.2 est utilisée dans la Recommandation CCIR :601-1, elle utilise un nombre d'échantillons pour la composante luminance Y double de celui utilisé pour la composante C_b et la composante C_r.

La norme 4.2.2 est une norme d'image de télévision numérique initialement destinée à la production en studio. Elle fut adoptée par le CCIR en 1981.

Ces caractéristiques sont :

- un signal de luminance Y
- deux signaux simultanés de différence de couleur ou chrominance :C_r et C_b (R601)

Fréquence d'échantillonnage :

Luminance..... 13.5MHz

Chrominance .....6.75MHz pour chaque différence de couleur

Fréquence de ligne.... 1525Hz

Structure d'échantillonnage :

Orthogonale avec échantillons coïncidents figure A.1.

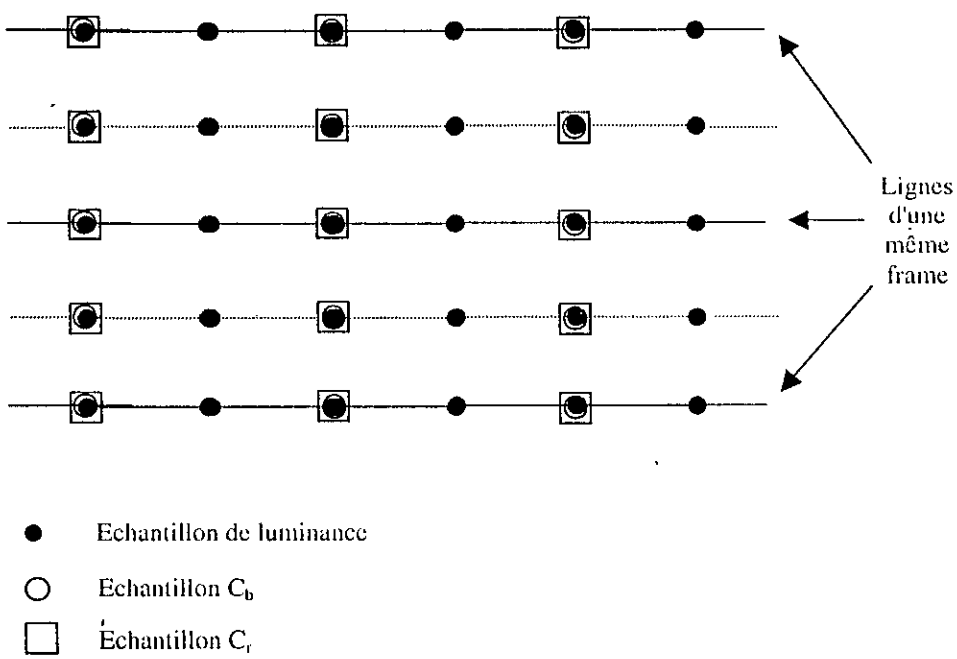


Figure A.1 *Structure d'échantillonnage selon la norme 4.2.2*

Nombre de points par ligne active :

Luminance 720

Chrominance .....360 pour chaque signal

Quantification .....linéaire 8 bits par échantillon pour chacune des composantes

Débit utile.....166Mbits/s

Débit brut total.....216Mbits/s (débit utile + signaux accessoires & synchronisation)

Nombre de lignes actives :

Il y a deux options :

576 la plupart des pays européens et associés,

484 aux USA et pays associés.

Format d'image :.....4/3

Fréquence image :

Il y a deux options:

25 la plupart des pays européens et associés,

30 aux USA et pays associés

Entrelacement d'ordre 2 :

Les lignes de l'image complète sont analysées, puis restituées sur le récepteur, en deux passes. Lors de la première passe on oublie une sur deux, les lignes restantes sont analysées lors de la deuxième passe.

Annexe B

Les standards JPEG et MPEG

B.1 JPEG

JPEG, pour Joint Photographic Experts Group, est un standard de la compression d'image fixe mis au point par le comité ISO. En 1982 ISO a formé l'équipe PEG pour faire des recherches dans le domaine de la transmission des images fixe et le texte à travers le réseau ISDN (*Integrated Services Digital Network*), donc le but du groupe PEG,était de réaliser des standards industriels pour la transmission des images et des graphiques à travers les réseaux de communication digitaux.

En 1986 un autre groupe du comité CCITT a commencé ses recherches dans le domaine de la compression des images couleurs et à niveaux de gris pour un but de transmission (fac-similé). Les méthodes utilisées par ce groupe étaient très similaires à celles utilisées par le groupe PEG. Pour cela, en 1987 les deux comités, ISO et CCITT, ont décidé de combiner leurs groupes en un seul baptisé JPEG .

JPEG n'est pas un algorithme unique on peut le modifier et l'adapter à l'utilisation désirée. Il est conçu pour la compression d'images couleurs et niveaux de gris du monde réel (photos, objets naturels etc.), car il peut représenter un pixel par 24 bits (16 millions couleurs). Le taux de compression du JPEG dépend des images traitées, par exemple une image de qualité photographique peut être compressé de 20:1 jusqu'à 25:1, en gardant une très bonne qualité.

Les étapes de l'algorithme JPEG dans sa version de base sont :

1- Séparer les composantes chrominance et luminance,
2- Sous - échantillonner les composantes chrominance,
3- Appliquer une DCT sur les blocs de l'image,
4- Quantifier les coefficients transformés,
5- Coder les coefficients quantifiés par le codage de Huffman.

Plusieurs versions ont été mis au points par la suite sous forme d'extensions, où chaque version a apporté des améliorations et des perfectionnements. L'extension citée dans Part 1 du standard JPEG a améliorée le taux de compression en utilisant le codage arithmétique et en permettant un décodage progressif de l'image. Une autre extension citée dans Part 3 utilise une quantification adaptative des coefficients transformés. Dans cette extension les éléments de la matrice de quantification sont modifiés et mis à jour pour chaque bloc.

B2. MPEG

Pour Motion Picture Experts Group, qui est un groupe travaillant pour le comité ISO, son objectif était la création d'un standard de compression des images numériques et le son. MPEG est conçu pour stocker le son et les séquences vidéo sur disque compact (CDROM), et sur les bandes audio digitales (DAT), alors que MPEG-II permet la transmission des séquences à travers la télévision et les réseaux de communication.

Pour stocker les données sous le format MPEG, il faut un matériel d'acquisition d'images en temps réel avec un débit de 30 images par seconde. Chaque image capturée sera compressée et stockée. Une autre partie se charge par l'acquisition du son et sa compression, puis le multiplexer avec les données vidéo, et enfin ajouter des informations de synchronisation des deux types de données (vidéo et audio).

Le standard MPEG utilise une compression asymétrique, où le codage des données est plus compliqué que le décodage. Dans la compression du signal vidéo, il utilise un codage intraframe-interframe . Le codage interframe se base sur des techniques de prédiction et d'interpolation. MPEG utilise une prédiction bidirectionnelle, i.e. il utilise l'image précédente et l'image prochaine pour prédire l'image courante. Pour cela le standard MPEG répartie les frames en I-frame (codage interframe), P-frame (codage prédictif), B-frame (codage prédictif bidirectionnel), figure B.2. I-frame est une frame complète qui ne dépend d'aucune autres frame. Une P-frame résulte de la prédiction de la différence entre la frame courante et une I-frame ou P-frame la plus proche. Une B-frame est construite en utilisant les deux I-frame ou P-frame les plus proche. Une séquence MPEG type est donnée par :

IBBPBBPBBPBBIBBPBBPBBPBBIBBPBBPBBPBBI

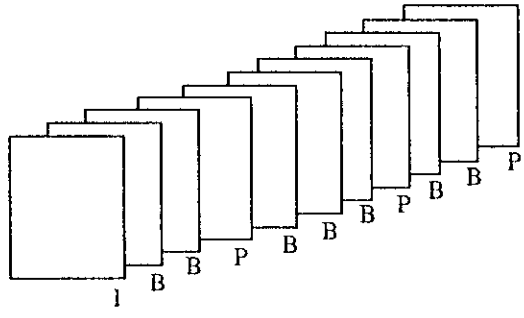


Figure B.2 *Les trois types de frames utilisées par MPEG*

Annexe C

Espace vectoriel des blocs d'image de taille M x N

Soit $\Im$ l'ensemble des blocs d'image dont :

- La taille est M x N où M est le nombre de lignes et N est le nombre de points par ligne.
- Les niveaux de gris de chaque point appartient au corps des complexes **C**.

L'ensemble des images ayant un sens pour le système visuel est un sous ensemble de $\Im$. Dans ce dernier les niveaux de gris appartiennent à $\Re$, l'ensemble des nombre réel. Définissons dans $\Im$:

* Une loi interne, notée " + ", appelée somme, telle que la somme de deux blocs est un bloc dont les niveaux de gris sont la somme, dans **C**, des niveaux de gris se correspondant spatialement.

* Une loi externe, notée " . ", appelée multiplication par un scalaire, telle que la multiplication d'un bloc par $\alpha \in \mathbf{C}$ soit un bloc dont les niveaux de gris sont ceux du bloc initial multiplié par α.

Il est possible de définir un isomorphisme entre $\Im$ et l'ensemble des blocs de taille M x N. Nous pourrons donc symboliser un bloc B à l'aide d'une notation vectorielle telle que $\overline{B}$.

Notons que :

- La définition de l'égalité entre deux blocs est triviale.
- La dimension de $\Im$ est M x N
- Il existe un ensemble de M x N blocs pouvant servir de base à tous les autres
- La base canonique est l'ensemble des blocs ayant tous les points noirs sauf un qui possède une valeur égale à un.

Un Bloc $\overline{B}$ s'écrit dans la base canonique $\overline{B}_1, \overline{B}_2, \overline{B}_3, \overline{B}_4$ comme :

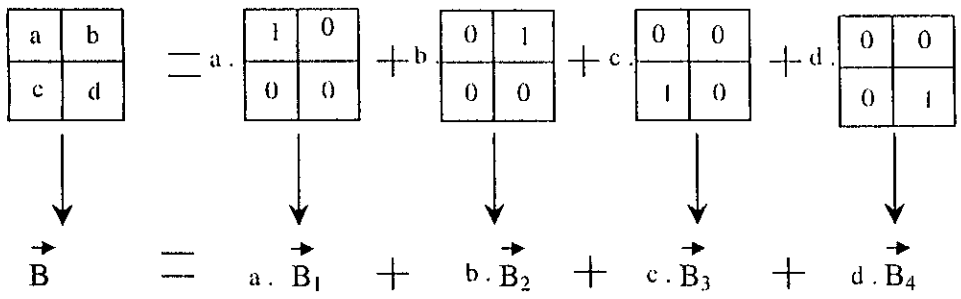


Figure C.1 *Génération d'un bloc dans la base canonique*

Bibliographie

[1] A.Gersho, R.M.Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers 1992, sixth printing 1997

[2] A.Gersho and B.Ramamurthi " Image Coding using Segmented Codebooks", in Proc IEEE Int Conf ASSP Apr 1982 pp428-431

[3] K.K.Truong and R.M.Mersereau "Variable Rate Vector Quantizer for Images based on Principle of Cache Memory Design" , IEEE Trans Image Processing, 1993

[4] A.Haoui and D.Messershmitt "Predictive Vector Quantization ", in Proc IEEE Int Conf ASSP May 1984. pp10.10.1-10.10.4

[5] N.M.Nasrabadi, R.A.King "Image Coding Using Vector Quantization : A review", IEEE Trans , Comm vol com 36.pp 971 Aug 1988

[6] M.Vetterli ," Multirate Filter Banks for Subband Coding" in Subband Image Coding (J.W.Wood,ed)Boston Kluwer 1990 pp 43-100

[7] C.S.Fuh, P.Maragos "Affine Model For Image Matching and Motion Detection ", in Proc, IEEE, Conf Image Processing 1990, pp 2409-2412

[8] M.Ibrahim Sezan, Reginald L.Lagendijk, Motion Analysis and Image Sequence, Kluwer Academic Publishers 1993

[9] T.Doyle "Interlaced to Sequential Conversion for EDTV Application " in Proceedings of second International Workshop on HDTV l'Aqvila Italy 1988

[10] Petri Haavisto, Yrjö Neuvo " Motion Adaptive Scan Signals, Kluwer Academics Publishers 1992, pp 5-22

[11] O.Yli Harja, J.Astola, and Y.Neuvo "Analysis of the Properties of Median and Weighted Median Filters using Threshold Logic and Stack Filter Representation" IEEE Trans on Signal Processing vol sp 39 Fab 1991 pp 395-410

[12] R.Srinivasam and K.R.Roo " Predictive Coding based on Efficient Motion Estimation " IEEE Trans on Comm vol com Nov Aug 85

[13] W.Equits, " A New Vector Quantizer Clustering Algorithm." IEEE Trans, ASSP vol 37 pp1568-1575 Oct 1989

[14] Y.Linde, A.Buzo, R.M.Gray " An Algorithm for Vector Quantizer Design " IEEE Trans Comm vol com 28 pp84-95 Jan 1980

[15] J.Shanbehzadeh and P.O.Ogvibona " On the Computational Complexity of the LBG and PNN Algorithm", IEEE Image Processing 1997

[16] C.Raimondo, C.Galand, E.Goubant, E.Lançon and J.Menes," Low Bit Rate Hybrid Coder Using Hierarchical Motion Compensation and Low Complexity Vector Quantization ", Int Conf on Image Processing 1990 M9.2

[17] J.Salinas and R.L.Baker," A Motion Compensated VQ with Filtered Prediction ", Proc, ICASSP Apr 1988

[18] M.Goldberg and H.Sun ," Image Sequence Coding Using Vector Quantization ", IEEE Trans Comm vol 34 pp703-710 Jul 1986

[19] T.Saito, H.Takeo, K.Aizawa, H.Harshima and H.Miyahana " Adaptive Discrete Cosine Transform Image Coding Using Gain/Shape Vector Quantization " in Pro, IEEE Int, Conf ASSP Apr 1986 pp 145-147

[20] D.S.Kim, T.Kim and S.U.Lee "On Testing Training Vector Quantizer Codebook",IEEE Transactions on Image Processing Vol 6,No 3, Marsh 1997

[21] J.Foster, R.M.Gray and M.O.Dunham "Finite-State Quantization for Waveform Coding", IEEE Transaction on Information Theory, Vol IT-31,No 3, May 1985

[22] André Davignon "Classification en Blocs de Taille Variable pour Codage d'Image par Quantification Vectorielle" Traitement du signal , volume 6-n°4-1989

[23] Jean-Pierre Adoul" La Quantification Vectorielle des Signaux : Approche Aalgébrique" Annales Télécomunication.,41,n° 3-4, 1986 pp 158-177

[24] Jean-Paul Guillois , Techniques de Compression des Images ,Edition Hermés 1996(Collection Informatique)

[25] N.Ahmed, T.Natarajan et K.R.Rao," Discrete Cosine Transform", IEEE. Trans. Computers., vol. C-23, N° 1, January 1974, pp. 90-93

[26] J.P.Leduc, J.M.Odobez and C.Labit "Adaptive Motion-Compensated Wavelet Filtering for Image Sequence Coding", IEEE Transaction on Image Processing, vol .,6, No ,6, June 1997

[27] H.H.Shen and R.L.Baker"A Finite State Difference Interpolative Vector Quantizer for Low Rate Image Sequence Coding" IEEE Int. Conf . Acoust. Speech. Signal Processing vol 2,pp 1188-1191, April 1988

[28] Giovani L.Sicuranza, Sanjit K.Mitra, " Multidimentional Processing of Video Signals", Kluwer Academic Publishers, 1992 USA.

[29] N.S.Jayant, Peter Noll, Digital Coding of Waveforms, Principles and Applications to Speech and Video, Prentice-Hall, INC 1983 USA.

[30] Rafae C.Gonzalez, Paul Wintz, Digital Image Processing, Addison Wesley Publishing Company, INC 1987 USA

[31] M.Nelson," La Compression de Données : Texte-Images-Sons, Dunod Paris 1993

[32] James D.Murray and William VanRyper, Encyclopaedia of Graphics Fila Formats, O'Reilly &Associates, INC Second Edition 1996

[33] R.L.Baker and R.M.Gray,"Image Compression Using Non-Adaptive Spatial Vector Quantization", IEEE, Circuits, Systems, Computer, Oct. 1982.

[34] A.Gersho and B.Ramamorthi," Image Coding Using Vector Quantization", IEEE.Conf, ASSP, vol. 1, pp 428-431, May 1982

[35] T.Murakami, K.Asai and E.Yamazaki ," Vector Quantizer of Video Signals", Electronics Letters, vol. 7, pp. 1005-1006, Nov. 1982.

[36] B.Ramamurthi and A.Gersho, " Image Vector Quantization with a Perceptually based Cell Classifier", IEEE Conf, ASSP, vol. 2, pp. 32.10.1-32.10.4, March 1984 .

[37] K.S.Thyagarajan, S.Parthasarathy and H.Abut," A Matrix Quantizer Incorporating the Human Visual Model", IEEE, Conf, ASSP. vol. 1, pp 141-144, March 1985.

[38] V.Ramamoorthy and N.S.Jayant," High Quanlity Image Coding With a Model –Testing Vector Quantizer and a Human Visual System Model", IEEE, Conf, ASSP, vol. 2, pp, 1167, April 1988.

[39] T.Murakami, K.Asar and A.Itch, " Vector Quantization of Color Image", IEEE, Conf , ASSP, vol. 1, pp. 133-136, April 1986.

[40] M.Goldberg, P.R.Boucher and S.Shlien, "Image Compression Using Adaptive Vector Quantization", IEEE, Trans. Commun., vol. COM 34, pp.180-187, Feb 1986.

[41] K.Aizawa, H.Harashima and H.Myakawa, " Adaptive Discrete Csine Transform Coding With Vector Quantization for Color Image ", IEEE, ICASSP, vol. 2, pp. 985-988, April 1986.

[42] C.W.Rutledge, "Variable Block Vector DPCM: Vector Predictive Coding of Color Images", IEEE, vol. 1, pp. 130-135, June 1987.

[43] K.S.Thyagarajan and S.Bhatt, " Linear Block Prediction with Source Classification for Images Encoding Applications ", IEEE, vol. 2, pp. 1308-1311, April 1988.

[44] K.S. Thyagarajan, "Image Coding Based on Segmentation using Region Growing", IEEE, Conf, ASSP. vol. 2, 752-755. April 1987.

[45] K.N.Ngan, K.S.Leong and H.Singh, "Adaptive Cosine Transform Coding of Images in Perceptual Domain", IEEE, Trans. ASSP, vol. 37, NO. 11 November 1989.

[46] S.Belkacemi, " Codage d'Image Fixe et Animées à Faible Débit", Thése de Magister, ENP, 1996

[47] B.Zekrini and D.Berkani, " Image Sequence Coding Using a Combined VQ with Rapid Motion Compensation", COMAIE'98, Dec. Bejaia. Algerie.

[48] I.Furukawa, M.Nomura and S.Ono," Hierarchical Coding of Super High Definition Image with Subband + Multistage VQ," in Proc. IEEE. Conf.Image. Processing. pp.2637-2640. 1991.

[49] R.J.Clarcke, Transform Coding of Images', Edition Academic Press, 1985.