الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR. YAHIA FARES DE MEDEA

FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

Département du Génie Électrique et de l'Informatique



Projet de Fin d'Etudes

Présenté pour l'obtention du Diplôme
de Master en Génie Électrique

Spécialité : *Systèmes Intelligents et Communicants*

THEME

IMPLEMENTATION DE LA PREDICTION INTRA DE L'ENCODEUR H.264

Proposé et dirigé par,

M. MAAZOUZ Mohamed

M. AGUEB Mohamed

Préparé et présenté par,

M. OULD CHEIKH BOUYA Ahmed Heiba

Remerciements

*Au terme de ce travail, je tiens à remercier en premier lieu «**Allah**» qui nous a donné la force et la volonté pour achever ce travail.*

*Je remercie vivement mon promoteur **M. MAAZOUZ Mohamed** pour son orientation, son aide, sa disponibilité et ces conseils durant mon travail.*

J'adresse également mes remerciements, à tous mes enseignants, qui m'ont donné les bases de la science.

Je remercie du fond du coeur mes chers parents et frères de tout ce qu'ils ont fait pour moi.

Je tiens également à remercier très sincèrement tous les membres de jury, qui font l'honneur de juger notre travail.

Dédicaces

Je dédie ce modeste travail :

A mon père Cheikh Bouya Mohamed Takiyo Allah *et ma mère* **Mariem Abd-Dayem**, *qui ont comblé d'amour et d'affection, qui m'ont toujours encouragé pour achever mes études tout en espérant voir le fruit de leurs sacrifices, qu'Allah les garde pour moi sains et saufs.*

A ma famille

A tous mes amis.

A tous mes enseignants depuis le primaire jusqu'à l'université.

Ahmed Heiba cheikh Bouya

الملخص:

الهدف من هذا العمل هو تطبيق خوارزميات التنبؤ البنيوية الداخلة في نظام ضغط الفيديو وفق المعيار المستخدم **H.264** للشروع في هذا العمل نعمل اولا على ازالة التكرار في مختلف اماكن الصورة . هذا التنبؤ ذو الوحدة البنيوية يتم تنفيذه ايضا يالاضافة الى التحويل ,التكميم ,التكميم المعاكس والتحويل المعاكس . قمنا بتطبيق هذا العمل باستخدام برنامج : **Matlab** يتم حساب **PSNR** عن طريق وسائط مختلفة للتنبؤ بهدف تقييم جودة الصورة التي أعيد بناؤها من جديد
كلمات مفتاحية : ضغط الفيديو , التنبؤالبنيوي ,.PNSR , H.264

Résumé :

L'objectif de ce projet est l'implémentation des algorithmes de compression de vidéo notamment l'intra-prédiction. Pour faire une telle implémentation nous commençons par l'enlèvement des redondances temporel et spatiales dans un encodeur vidéo conformément à la norme H.264. Dans notre travail nous allons implémenter la prédiction intra conjointement avec la transformation, la quantification, la quantification inverse et la transformation inverse sur MATLAB. Le PSNR est calculé pour différents modes de prédiction dont le but est l'évaluation de la qualité de l'image reconstruire.

Mots clés:

Compression vidéo, H.264, intra prédiction, PSNR.

Abstract:

The objective of this project is the implementation of compression algorithms including intra-prediction. This algorithms operate by removing temporal and spatial redundancies in a video encoder conforming to standards such as H.264. Intra Prediction modes improve the compression, exploiting spatial redundancy. In this work, we have implemented Transformation, quantization, Inverse transformation and inverse quantization for modes intra prediction in Matlab. For the reconstructed image PSNR is computed for Different quantization steps. It is observed that the higher value of quantization step degrades the image quality.

Key words:

Compression vidéo, Advanced Video Coding (AVC), H.264, intra prediction, PSNR.

TABLE DES MATIERES

LISTE DES FIGURES ..... **I**
LISTE DES TABLEAUX ..... **III**
LISTE DES ABREVIATIONS ..... **IV**
Introduction générale **1**
CHAPITRE I : PRESENTATION GENERALE DU CODEC H.264 ..... **2**
1. Introduction **3**
2. Historique **3**
3. Le codage vidéo **4**
3.1. Définition **4**
3.1.1. L'espace colorimétrique des pixels 5
3.1.2. Le format YUV 5
3.1.3. Format d'image (première idée de compression) 5
3.2. La compression vidéo **6**
3.2.1. Processus de codage vidéo 7
3.2.1.1. Prédiction *7*
3.2.1.2. Transformation et quantification *7*
3.2.2. Processus de décodage 8
3.2.3. Evaluation de la qualité des images décodées 8
3.3. Les normes de compression vidéo **9**
3.3.1 H.264/AVC ou bien MPEG4/AVC part 10 9
3.3.2. Evolution des normes de compression vidéo et leurs applications 10
4. La nouvelle norme de compression H.264 ..... **11**
4.1. Présentation générale **11**
4.1.1. Division des images en blocs et macroblocs 11
4.1.2. Les critères de distorsion dans H.264 13
4.2. Vue d'ensemble du codec H.264/AVC **13**
4..2.1. Flux de données dans H.264 14
4.2.2. L'encodeur H.264 14
4.2.2.1. Le codage par prédiction dans H.264 15
4.2.2.2. La reconstruction des images 16
4.3. Le décodeur H.264 ..... **17**

4.4. Les profils et les niveaux de H.264/AVC 18
4.5. Applications de H.264 20
4.6. Produits et mises en œuvre de H.264 21
5. Conclusion 22
CHAPITRE II : LA PREDICTION SPATIALE (INTRA) 23
1. Introduction 24
2. Le codage intra -image 24
2.1. Prédiction intra 4x4 25
2.2. Calcul de la prédiction intra 4x4 26
2.2.1. Mode vertical (mode 0) 26
2.2.2. Mode Horizontal (mode 1) 27
2.2.3. Mode DC (mode 2) 27
2.2.4. Diagonal Bas gauche (mode 3) 28
2.2.5. Diagonal Bas Droite (mode 4) 28
2.2.6. Diagonal Vertical droit (mode 5) 29
2.2.7. Diagonal Horizontal bas (mode 6) 29
2.2.8. Diagonal Vertical gauche (mode 7) 30
2.2.9. Diagonal Horizontal haut (mode 8) 30
2.3. Prédiction intra 16x16 31
3. Mode de décision et cout de calcul codage intra 32
4. Transformation et quantification directes et inverses 33
4.1. Développement de la transformée et de la quantification directes 34
4.2. Développent de la quantification et de la transformée inverses 34
4.3. Elaboration des matrices de la transformée directe 35
5. Prédiction du mode de prédiction 37
6. Conclusion 38
CHAPITRE III : IMPLEMENTATION ET RESULTATS..... 40
1. Introduction 41
2. Outil de développement 41
3. Prétraitement 41
3.1. Lecture d'image 43

3.2. Conversion de couleur (RGB ver YUV) 43
3.3. Sous-échantillonnage sure U et V 43
3.4. Construction des macro-blocs (MB) 44
4. Prédiction intra 44
5. Prédiction du mode de prédiction 51
6. Tests et résultats 53
7. Conclusion 61
Conclusion générale 62
Bibliographies 63
ANNEXES 66
1. Codes source de l'implémentation: 66
2. Images de test: 82

LISTE DES FIGURES

Figure I.1.*Chaine direct et inverse d'un codec vidéo* 7
Figure I.2.*Evolution des standards de compression vidéo* 10
Figure I.3. *Organisation des images dans une séquence vidéoH.264* 12
Figure I.4. *Les macroblocs 16x16 et 8x8 dans H.264* 13
Figure I.5. *Le diagramme de l'encodeur H264.* 15
Figure I.6 . *Architecture de décodeur H264.* 18
Figure I.7. *Les versions de H.264 et les outils correspondants.* 19
Figure I.8. *Les transpositions logicielles de H.264.* 22
Figure II. 1. *Prédiction Intra4×4 à l'aide des pixels voisins dans H.264.* 25
Figure II. 2 . *Les 9 modes de prédiction Intra4×4 dans H.264* 26
Figure II. 3. *Prédiction (mode 0) Mode vertical.* 26
Figure II. 4. *Prédiction (mode 1) Mode Horizontal.* 27
Figure II. 5. *Prédiction (mode 2) Mode DC.* 27
Figure II. 6. *Prédiction (mode 3) Diagonal Bas gauche*. 28
Figure II. 7. *Prédiction (mode 4) Diagonal Bas Droite.* 29
Figure II. 8. *Prédiction (mode 5) Diagonal Vertical droit.* 29
Figure II. 9: *Prédiction (mode 6) Diagonal Horizontal bas.* 30
Figure II. 10.*Prédiction (mode 7) Diagonal Vertical gauche..* 30
Figure II. 11. *Prédiction (mode 8) Diagonal Horizontal haut.* 31
Figure II. 12. *Les 4 modes de prédiction intra16×16 dans H.264.* 31
Figure II. 13. *Développement de la transformée et la quantification directes.* 34
Figure II. 14 : *Développent de la transformée et la quantification inverses.* 35
Figure II. 15: *Intra blocs 4x4 codés adjacents.* 38
Figure III. 1. *Organigramme de Prétraitement.* 42
Figure III. 2. *Organigramme de prédiction.* 45
Figure III. 3. *Bloc initial de taille 16x16* 46
Figure III. 4. *La prédiction obtenue pour les quatre modes (16x16).* 47
Figure III. 5. *Bloc initial de taille 4x4* 48
Figure III. 6. *La prédiction obtenue pour les neuf modes (4x4).* 50

Figure III. 7. *Les blocs adjacents* 51
Figure III. 8. *Organigramme de prédiction de mode prédiction* 52
Figure III. 9. *Image originale* 53
Figure III. 10 . *Image des blocs de prédiction (4x4)* 54
Figure III. 11. *Image résiduelle des blocs 4x4* 55
Figure III. 12. *Bloc résiduel (4x4) après cycle T,Q,Q-1,T-1.* 56
Figure III. 13. *Image reconstruite* 57
Figure III. 14.*image predite des blocs (16x16)* 58
Figure III. 15. *Image résiduelle des blocs (16x16).* 59
Figure III. 16. *Image reconstruite des blocs (16x16)* 60

LISTE DES TABLEAUX

Tableau I.1. *Les normes de codage vidéo et leurs applications.* 10
Tableau I. 2. *Les trois profils de la norme H.264.*.....19
Tableau III. 1. *Résultats de test* 61

LISTE DES ABREVIATIONS

AVC : Advanced Video Coding

DCT : Discrete Cosine Transform.

GOP : Group of Pictures

ITU-T : International Telecommunication Union

MPEG : Moving : Picture Experts Group

MSE : Mean Square Error

PSNR : Peak Signal to Noise Ratio

NAL : Network Abstraction Layer

SAD :Sum of Absolute Différences

Introduction générale

De nos jours, l'avènement de la vidéo numérique a été rendu possible principalement grâce aux possibilités de compression. En effet, la compression vidéo permet de réduire le débit nécessaire à la diffusion d'une émission, et facilite donc son transport sur tout type de réseau (hertzien, internet, satellite, téléphone mobile, ...). Le coût de diffusion ainsi diminué permet d'envisager une meilleure qualité de service. L'évolution des standards de compression vidéo aboutit à l'amélioration de l'efficacité de codage. Au début des années 2000, une inflexion s'est produite. A la suite des recherches menées par différents organismes de normalisation, une nouvelle technologie de compression des données vidéo (H264) était développée. Un défi très important est donc à relever du côté de l'impact des vidéos sur le réseau mondial. En termes de type de contenu et de modes de consommation, la diversité croît elle-aussi de façon significative, notamment avec le développement de la vidéo en relief d'une part et la généralisation de la HD (*High Definition*) d'autre part.

H.264 est une norme d'industrie pour la compression vidéo, le processus pour convertir la vidéo numérique en format qui prend moins de capacité quand elle est stockée ou transmise. H.264 offre un excédent significatif d'amélioration d'exécution des normes visuelles précédentes de codage en termes de meilleur rapport signal/bruit et qualité visuelle.

La prédiction intra est une technique parmi les techniques les plus importantes qui contribuent aux succès de H.264/AVC.

Dans ce travail, nous avons implémenté la prédiction spatiale (intra) plus quelques blocs de l'encodeur H.264 comme la transformation, la quantification, la quantification inverse, la transformation inverse. À cet effet, notre mémoire est structuré en trois chapitres, le premier chapitre est réservé à la description générale du codage vidéo et en particulier la norme H.264. Un deuxième chapitre est consacré à présentation détaillée de la prédiction spatiale intra. Le troisième et dernier chapitre, concerne la partie implémentation des différents algorithmes de la prédiction intra, et la présentation des résultats.

CHAPITRE I : PRESENTATION GENERALE DU CODEC H.264

1. Introduction

H.264, ou MPEG-4 AVC (Advanced Video Coding), ou MPEG-4 Part 10, est une norme de codage vidéo développée conjointement par l'UIT-T Q.6/SG16 Video Coding Experts Group (VCEG) ainsi que l'ISO/CEI Moving Picture Experts Group (MPEG), c'est le produit d'un effort de partenariat connu sous le nom Joint Video Team (JVT). La norme UIT-T H.264 et la norme ISO/CEI MPEG-4 Part 10 (ISO/CEI 14496-10) sont techniquement identiques, et la technologie employée est aussi connue sous le nom AVC, pour Advanced Video Coding. La première version de la norme a été approuvée en mai 2003 et la plus récente date d'avril 2012. [1]

Ce chapitre est réservé à l'étude de la norme de compression vidéo H.264 connue aussi sous le nom MPEG4-part10. Nous définissons dans un premier temps le codage vidéo ainsi que l'évolution des Normes de compression. Ensuite, nous décrivons les spécificités de la nouvelle norme de compression H.264. Les modules de transformation et de Quantifications directes et inverses sont aussi jugé importants et peuvent être réalisé en matériel. Pour Cela, nous avons détaillé au maximum l'encodeur H.264, ainsi que l'organisation des données à l'entrée et à la sortie de chaque module.

2. Historique

Le H.264 est une norme ouverte et sous licence compatible avec la plupart des techniques de compression disponibles aujourd'hui. Un encodeur H.264 peut réduire la taille d'un fichier vidéo numérique de plus de 80 % par rapport à la norme Motion JPEG et de 50 % par rapport à la norme MPEG-4 Partie 2, sans que la qualité d'image ne soit compromise. *Résultat* : un fichier vidéo nécessite nettement moins d'espace de stockage et de bande passante sur le réseau. Ce qui signifie aussi que vous pouvez bénéficier d'une qualité vidéo nettement supérieure à un débit binaire donné [2].

Ayant été élaborée en commun par des organisations de normalisation des secteurs des télécommunications et de l'informatique, la norme H.264 est appelée à être plus largement adoptée que les normes précédentes.

Le H.264 a déjà été intégré dans de nouveaux gadgets électroniques tels que les téléphones mobiles et les lecteurs vidéo numériques. Son acceptation par les utilisateurs a été très rapide. Des prestataires de services tels que les sociétés de stockage vidéo en ligne et de télécommunications commencent également à adopter le H.264.

Dans le secteur de la vidéosurveillance, il est fort probable que le H.264 trouvera rapidement sa place dans des applications où les utilisateurs exigent à la fois une haute résolution et une fréquence d'images élevée, comme pour la surveillance des autoroutes, des aéroports et des casinos, où une fréquence de 30/25 images (NTSC/PAL) par seconde est nécessaire. C'est à ce niveau que la réduction des besoins de bande passante et de stockage entraînera les plus grandes économies. Le H.264 devrait également accélérer l'adoption des caméras mégapixel dans la mesure où cette technologie de compression très efficace peut réduire la taille des fichiers encombrants et le débit obtenus sans compromettre la qualité d'image. Mais tout n'est pas parfait. Si le H.264 permet d'économiser sur les frais de stockage et la bande passante, il nécessite cependant des caméras réseau et des postes de surveillance plus performants.

3. Le codage vidéo

3.1. Définition

Une image en physique est une représentation d'un objet, produite par la réunion des rayons ou des faisceaux lumineux qui en émanent et se reconstituent sur un miroir, sur un écran ou sur l'œil qui perçoit cette image. Une image numérique est une image échantillonnée selon les deux axes spatiaux. Donc, une image numérique est composée d'une grille de points élémentaires appelés pixels [3]. Le Nombre de pixels par image, le nombre de bits pour le codage de chaque pixel et la nature de ces pixels constituent les principales caractéristiques d'une image numérique. Une vidéo numérique est une séquence d'images numériques [4]. L'œil humain a comme caractéristique d'être capable de distinguer environ 20 images par seconde. Ainsi, en affichant plus de 20 images par seconde, il est possible de tromper l'œil et de lui faire croire à une image animée [5]. En plus des caractéristiques d'une image

numérique fixe, une vidéo numérique est caractérisée par le nombre d'images par seconde (FPS = Frame Per Second). Le codage vidéo consiste à l'opération d'identification et de représentation d'une vidéo à l'aide d'un code informatique.

3.1.1. L'espace colorimétrique des pixels

Pour représenter une couleur, une première approche consiste à estimer l'ensemble des composantes spectrales, en filtrant le signal de couleur par de nombreux filtres à bande étroite. L'intensité de chaque sortie filtrée donnerait une estimation des longueurs d'onde présentes dans le spectre. Seulement cela Nécessite d'utiliser un trop grand nombre de capteurs. Des expériences psycho-visuelles d'égalisation ont montré qu'en combinant trois stimuli de longueurs d'onde particulières, il est possible de synthétiser presque toutes les couleurs existantes. L'espace de couleurs RGB est fondé sur les trois couleurs monochromatiques suivantes : Rouge (700 nm), Vert (546 nm) et Bleu B (435.8 nm). Le Mélange de ces 3 couleurs permet d'obtenir toutes les autres [6].

3.1.2. Le format YUV

En plus du système RGB, il existe d'autres systèmes de couleurs tels les systèmes YUV ou YCbCr. Ces systèmes exploitent le fait que le cerveau traduit le signal trichromatique perçu par l'œil, comme un signal composé de trois composantes, une luminance Y et deux chrominances Cb et Cr [7]. La compression d'images et de vidéo exploite ce principe. Cette représentation est obtenue par la Transformation suivante :

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ U &= 0.564(B - Y) \\ V &= 0.713(R - Y) \end{aligned} \qquad \textbf{(I.1)}$$

On préférera donc un espace de luminance et chrominances (YUV) où les primaires sont décorrélées, Ce qui offre l'avantage de séparer les informations d'intensité lumineuse et de couleur.

3.1.3. Format d'image (première idée de compression)

Comme l'œil humain est moins sensible à la couleur qu'à l'intensité lumineuse

(question de proportion de bâtonnets et cônes dans la rétine [8]), on sous-échantillonne la couleur, en prenant la moitié des Informations (format 4:2:2) ou bien le quart des informations (format 4:2:0) [4] :

- Format (4:4:4): Chaque composante est codée de la même manière, il n'y a pas de sous-échantillonnage des chrominances.

- Format (4:2:2): Les deux chrominances ont le même nombre de lignes que la luminance, mais deux fois moins de colonnes.
- Format (4:2:0): Les chrominances ont deux fois moins de lignes et deux fois moins de colonnes que la luminance. Chaque composante du signal vidéo peut être codée sur 8 à 10 bits. 8 bits sont suffisants pour représenter une composante puisque l'œil ne distingue pas deux niveaux voisins. D'une manière générale, les composantes du signal vidéo sont le plus souvent codées sur 8 bits avec un format 4:2:0. On a donc en moyenne 1,5 octet par pixel [9].

3.2. La compression vidéo

La compression vidéo (codage vidéo) a pour but de réduire et de supprimer la redondance spatiale et temporelle des données vidéo de façon à permettre leurs transmissions et leurs stockages sur des bandes fréquentielles étroites et des supports de capacités limités. Le processus consiste à traiter la source vidéo à l'aide d'un algorithme. Pour lire le fichier compressé, un algorithme inverse est appliqué, ce qui permet d'obtenir une vidéo contenant pratiquement le même contenu que la source vidéo d'origine (Figure I.1). Une paire d'algorithmes travaillant ensemble est appelée un **codec** vidéo (Codeur /décodeur).

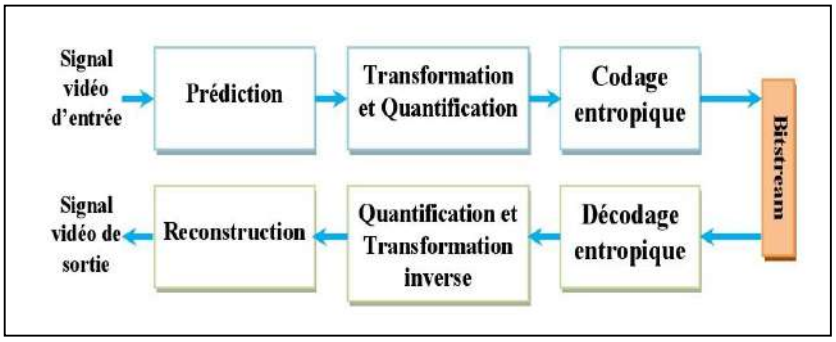


Figure I.1.*Chaine direct et inverse d'un codec vidéo*

3.2.1. Processus de codage vidéo

Un codeur vidéo consiste à mettre en œuvre séquentiellement des traitements de base (prédiction, transformation et codage) pour produire un flux binaire compressé. La condition sur les traitements de base est qu'ils soient réversibles et produisent une quantité de données compressées et inférieurs aux flux binaire initiale.

3.2.1.1. Prédiction

Une image d'une vidéo est subdivisée en macroblocs (N×N pixels). Le codeur calcule une prédiction du macrobloc courant d'après les données déjà codées provenant soit de la même image. (Prédiction Intra) ou bien à partir d'autres images (prédiction Inter). Le codeur soustrait ensuite la prédiction ainsi formée du macrobloc courant pour former un résidu (compensation) [10].

3.2.1.2. Transformation et quantification

C'est une manipulation mathématique (généralement fréquentielle) permet de concentrer l'information dans un nombre réduit de coefficients [4]. La majorité des méthodes de compression utilise des transformations sur les blocs prédits ou bien sur les blocs résidus. La quantification est l'étape de compression non-conservative qui diminue la quantité d'information contenue dans les coefficients transformés. Elle consiste à diviser chacun des coefficients d'un bloc par un pas de quantification QP et de n'en conserver que la partie entière. Les coefficients DCT quantifiés sont ensuite

réorganisés et réordonnancés dans un vecteur unidimensionnel en les balayant en zigzag pour augmenter le nombre des zéro successifs, et par conséquent l'augmentation de l'efficacité de l'étape suivante (codage entropique).

3.2.2. Processus de décodage

Un décodeur vidéo consiste à effectuer les opérations inverses du codeur (décodage, transformation inverse et reconstruction) pour produire une séquence vidéo décodée la plus proche possible de la séquence initiale [10].

Décodage du flux binaire : Le décodeur vidéo reçoit le flux binaire compressé. La première étape consiste à décoder chacun des éléments de syntaxe et extrait les coefficients de transformée quantifiés et les paramètres de compression. Ces informations sont ensuite utilisées dans les deux étapes suivantes pour inverser le processus de codage et recréer la séquence d'images initiale.

Quantification et transformation inverses : Premièrement, les coefficients sont remis à l'échelle, ceci en les multipliant par le bon pas de quantification. Comme la quantification est une compression non conservative, généralement on n'obtient pas les mêmes valeurs que les coefficients de départ. Ensuite, la transformée inverse est utilisée pour reconstruire les macroblocs ou bien les résidus.

Reconstruction : Pour chaque macrobloc, le décodeur reforme une prédiction identique à celle effectuée par le codeur. Il lui ajoute ensuite les résidus précédemment calculés pour reconstruire le macrobloc, qui pourra enfin être affiché comme une partie d'une image de la séquence vidéo.

3.2.3. Evaluation de la qualité des images décodées

La compression vidéo avec pertes exploite les failles du système visuel humain pour réduire la quantité des informations à transmettre. La qualité des images est donc réduite de façon plus ou moins visible. L'évaluation de la qualité d'une image ou d'une vidéo est très complexe. Il existe des métriques de qualité objective, basées sur un calcul de distorsions comme le PSNR (Peak Signal to Noise Ratio) très utilisé dans le traitement du signal en général. Le PSNR mesure une erreur quadratique moyenne non pondérée sur toute l'image.

$$PSNR = 10 * \log_{10}\left(\frac{(2^n-1)^2}{\frac{1}{N}*\sum_0^{N-1}(originale-recostruite)^2}\right) \quad \textbf{(I.2)}$$

3.3. Les normes de compression vidéo

Au cours des dernières années, l'intérêt pour le multimédia a entraîné l'investissement d'une somme importante d'efforts de recherche sur le codage vidéo dans les universités et l'industrie, efforts qui se sont traduits par l'élaboration de plusieurs normes [3]. Ces normes visent un vaste éventail d'applications aux exigences variées quant au débit binaire, à la qualité d'image, à la complexité et au délai ainsi qu'à l'amélioration des rapports de compression. Les standards de compression vidéo ont pour but d'assurer la compatibilité entre encodeur et décodeur, tout en laissant le champ libre aux fabricants pour développer des produits innovants et compétitifs. D'où la définition qu'un standard ne définit pas un encodeur, mais le résultat qu'un encodeur doit produire. Autrement dit, il porte sur la syntaxe du flux et la manière de le décoder [9].

3.3.1 H.264/AVC ou bien MPEG4/AVC part 10

En 1998, l'UIT-T lança le projet H.26L dans le but de créer une nouvelle architecture de codec ayant pour but un gain en efficacité de codage d'un rapport au moins égal à 2 par rapport aux standards existants. Un autre but était de créer une interface simple pour pouvoir adapter le codec aux différents protocoles de transport (commutation de paquets et de circuits). Le codec a été développé en s'assurant qu'il serait transposable sur plateformes à un coût raisonnable en tenant compte des progrès réalisés par l'industrie des semi-conducteurs. En 2001, le projet H.26L avait atteint ses objectifs en taux de compression [11]. En décembre 2001, et après s'être réunis pour la définition en commun du standard MPEG-2/H.262, les deux groupes, VCEG de l'UIT-T et MPEG de l'ISO/IEC, ont à nouveau mis leurs travaux en commun au sein du JVT (Joint Video Team) pour aboutir un nouveau standard, adopté de manière identique des deux côtés [8]. Le H.264/AVC est le nom employé par l'UIT-T, l'ISO/IEC préférant pour sa part opter pour l'appellation MPEG-4 Partie10/AVC, la norme étant présentée comme un nouveau élément de la série de normes MPEG-4. Cette nouvelle norme est approuvée en mai 2003 par

l'UIT-T et en octobre 2003 par l'ISO [12]. La version finale de H.264/AVC est approuvée en mars 2005 [13]. Après cette version, plusieurs autres améliorations ont été approuvées (2009, 2012).

3.3.2. Evolution des normes de compression vidéo et leurs applications

La figure I.2 montre l'évolution des différents standards et normes de codage des séquences vidéo des deux comités UIT-T et ISO/IEC. Cette figure représente aussi une classification des normes de codage vidéo.

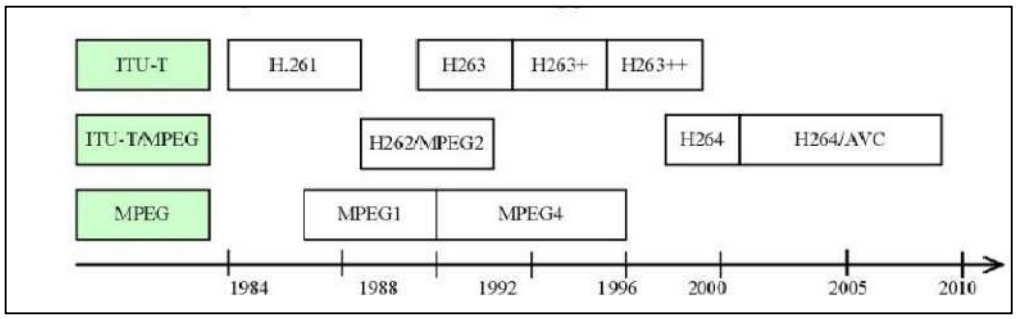


Figure I.2.*Evolution des standards de compression vidéo*

Tableau I.1. *Les normes de codage vidéo et leurs applications.*

Standard	Application
MPEG-1	Vidéo-CD, CDI
MPEG-1	Télévision numérique, Satellite, TNT, TV Haute Définition, DVD
MPEG-4 Visual	Internet, vidéo mobile, VOD (Video On Demand), Studio
MPEG-4/AVC H.264/AVC	Internet, vidéo mobile, Haute Définition, VOD (Video On Demand), Studio

4. La nouvelle norme de compression H.264

La norme H.264 est spécialement conçue pour remédier à plusieurs faiblesses des normes de compression vidéo précédentes. Cette norme est basée sur un codage hybride fondé sur les blocs et une amélioration des concepts et outils existants, notamment pour les prédictions Intra et Inter.

L'amélioration des performances est significative sur le taux de bits avec la même qualité visuelle, mais la complexité algorithmique est bien plus élevée [14]. H.264 offre les avantages suivants :

- A une qualité vidéo équivalente, H.264 offre la réduction moyenne du débit de 50% comparé à MPEG-2. Qui donne la possibilité de transmettre de la télévision à haute définition et de la télévision mobile avec un taux de bits raisonnable.
- Les erreurs de transmission sur différents réseaux sont tolérées (Tolérance d'erreurs) ;
- Latence réduite et meilleure qualité en cas de latence supérieure ;
- Spécification de syntaxe simple facilitant l'implémentation logicielle et matérielle ;
- Décodage correspondant exactement à la source et définissant de façon exacte les numérisations qui doivent être effectuées par un encodeur et un décodeur pour éviter l'accumulation d'erreurs.

4.1. Présentation générale

Le document « Recommandation H.264 : Advanced Video Coding [12]», publié en 2003, ne définit pas un codec à proprement parler, mais plutôt une syntaxe pour le flux vidéo compressé et une méthode pour décoder cette syntaxe et produire une séquence vidéo affichable [10].

4.1.1. Division des images en blocs et macroblocs

Une séquence vidéo compressée en utilisant H.264/AVC doit être constituée de plusieurs images, de type différent, structurées en GOP (Group Of Picture). Un GOP (Figure I.3) est une suite d'images codées suivant trois méthodes : codage intra-image (I-frame), prédictif (P-frame) et bidirectionnel (B-frame) [3].

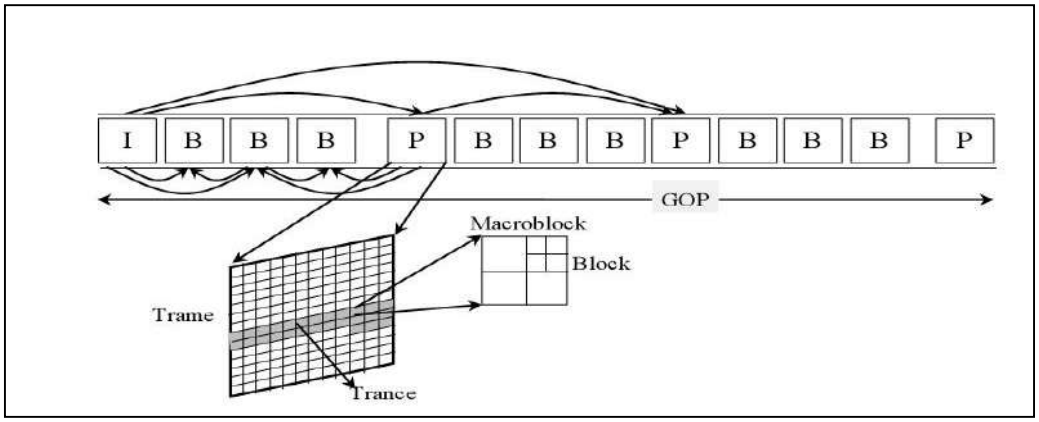


Figure I.3. *Organisation des images dans une séquence vidéoH.264*

Dans un GOP, l'image "I" est une image de référence codée, indépendamment des autres, par prédiction Intra. Elle fournit les points d'accès dans le train binaire, elle sert aussi au codage des images de type P et B, son taux de compression est moyen. L'image "P" est une image codée par prédiction Inter en utilisant l'estimation et la compensation de mouvement à partir des autres images codées et décodées (I et/ou P). Dans la dernière image de type "B", les macroblocs peuvent être prédits soit par simple compensation de mouvement avant ou arrière à partir d'images "P" ou "I" soit par double compensation de mouvement avant et arrière. Chaque image (ou bien trame) est partitionnée en macroblocs de dimension fixe qui couvrent une zone rectangulaire de l'image (16×16 échantillons de luminance et de 8×8 échantillons des deux composants de chrominance) représentant les unités maximales de codage (Figure I.4). Ces macroblocs sont séparés en sous-macroblocs qui peuvent avoir les tailles suivantes (16×8), (8×16), (8×8), (8×4), (4×8) ou (4×4) [49].

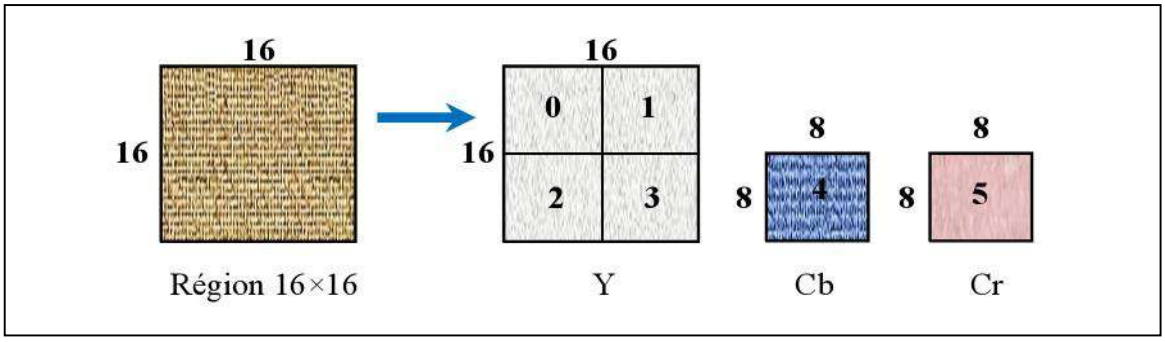


Figure I.4. *Les macroblocs 16x16 et 8x8 dans H.264*

4.1.2. Les critères de distorsion dans H.264

Dans un encodeur H.264, les deux modes de prédiction sont basés sur la comparaison des macroblocs dans la même image (prédiction Intra) ou bien à partir des images différentes (prédiction Inter). La comparaison se résume à la minimisation d'un des critères de distorsion. Le choix de l'opérateur de distorsion n'est pas imposé par les deux comités de standardisation (ISO/IEC, UIT-T), mais le plus souvent utilisé c'est le SAD [16].

4.2. Vue d'ensemble du codec H.264/AVC

La norme visuelle H.264 partage un certain nombre de dispositifs avec les normes passées, y compris H.263 et MPEG-4. Principalement, H.264 est basée sur un modèle hybride de la modulation de code d'impulsion différentielle adaptative (ADPCM=Adaptive Différentiel Pulse Code Modulation) et une transformation basée sur le codage des entiers semblable à la transformation des cosinus discret (DCT) utilisé dans des normes antérieures [16]. Ce codage complexe est fait pour tirer profit des redondances temporelles et spatiales qui se produisent dans les images visuelles successives. L'utilisation d'algorithmes complexes au niveau de l'encodeur implique une charge plus importante des calculs. Par comparaison, l'encodeur H.264 est dix fois plus complexe que l'encodeur MPEG-2 pour un gain de 50% en taille de fichier [14].

4..2.1. Flux de données dans H.264

Le standard H.264/AVC regroupe une couche vidéo (Video Coding Layer | VCL), qui représente le contenu vidéo, et une couche réseau (Network Abstraction Layer | NAL), qui comprend les en-têtes appropriées pour le transport de l'information par des couches transports ou des supports de stockages particuliers. La conception du VCL suit l'approche de codage vidéo par blocs, comme dans toutes les normes antérieures (MPEG et ITU) [10]. La sortie du processus de codage des données est VCL (une séquence de bits représentant les données vidéo codées) qui sont mappés à des unités NAL avant la transmission ou le stockage. Chaque unité NAL contient une séquence d'octets bruts de charge utile (RBSP), un ensemble de données correspondant à des données vidéo codées ou des informations d'en-tête. Une séquence vidéo codée est représentée par une séquence des unités NAL qui peuvent être transmis sur un réseau en mode paquet ou une liaison de transmission à haut débit ou stockées dans un fichier [17].

4.2.2. L'encodeur H.264

La norme H.264 permet le codage du signal vidéo au format chroma 4:2:0, avec des images progressives ou entrelacées, éventuellement combinées dans une même séquence [3]. La couche VCL est définie pour représenter de manière efficace le contenu des données vidéo. Le concept de la couche de codage vidéo H.264/AVC est semblable à celui des autres normes telles que MPEG-2. Il s'agit d'un hybride de prédiction temporelle et de prédiction spatiale, allié à un codage par transformée. Le standard H.264/AVC est un codeur en boucle fermée [18], le codeur H.264 comprend le décodeur (en boucle) afin de réaliser la prédiction pour les blocs suivants ou bien l'image suivante (Figure I.5). Les coefficients quantifiés subissent un rééchelonnement et une transformée inverses similaires à ceux du décodeur, aboutissant au reliquat de prédiction décodé. Le reliquat est ajouté à la prédiction. Le résultat de cette opération est transmis au filtre anti-blocs permettant d'éliminer certaines dégradations produites par le module de quantification qui engendre des pertes d'informations. Ce filtre lisse les images de référence en bordure des blocs. Les macroblocs et slices décodés sont ensuite stockés en mémoire. Les blocs décodés de l'image courante sont utilisés pour le calcul des prédicteurs Intra. De même, les images précédemment décodées et filtrées sont utilisées pour le codage Inter.

Dans la chaine de compression directe, et afin que le décodeur soit capable de retrouver le prédicteur Intra/Inter, les modes de prédiction Intra et les vecteurs correspondant au mouvement dans la prédiction Inter sont transmis au décodeur. Ces informations sont envoyées avec les blocs résiduels dans l'encodeur entropique pour former le train binaire [19].

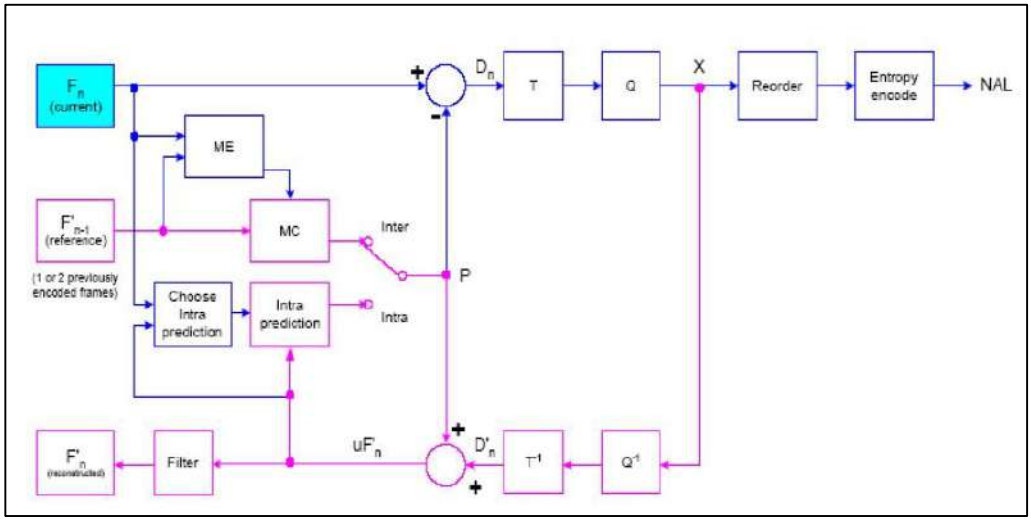


Figure I.5. *Architecture de l'encodeur H264.*

4.2.2.1. Le codage par prédiction dans H.264

Dans les images de type Intra, chaque macrobloc peut être transmis en utilisant un type de codage parmi d'autres en fonction de celui utilisé pour le codage de la trame. Contrairement aux normes de codage vidéo précédentes où la prédiction intervenait dans le domaine transformé, la prédiction spécifique à la norme H.264/AVC intervient toujours dans le domaine spatial en se référant aux échantillons voisins de blocs déjà codés/décodées mais non filtrés. Dans tous les types de codage de trames, deux classes de codage intra-image sont pris en charge: Intra4×4 et Intra16×16. Dans le mode Intra4×4, chaque bloc (4×4pixels) de luminance recourt à un des neuf modes de prédiction définit dans le logiciel de référence [13]. En effet, outre la prédiction DC, huit modes de prédiction directionnelle sont spécifiés. Dans le mode Intra16×16 (recommandé dans le cas des zones d'image régulières) une prédiction uniforme est réalisée pour l'ensemble de la luminance d'un macrobloc. Pour ce mode de prédiction, quatre modes de prédiction

coexistent. La prédiction des échantillons de chrominance d'un macrobloc repose toujours sur une technique semblable à celle utilisée pour la luminance (Intra16×16). La prédiction intra-image au-delà des limites de la trame n'est pas autorisée afin de maintenir l'indépendance des trames les unes par rapport aux autres. Outre les types de codage intra-image des macroblocs, plusieurs types de codage prédictif ou à compensation de mouvement sont définis pour les macroblocs d'une trame P. Des partitions avec des macroblocs et des sous-macro-blocs sont prises en charge par la syntaxe correspondant aux types de prédiction Inter (Inter16×16, Inter16×8, Inter8×16 et Inter8×8). Dans le mode Inter8×8, un élément syntaxique supplémentaire est transmis pour déterminer les autres types de prédiction (Inter8×4, inter4×8 ou Inter4×4). Le signal de prédiction pour chaque bloc de luminance N×M à codage prédictif est obtenu en déplaçant une zone de l'image de référence. Il est définit par un vecteur de translation et un index de référence d'image. Par conséquent, si le macrobloc est codé en utilisant le mode Inter8×8, et que chaque sous-macro-bloc est codé en utilisant l'Inter4×4, un maximum de 16 vecteurs de mouvement peut être transmis pour un seul macrobloc de trame P [18].

4.2.2.2. La reconstruction des images

Parmi les nouveautés de H.264 est la reconstruction des images dans l'encodeur comme dans le décodeur, et cela dans le but de manipuler les mêmes images au niveau de l'encodeur qu'au niveau du décodeur. En effet, dans l'encodeur et le décodeur H.264, les macroblocs résiduels après quantification et transformation inverses sont combinés avec les macroblocs déjà décodé pour former le nouveau macrobloc reconstruit. Pour le codage intra, nous ajoutons le macrobloc résiduel au macrobloc reconstruit selon le mode intra choisi dans l'étape de codage, alors que pour le codage inter-image nous ajoutons ce résiduel au macrobloc reconstruit suivant les vecteurs de mouvement indiqués dans le bitstream [19].

Généralement, pour améliorer la qualité visuelle des images après encodage/décodage, il est nécessaire d'utiliser un filtrage après l'étape de transformation inverse dans le décodeur. En effet, l'un des artéfacts les plus visibles, dans les techniques basées sur les blocs, est l'apparition de l'effet de blocs dans les images décodées (la

visibilité de la structure en blocs ou bien l'effet de pixellisation). Les bords d'un bloc sont en général reconstitués avec moins de précision que les pixels intérieurs. De nombreux filtres, débloquant atténuant les contours des blocs et MB, sont définis dans la littérature, et cela par lissage de pixels. H.264/AVC inclus un filtre anti-blocs (deblcoking filtre) lissant les trames lors de la reconstruction. Ce filtre est jugé, dans plusieurs travaux, hautement adaptatif où la puissance du filtrage est contrôlée par les valeurs de plusieurs éléments syntaxiques. La pixellisation est alors réduite sans affecter outre mesure la netteté du contenu et la qualité subjective est considérablement améliorée [18]. En plus, l'une des nouveautés de H.264 est l'utilisation du filtre en boucle au niveau de l'encodeur, cela est dans le but de manipuler les mêmes images, pour la reconstruction Inter, au niveau de l''encodeur qu'au niveau du décodeur. Ceci permet une prédiction de meilleure qualité apportant des gains dans le bit-rate typiquement de 5 à 10 %, sans diminution de la qualité tout en produisant la même qualité objective que la vidéo non filtrée [18].

4.3. Le décodeur H.264

Le processus de décodage consiste à interpréter les symboles codés à partir d'un bitstream conformes, et traiter ces données conformément à la spécification du standard, afin de générer des séquences vidéo reconstruite. Ainsi, le processus de décodage est généralement constitué de deux voies principales : la génération des macroblocs prédits et le décodage des macroblocs résiduels. Les valeurs des échantillons des macroblocs résultant de ces deux voies sont additionnées [20]. Le schéma généralisé d'un décodeur H.264/AVC est donné sur la figure(I.6).

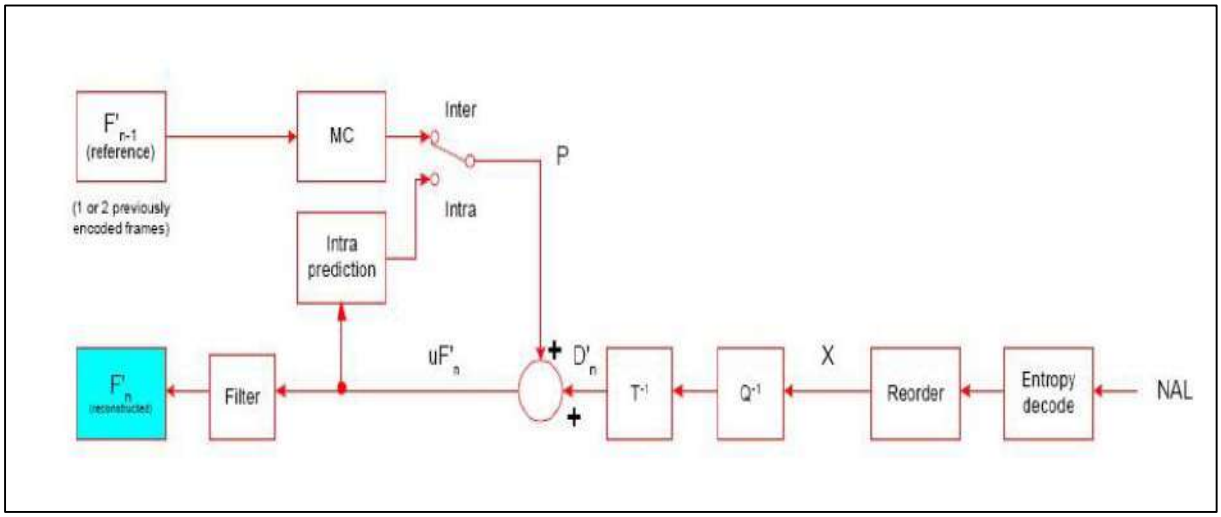


Figure I.6 . *Architecture de décodeur H264.*

4.4. Les profils et les niveaux de H.264/AVC

Dans la norme H.264, trois profils sont développés, chaque profil est adapté à des domaines d'applications bien définis [18]. Un profil de base (baseline profile) utilise uniquement des images de type 'I' et 'P'. Ce profil utilise uniquement les algorithmes Intra et Inter, l'encodeur entropique CAVLAC et le filtre anti-blocs. Le deuxième profil est le profil principal (main profile) qui manipule, en plus des images 'I' et 'P', des images intermédiaires de type 'B'. Pour ce profil, des améliorations sont apportées sur l'algorithme inter-prédiction et le codage entropique CABAC. Ensuite le profil étendu (Extended profile), utilise en plus des images de type SI et SP avec des améliorations significatives au niveau des algorithmes. Aux trois premiers profils, un nouveau profil (Hight profile) a été ajouté [8]. La Figure I.7 montre l'interrelation entre les trois versions. En plus, le tableau I.2 Montre les détails des trois profils [21].

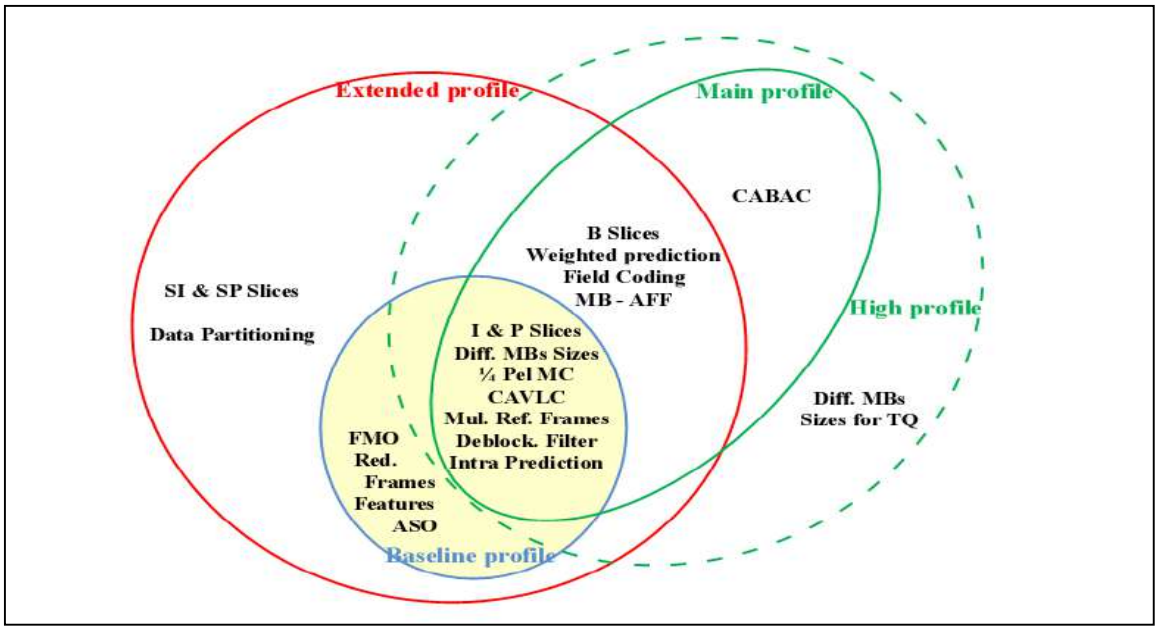


Figure I.7. *Les versions de H.264 et les outils correspondants.*

Tableau I. 2. *Les trois profils de la norme H.264.*

Outils	Baseline	Main	Extended	High
I and P Slices	x	x	x	x
CAVLC	x	x	x	x
CABAC		x		x
B Slices		x	x	x
Entrelacé (PicAFF, MBAFF)		x	x	
Enh. Error Resil. (FMO, ASO, RS)	x		x	x
Further Enh. Error Resil (DP)			x	x
SP and SI Slices			x	x
Weighted prediction		x	x	x
Intra prediction	x	x	x	x

Variable block size MC	**x**	**x**	**x**	**x**
¼ pel MC	**x**	**x**	**x**	**x**
Multiple ref. frame	**x**	**x**	**x**	**x**
In loop deblocking filter	**x**	**x**	**x**	**x**
8×8 Transform				**x**
10 bits				**x**
4:2:2 and 4:4:4				**x**
Lossless				**x**

4.5. Applications de H.264

Le format H.264 offre, en plus de son efficacité de compression, une grande flexibilité en termes d'options de compression et de support de transmission. Un encodeur H.264 peut utiliser une grande variété de méthodes de compression, s'adaptant ainsi à des applications présentant des contraintes spécifiques comme de la transmission mobile temps réel à bas débit, de la télévision haute-définition ou encore de la production vidéo professionnelle. Les problèmes du stockage et de la transmission ont été intégrés dans la démarche de conception du standard, débouchant sur un format compressé paquétisé et des options permettant de réduire les effets des erreurs de transmission, comme des codes correcteurs d'erreurs [3]. H.264 a été adopté pour de nombreuses applications, parmi lesquelles [10] :

- Les DVD haute-définition (HD-DVD et Blu-Ray) ;
- La télévision haute-définition ;
- Les produits vidéo de la marque Apple (les téléchargements vidéo sur la plate-forme iTunes, les vidéos sur iPod et MacOs) ;
- Les applications vidéo de l'OTAN et du Département de la Défense américain;
- La télévision mobile ;
- Les contenus vidéo en streaming sur Internet ;
- La vidéoconférence, etc.…

Le plus grand avantage du format H.264 sur les formats antérieurs est sans doute son efficacité de compression. En comparaison des standards MPEG-2 et MPEG-4 Visual, H.264 fournit en effet une meilleure qualité d'image à débit binaire égal ou un débit plus faible à qualité équivalente. Par exemple, un DVD simple-couche peut stocker un film d'environ deux heures au format MPEG-2 ; avec le format H.264, on pourrait stocker plus de quatre heures de vidéo en qualité DVD sur le même disque. Evidemment, ce gain en taille mémoire à une contrepartie : la complexité, multipliée par 10 par rapport aux standards précédents.

4.6. Produits et mises en œuvre de H.264

Comme de nombreuses autres normes vidéo du groupe ISO/CEI MPEG, le H.264/AVC dispose d'une application logicielle de référence, qui peut être gratuitement téléchargée. Le principal objectif de cette application est de donner des exemples des différentes possibilités du H.264/AVC, plutôt que de fournir un produit réellement utilisable et performant. Des applications matérielles de référence sont aussi en cours de normalisation. La figure I.8 montre les transpositions logicielles de l'encodeur H.264/AVC, et ceci sur la base du logiciel de référence JM [13].

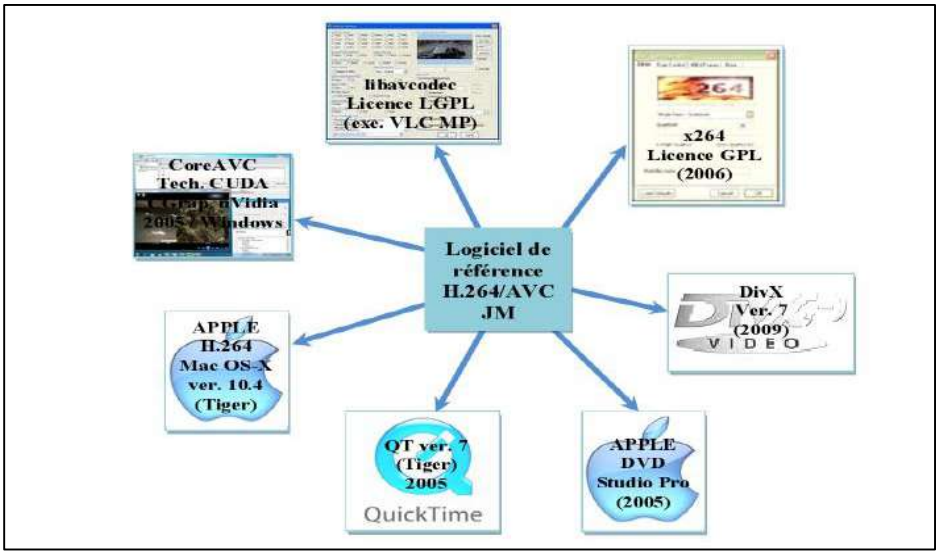


Figure I.8. *Les transpositions logicielles de H.264.*

5. Conclusion

La norme H.264 représente un énorme pas en avant dans le domaine des technologies de compression vidéo. Elle garantie une meilleure efficacité de compression en raison de la plus grande précision des fonctions prédictives et d'une meilleure tolérance des erreurs. Elle offre de nouvelles possibilités pour la création d'encodeurs vidéo offrant des flux vidéo de plus haute qualité, des fréquences d'image plus importantes et des résolutions plus élevées, à débit égal (par rapport aux normes précédentes) ou, à l'inverse, la même qualité de vidéo à des débits inférieurs.

CHAPITRE II : LA PREDICTION SPATIALE (INTRA)

1. Introduction

Le codage de la couche vidéo de H.264/AVC est réalisé dans le même esprit que les précédentes normes comme le standard MPEG-2. Elle consiste principalement en une double prédiction temporelle et spatiale de la séquence. La prédiction intra- image utilise uniquement l'image courante. Il s'agit donc d'exploiter la redondance spatiale présente dans toute image.

En effet, le principe de base est que dans un certain voisinage, la structure de l'image varie peu globalement. Bien sûr, ceci dépend de la taille de l'échantillon considéré, c'est pourquoi chaque taille de bloc correspond à un traitement spécifique. De fait, pour la luminance, la prédiction peut être réalisée sur des blocs 4x4 ou des macro-blocs 16x16. De plus, bien que la luminance et la chrominance n'aient pas la même résolution, la chrominance n'est codée que par blocs 8x8. Un bloc prédit est donc construit à partir de ces voisins déjà codés et reconstruits. Il est ensuite soustrait au bloc original et le bloc différence est codé et transmis. Toute la procédure de prédiction doit ensuite être insérée au flux codé, ce qui nécessite de l'information supplémentaire dont l'importance ne doit pas être négligée. Cette section présente les différentes procédures de prédiction intra- image, suivant la taille de bloc considérée, et la manière dont la procédure de prédiction est codée dans le flux.

2. Le codage intra -image

Comme mentionné auparavant, la prédiction et l'encodage Intra exploitent la redondance spatiale de l'image de référence de type I, celle-ci est décomposée en macroblocs chacun de taille 16×16 pixels. Ce principe est identique au MPEG-2 et MPEG-4 part-2, mais les techniques suivantes sont spécifiques au standard H.264 :

- Les macroblocs de taille 16×16 pixels peuvent être décomposés en 16 blocs de taille 4×4 pixels.
- 4 prédictions sont définies pour les macroblocs 16×16 (vertical, horizontal, DC et plane)
- 9 prédictions sont définies pour les blocs 4×4 (vertical, horizontal, DC et 6 diagonaux).

Dans le codage intra-image, la prédiction d'un macrobloc de l'image de

référence est réalisée en utilisant les pixels de macroblocs voisins précédemment encodés et reconstruits mais non filtrés, à l'aide de deux méthodes (Intra16×16 et Intra4×4).

2.1. Prédiction intra 4x4

Dans la Figure (II.1), les pixels {a, b, c, …, p}, du bloc en cours de traitement, sont interpolés à partir des pixels, en haut et à gauche, déjà décodés {A, B, C, …, L}, et cela suivant plusieurs modes [17]. Les modes disponibles pour chaque bloc, dépendent des pixels voisins disponibles. Si des pixels voisins manquent (exemple du premier bloc de la trame), des valeurs par défaut sont utilisées. Dans H.264, neuf modes ont été choisis pour l'Intra4×4.

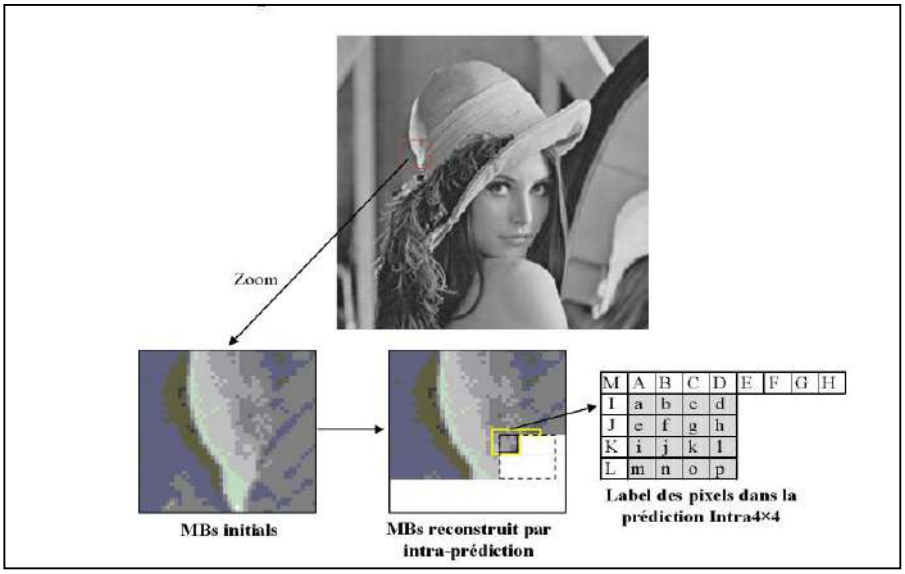


Figure II. 1. *Prédiction Intra4×4 à l'aide des pixels voisins dans H.264.*

Dans les deux types de prédiction, le calcul du mode se fait par interpolation des voisins selon un angle défini dans la Figure (II.2). Pour l'Intra4×4, huit directions ont été choisies. Si nous prenons le premier mode (direction verticale), les pixels {a, b, c, … et p} sont interpolés verticalement, ainsi les pixels a, e, i et m reçoivent la valeur de A. Un calcul plus complexe est utilisé en mode 7 où l'angle d'interpolation est $\frac{11\pi}{8}$ de radians, ce qui nous donnera par exemple a = (A+B) /2 et e = (A+2B+C) /4. Deux autres modes

différents sont aussi utilisés, le mode « DC » où les pixels prennent la moyenne des valeurs de A, B, C, D, I, J, K et L, et le mode « PLANE » où la prédiction est un dégradé oblique, calculé par moyenne des valeurs obliques (une moyenne des deux sens de la direction « mode 3 ») [15].

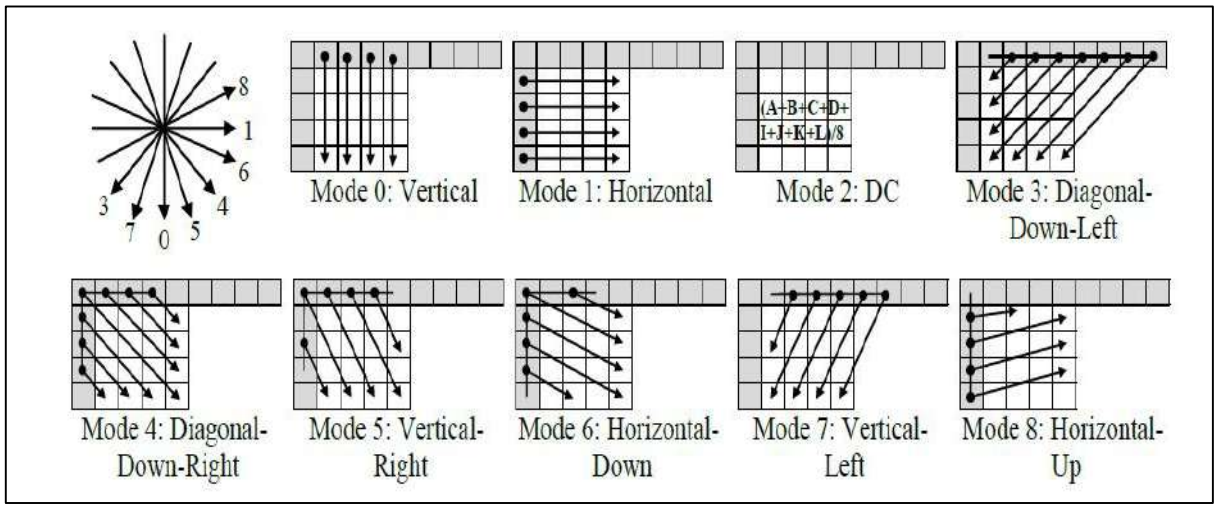


Figure II. 2 . *Les 9 modes de prédiction Intra4×4 dans H.264*

2.2. Calcul de la prédiction intra 4x4

2.2.1. Mode vertical (mode 0)

Dans ce mode on utilise les pixels voisins horizontal, comme il monter la figure II.3.

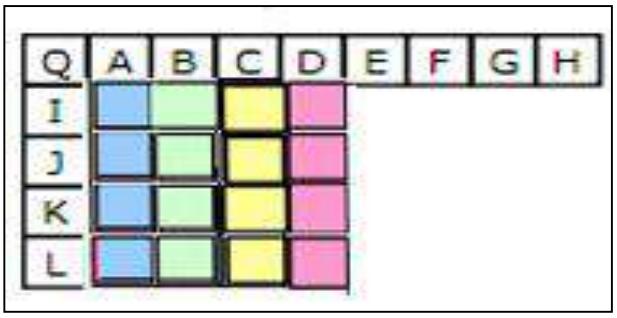


Figure II. 3. *Prédiction (mode 0) Mode vertical.*

Où : A ; B ; C ; D ;

2.2.2. Mode Horizontal (mode 1)

Dans ce mode on utilise les pixels voisinage vertical comme il monter le figure II.4.

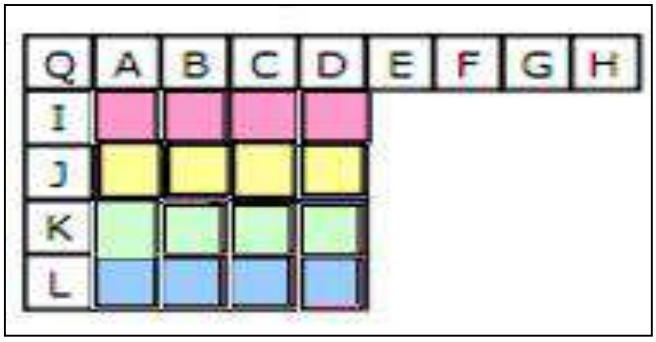


Figure II. 4. *Prédiction (mode 1) Mode Horizontal.*

Où : I J K L

2.2.3. Mode DC (mode 2)

Dans ce mode, les pixels voisins horizontal et vertical sont utilisés pour calculer la moyenne, comme il monter le figure II.5.

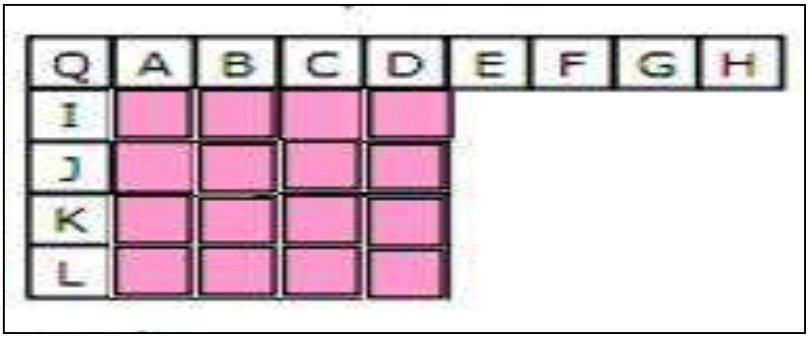


Figure II. 5. *Prédiction (mode 2) Mode DC.*

Si les pixels Voisinages supérieurs et gauches sont disponibles :

(A + B + C + D + I + J + K + L + 4) / 8.

Si non, si les pixels Voisins Horizontal sont disponibles :

(A + B + C + D + 2) / 4

Si non, si les pixels Voisins Vertical sont disponibles :

(+ I + J + K + L + 4+ 2) / 4; Si non a = 128.

2.2.4. Diagonal Bas gauche (mode 3)

Dans ce mode on utilise les pixels voisins horizontal, comme il monter la figure II.6.

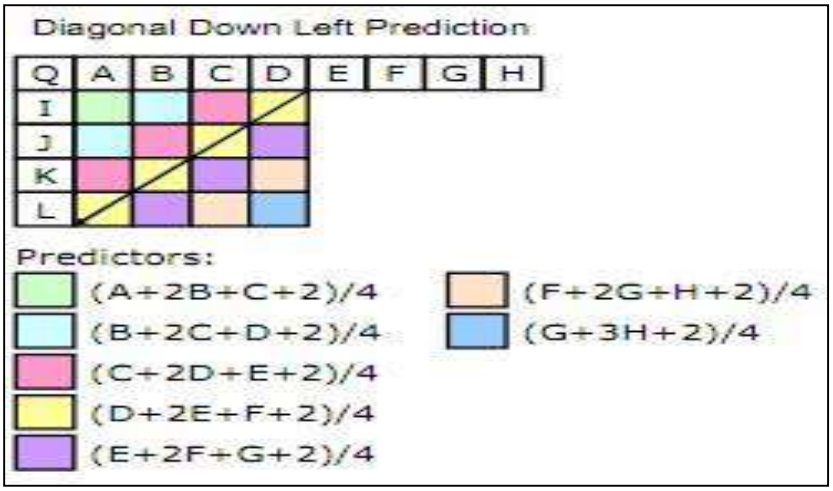


Figure II. 6. *Prédiction (mode 3) Diagonal Bas gauche***.**

2.2.5. Diagonal Bas Droite (mode 4)

Dans ce mode on utilise les pixels voisins horizontal et vertical, comme il monter le figure II.7.

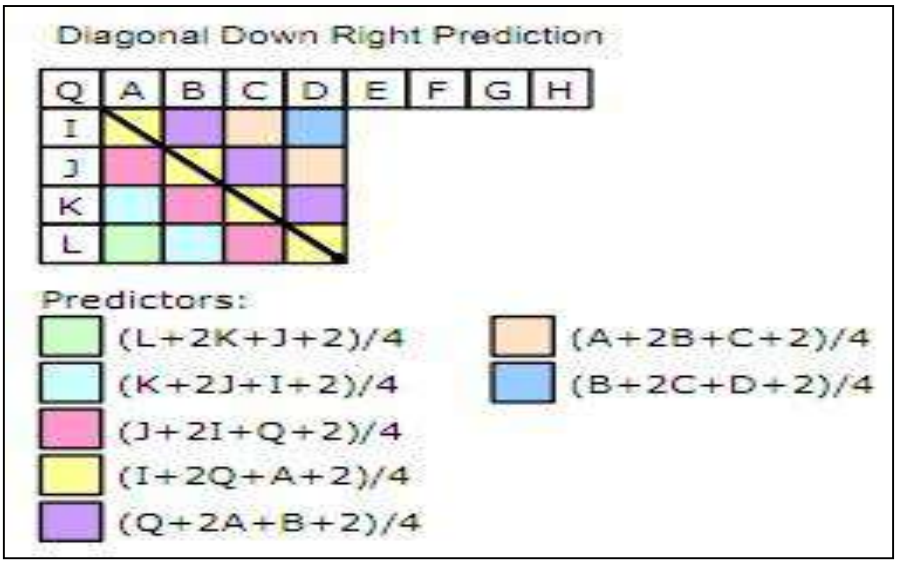


Figure II. 7. *Prédiction (mode 4) Diagonal Bas Droite.*

2.2.6. Diagonal Vertical droit (mode 5)

Dans ce mode on utilise les pixels voisins horizontal et vertical, comme il monter la figure II.8.

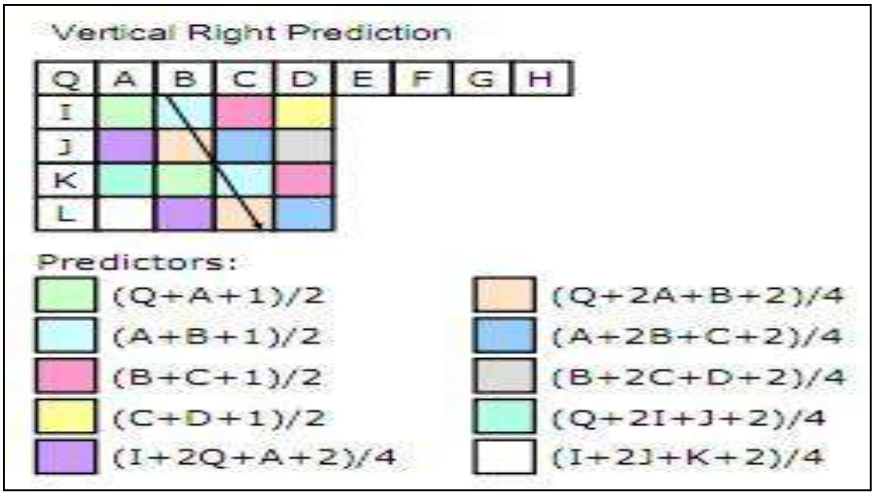


Figure II. 8. *Prédiction (mode 5) Diagonal Vertical droit.*

2.2.7. Diagonal Horizontal bas (mode 6)

Dans ce mode on utilise les pixels voisins horizontal et vertical, comme il monter le figure II.9.

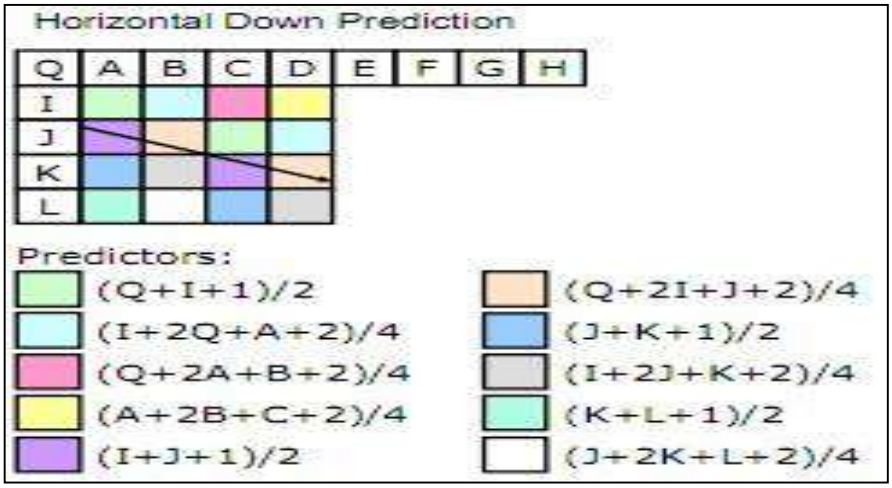


Figure II. 9: *Prédiction (mode 6) Diagonal Horizontal bas.*

2.2.8. Diagonal Vertical gauche (mode 7)

Dans ce mode on utilise les pixels voisins horizontal, comme il monter la figure II.10

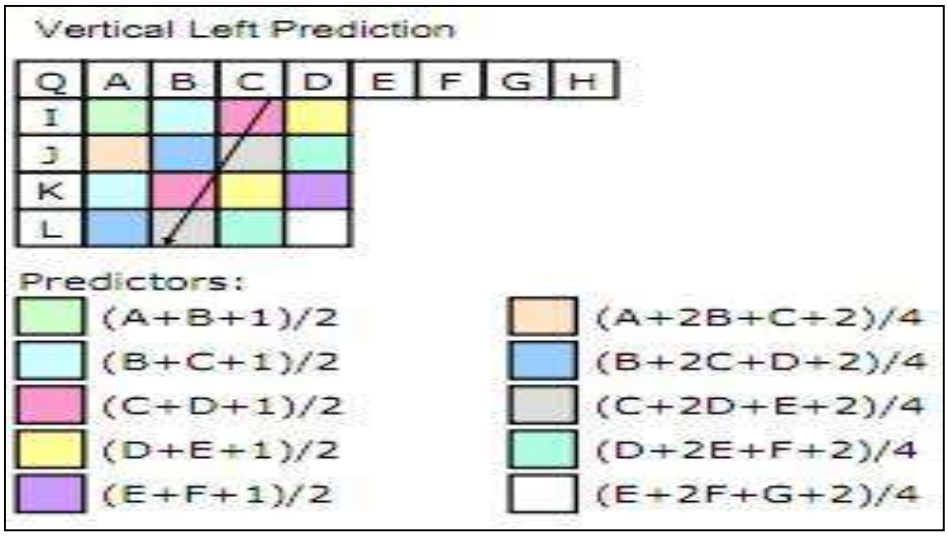


Figure II. 10.*Prédiction (mode 7) Diagonal Vertical gauche..*

2.2.9. Diagonal Horizontal haut (mode 8)

Dans ce mode on utilise les pixels voisins vertical, comme il monter la figure II.11.

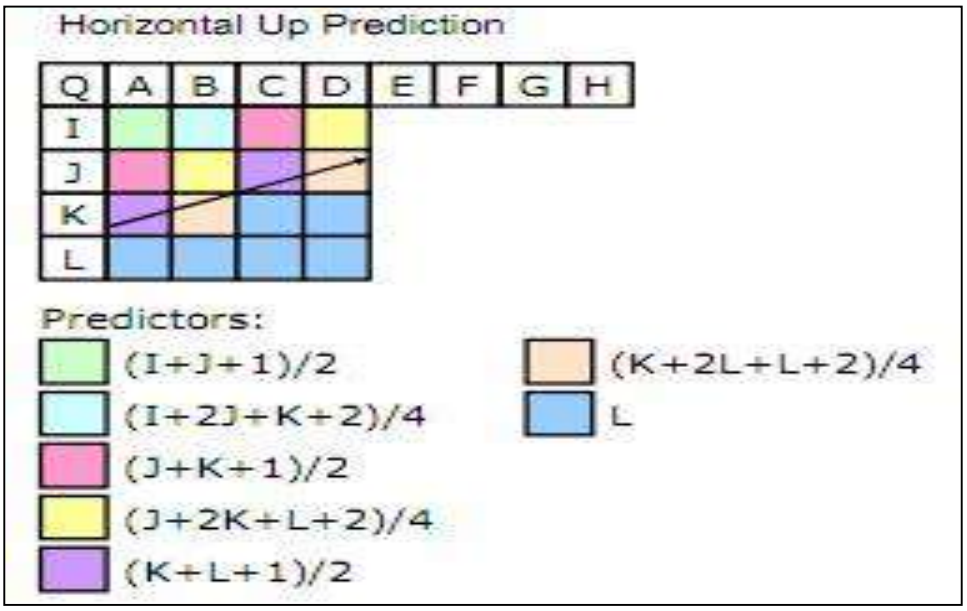


Figure II. 11. *Prédiction (mode 8) Diagonal Horizontal haut.*

2.3. Prédiction intra 16x16

La même stratégie est appliquée pour l'Intra16×16 avec 4 modes uniquement. La Figure (II.12) montre ces 4 modes basés sur les directions (verticale, horizontale et plane), en plus le mode DC où les pixels du bloc P prennent la moyenne de l'ensemble des pixels H et V :

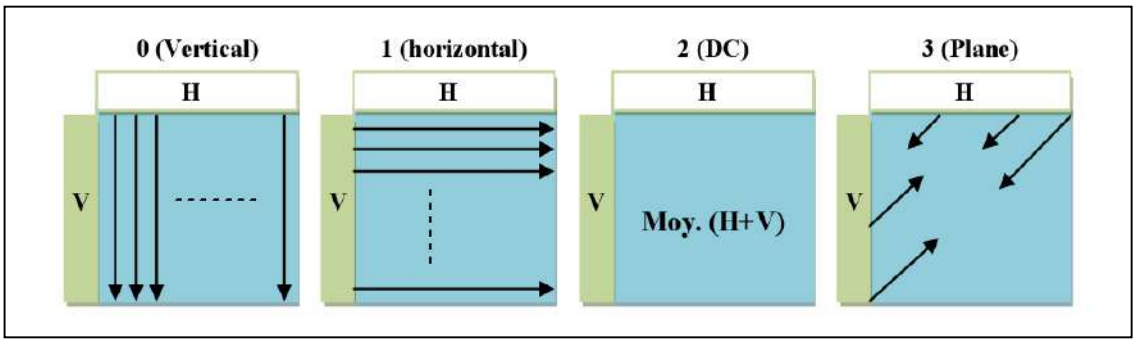


Figure II. 12. *Les 4 modes de prédiction intra16×16 dans H.264.*

Les différents modes déjà étudiés, sont appliqués au signal luminance. Pour les blocs 8×8 pixels de chrominance, 4 modes Intra4×4C sont adaptés. Les 4 modes de chroma ont la

même architecture que les modes Intra16×16 de la luminance, sauf que l'ordre est inversé (DC=mode0, horizontale=mode1, verticale=mode2 et plane=mode3) [17].

3. Mode de décision et cout de calcul codage intra

Le mode de décision n'est pas spécifié dans la norme H.264/AVC. Cette étape est laissée au choix du programmeur. Les recommandations de H.264/AVC [12] définissent deux types d'algorithmes de décision de mode (haute et basse). Dans la décision de mode de faible complexité, la distorsion est évaluée par le SAD ou bien SATD. Le taux est estimé par le nombre de bits nécessaires pour coder les informations de mode. Dans la décision de mode de haute complexité, la distorsion est évaluée par SSE. Le taux est estimé par le nombre de bits nécessaires pour coder les informations de mode et le résidu. Cette dernière méthode nécessite plus de calculs mais conduit à un meilleur rendement, elle est moins appropriée pour des applications en temps réel [18]. Pour la prédiction Intra4×4, il est nécessaire de calculer 9 coûts pour chaque bloc (16 blocs dans un macrobloc). Chaque calcul de coût nécessite au moins 16 soustractions avec des valeurs absolues et 15 additions.

De même pour la prédiction Intra16×16 avec quatre coûts pour chaque macrobloc, et chaque coût nécessite 256 soustractions, 17 transformations de Hadamard (sur des blocs 4×4 pixels) et 255 additions au moins. L'algorithme Full Search consiste à calculer tous les coûts pour les deux modes de prédiction. Le mode (les modes) au minimum de coût est sélectionné. Plusieurs autres algorithmes ont été proposés dans le but d'accélérer les calculs [22-23], et cela par sélection des modes de prédiction.

- La Somme des Différences Absolues (SAD):

$$SAD(K,L) = \sum_{r=0}^{15} \sum_{c=0}^{15} abs[I(i+r, j+c) - P(k+r, l+c)] \qquad \textbf{(II.1)}$$

P, I: La luminance du macrobloc prédite et du macrobloc originale ;

(i, j) : Les coordonnées du macrobloc dans l'image I ;

(K, l): Les coordonnées du macrobloc dans l'image P.

Le SAD (Sum of Absolute Difference) est le critère le plus utilisé dans les normes de codage vidéo à cause de sa complexité réduite (nécessite seulement des soustractions et des additions). Le SSE (Sum of Squared Errors), qui représente l'énergie résiduelle, est moins utilisé que le précédent à cause de sa complexité (nécessite des multiplications) et de sa sensibilité au bruit (une grande différence due au bruit sur un pixel de l'image a un grand impact sur la SSE). Le SATD (Sum of Absolute Difference Transform) est largement répandue dans un contexte d'encodage vidéo avec compensation de mouvement : elle représente le coût de codage du résidu grâce à une transformée rapide.

4. Transformation et quantification directes et inverses

Dans un encodeur vidéo y compris la norme H.264/AVC, après l'étape de prédiction, les données organisées sous forme de macroblocs sont transformées et quantifiées, et cela avant l'opération finale de codage et de réarrangement en bitstream. De même dans un décodeur, une transformée inverse est préalable à la reconstruction et l'affichage. Plusieurs transformations sont spécifiées dans la norme H.264 : une transformation de base 4×4, des transformations dérivées 4×4 et 2×2 et une transformation haute 8×8. En effet, dans un codec H.264, des transformations et des quantifications directes et inverses sont appliquées sur des blocs 4×4 pixels de luminance et de chrominance. La transformation est une approximation à l'échelle de la fameuse transformation DCT qui peut être calculée en utilisant une arithmétique entière simple. Une étape de normalisation est incorporée dans les opérations de quantification directe et inverse.

4.1. Développement de la transformée et de la quantification directes

La transformée de base 4×4 utilisée dans la norme H.264/AVC est une approximation d'échelle de la DCT. La transformation et la quantification sont restructurées de telle sorte que la complexité de calcul soit réduite au minimum. Cet objectif est atteint en réorganisant les processus en un noyau de transformation (Core part) et une partie de mise à l'échelle (Scaling part). Considérons un bloc 4×4 pixels, ce bloc est traité par une transformation DCT à deux dimensions suivie par une quantification (la quantification dans ce cas est une simple division par un coefficient de quantification $\mathbf{Q}_{step}$ suivie par un arrondissement du résultat). L'étape suivante est de mettre à l'échelle le processus de quantification par une constante (2^{15}) et de compenser en divisant par la même constante et en arrondissant le résultat .Et enfin, combiner $\mathbf{S}_f$ et le processus de quantification dans $\mathbf{M_f}$, la figure II.13 montre le principe détaillé où : $\boldsymbol{M_f = \frac{S_f.2^{15}}{Q_{step}}}$

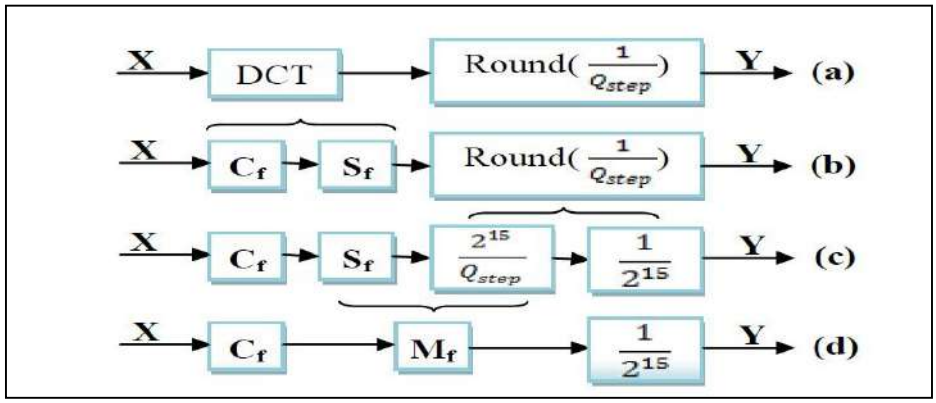


Figure II. 13. *Développement de la transformée et la quantification directes.*

4.2. Développent de la quantification et de la transformée inverses

Considérons l'opération de quantification inverse bien de remise à l'échelle (re-scaling), cette opération est suivie par la transformée inverse IDCT à deux dimensions. De même que la transformée directe, réorganisons la IDCT en un noyau $(\mathbf{C}_i)$ et une matrice $(\mathbf{S}_i)$. L'étape suivante consiste à corriger le processus de remise à l'échelle par une constante (2^6)

et à compenser le résultat final en divisant et en arrondissant. Ensuite, combiner le processus de remise à l'échelle et S_i dans V_i. La figure suivante montre le principe détaillé, avec : $V_i = S_i.2^6.Q_{step}$.

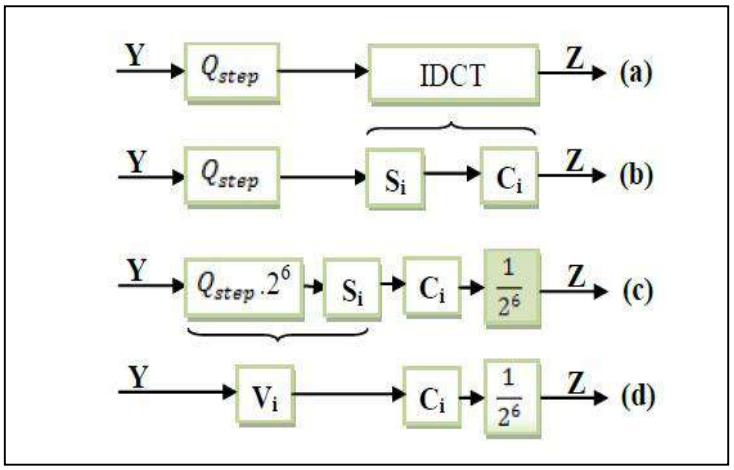


Figure II. 14 : *Développent de la transformée et la quantification inverses.*

4.3. Elaboration des matrices de la transformée directe

Considérons la transformation DCT appliquée sur un bloc X de taille 4×4 [39]

$$Y = A*X*A^T \qquad \textbf{(II.2)}$$

Où (*) indique le produit matriciel, et :

$$A = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \quad a = 1/2 \text{ , } b = \sqrt{1/2\cos(\tfrac{\pi}{8})} = 0{,}653 \text{ et } c=\sqrt{1/2\cos(\tfrac{3\pi}{8})} = 0{,}2706$$

…

Les lignes de la matrice A sont orthogonales et ont des normes unité (les lignes sont orthonormées). L'utilisation de ces coefficients sur un processeur en pratique nécessite des

approximations sur les nombres fractionnaires b et c. Considérons, donc, l'approximation suivant :

$$\mathbf{A_1 = Cf.\ Rf \approx A} \qquad \textbf{(II.3)}$$

Avec : $C_f = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{vmatrix}$ et $R_f = \begin{vmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \\ 1/2 & 1/2 & 1/2 & 1/2 \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \end{vmatrix}$

Le symbole (.) dénote la multiplication élément par élément (Hadamard-Schur produit). Noter aussi que la nouvelle matrice A1 est orthonormée. Cette approximation est choisie pour minimiser la complexité de mise en oeuvre de la transformation (la multiplication par **Cf** ne demande que des additions et des changements binaire) tout en conservant les bonnes performances de compression. La transformée bidirectionnelle devienne :

$$Y = A_1 * X * A_1^T = [C_f . R_f] * X * [C_f^T . R_f^T] \qquad \textbf{(II.4)}$$

Après réorganisation on obtient:

$$Y = [C_f * X * C_f^T] . [R_f . R_f^T] = [C_f * X * C_f^T] . S_f \qquad \textbf{(II.5)}$$

La nouvelle transformation devienne alors :

$$W = C_f X C_f^T \qquad \textbf{(II.6)}$$

Les valeurs absolues des coefficients de la matrice C_f sont soit de 1 ou 2. Ainsi, l'opération de transformation représentée par l'équation (II.7) peut être calculée en utilisant

des additions signées et de décalage à gauche, et cela dans le but d'éviter les multiplications par des nombres réels. Comme montrer dans la Figure(II.13), la matrice [$\mathbf{R_f.R_f}^{\mathbf{T}}$] est combinée avec l'équation de quantification. Après quelques simplifications et réarrangement, les formules de post-mise à l'échelle et de quantification seront données par :

$$Z_{ij} = round(W_{ij}.\frac{MF}{2^{qbits}}) \qquad \textbf{(II.7)}$$

Avec *qbits* = 15 + (QP DIV 6)

QP est le paramètre de quantification qui permet à l'encodeur de contrôler avec précision et souplesse le compromis entre le débit et la qualité, il peut prendre n'importe quelle valeur entière de 0 à 51. (Z_{ij}) est un élément dans la matrice des résultats du processus de quantification. (*MF*) est un facteur de multiplication qui dépend de QP et la position de l'élément dans la matrice.

De même pour la transformation inverse (IDCT) à deux dimensions appliquée sur un bloc de taille 4×4, dont la formule initiale est donnée par :

$$\mathbf{Z = A^T * Y * A} \qquad \textbf{(II.8)}$$

Cette transformation est combinée avec la quantification inverse (multiplication par QP) pour avoir des opérations simples.

5. Prédiction du mode de prédiction

Le choix du mode de prédiction en codage intra pour les blocs 4x4 doit être signalé au décodeur, ceci pourrait potentiellement exiger un grand nombre des bits. Cependant, les modes intra du voisin bloc 4x4 sont fortement corrélés. [24]

Par exemple si les blocs A et B (Figure II.12) étaient codé précédemment avec mode DC le mode le plus probable donc pour le bloc C c'est le mode DC.

Pour chaque bloc courant C l'encodeur et le décodeur calculent le mode le plus probable. Si A et B sont tout le deux codés en 4x4 le mode le plus probable est le minimum entre eux. Si non le mode le plus probable est le mode DC. L'encodeur envoie un drapeau pour chacun des blocs 4x4, il utilise le mode le plus probable. L'encodeur emploie une condition pour le mode le plus probable avec un drapeau qui signale le mode de prédiction, Si le mode le plus probable est le même que le mode de prédiction le drapeau envoie '1' et seulement un bit est nécessaire pour signaler le mode de prédiction. Quand le mode le plus probable et le mode de prédiction sont différent le drapeau envoie '0' et 3 bits additionnels sont exigés au signal intra prédiction. L'encodeur doit dépenser 1ou 4 bits pour représenter le mode prédiction.

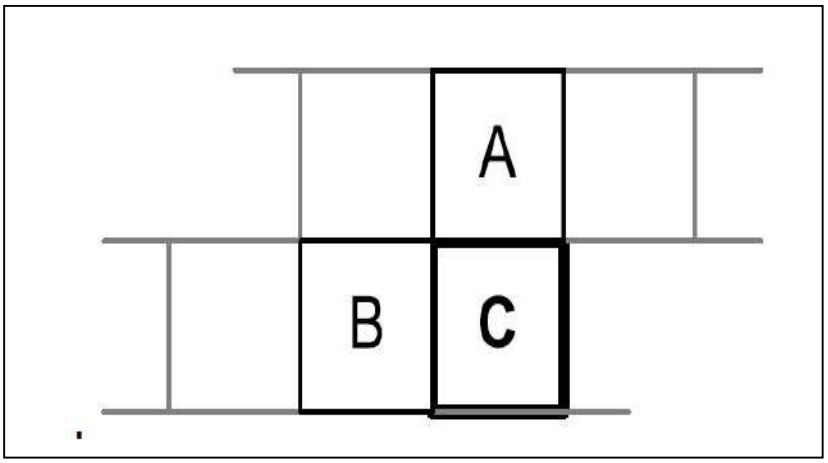


Figure II. 15: *Intra blocs 4x4 codés adjacents.*

6. Conclusion

Dans ce chapitre on a présenté le codage intra prédiction, la transformée et la quantification de l'encodeur H.264. Le codage intra utilise neuf modes de prédiction. L'avantage de ce dernier est de réduire les redondances spatiales. Dans la méthode proposée pour le codage, les nombres de modes de prédiction pour chaque bloc 4x4 sont choisis de

manière adaptative. Basé sur des similitudes des pixels de référence. On a présenté également le mode le plus probable (MPM) en direction du mode de prédiction de blocs voisins qui ont différents poids selon leurs positions.

CHAPITRE III : IMPLEMENTATION ET RESULTATS

1. Introduction

Dans ce chapitre, on présente la partie implémentation de notre travail. Pour implémenter une partie de l'encodeur H.264 (codage intra image), dont le but est de tester et valider notre algorithme.

2. Outil de développement

L'environnement de travail est basé sur le langage de calcul scientifique MATLAB (**MAT**rix **LAB**oratory) pour évaluer et valider les résultats du codage intra prédiction. On a utilisé la version R2012b de MATLAB.

3. Prétraitement

Dans cette partie nous allons présenter les étapes à suivre pour faire le prétraitement de l'image source ; en commençant par l'étape de lecture d'image, puis on la convertit au format YUV ou bien YCRCB (Y : composants de luminance. Et CrCb : composants de chrominance), par la suite on fait l'échantillonnage de type 4 :2 :0 utilisé par la norme H.264 sur les composants de chrominance.

Pour faire le codage Inta sur la luminance Y, on a le choix entre le partitionnement de l'image en blocs 16x16 ou bien 4x4. La figure III.1 montre l'organigramme général à suivre dans l'étape de prétraitement.

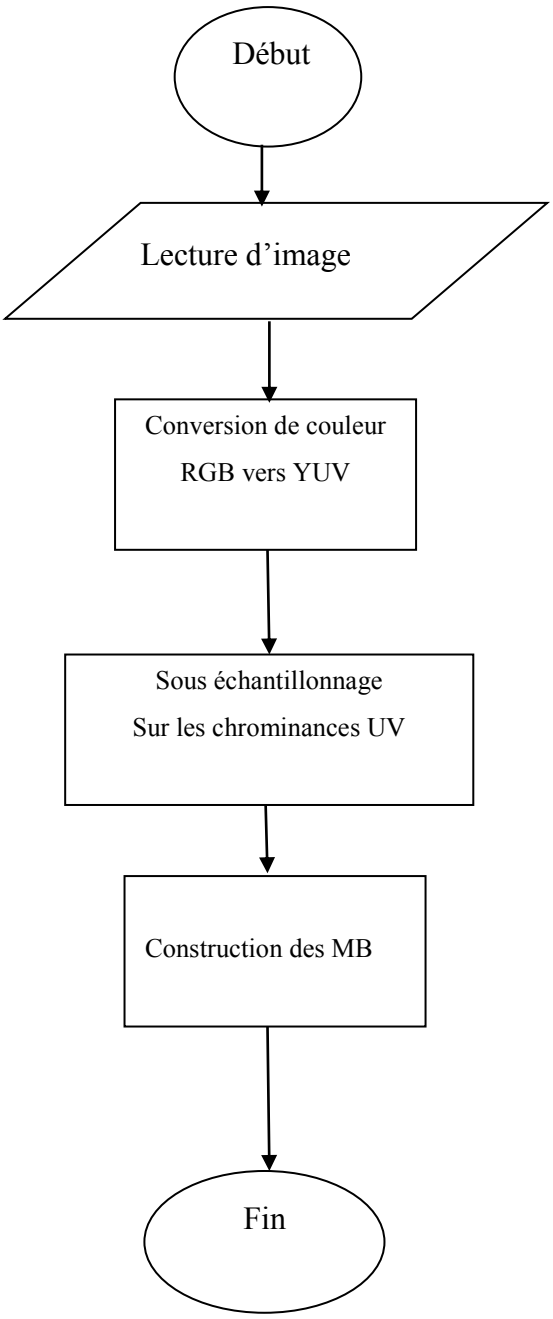


Figure III. 1. *Organigramme de Prétraitement.*

3.1. Lecture d'image

Cette opération est réalisée à l'aide de l'instruction imread du matlab, cette instruction permet de récupérer une image de format JPG dans une variable selon la syntaxe suivante :

I= imread ('nom et emplacement de l'image');

3.2. Conversion de couleur (RGB ver YUV)

Dans cette étape, l'espace de couleur RGB et converti en YUV selon la formule (I.1).

On a utilisé le code suivant :

```
R = I(:,:, 1);
G = I(:,:,2);
B = I(:,:,3);
[x y]=size(R);
for i=1:x
    for j=1:y
        Y(i,j) = 0.299*R(i,j)+ 0.587*G(i,j) +0.114*B(i,j;
        U(i,j) = 0.564*(B(i,j) - Y(i,j) );
        V(i,j) = 0.713*(R(i,j) - Y(i,j) );
    end
end
```

3.3. Sous-échantillonnage sure U et V

Cette partie concerne les composantes U et V pour produire le format 4 :2 :0, Les chrominances sont deux fois moins de lignes et deux fois moins de colonnes par rapport à la luminance Y.

Le code suivant est utilisé pour réaliser l'opération de décimation

% sous échantillonnage de U et V

```
c1=1;
for i=1:2:x
    c2=1;
    for j=1:2:y
              Ue(c1,c2)=U(i,j);
              Ve(c1,c2)=V(i,j);
         c2=c2+1;
    end
    c1=c1+1;
End
```

3.4. Construction des macro-blocs (MB)

La norme H.264 utilise des MB de taille fixe 16x16 ces MB peuvent être a leurs tours divisés en bloc 4x4.

4. Prédiction intra

Les macro-blocs obtenues en prétraitement seront divisés pour former des blocs soit de 4x4 ou 16x16 de la composante Y.

L'organigramme de la figure (III.2) montre le schéma général pour établir la prédiction intra.

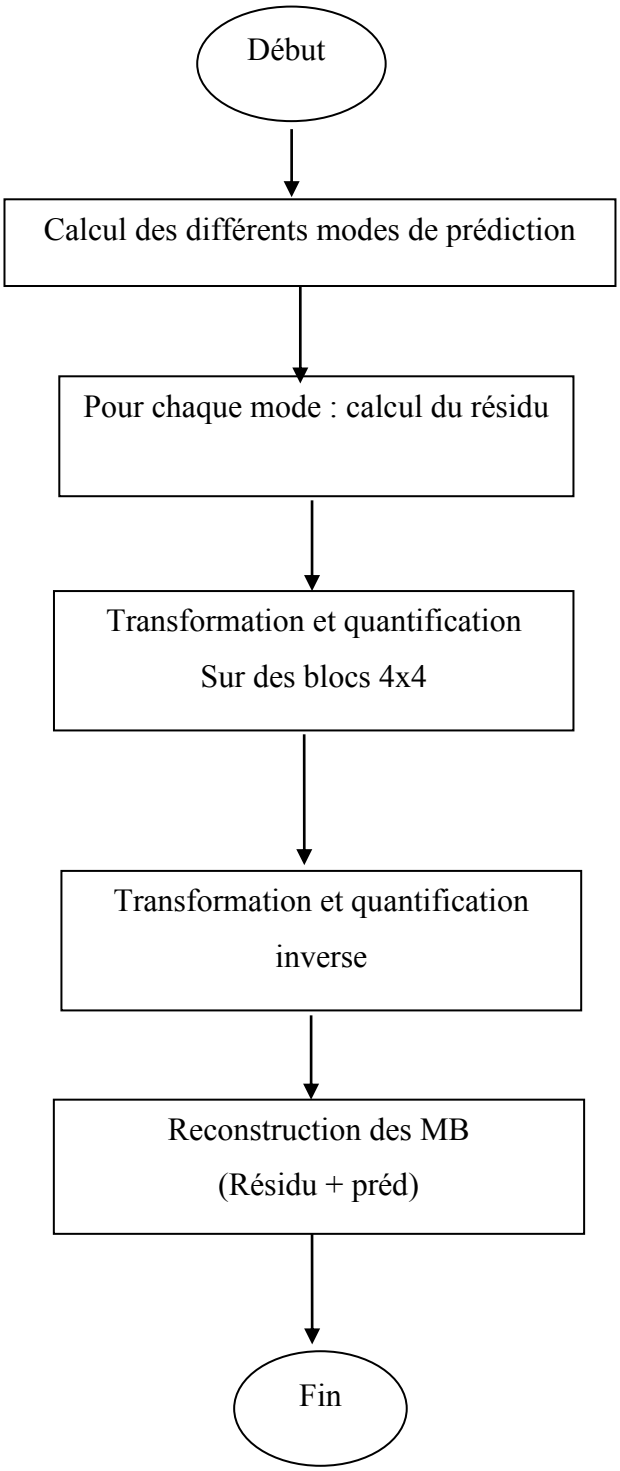


Figure III. 2. *Organigramme de prédiction.*

Lorsque le type Intra16x16 est utilisé, un seul mode de prédiction est appliqué pour encoder l'ensemble du macrobloc. Quatre modes de prédiction sont définis dans la norme H.264 et l'objectif du codeur est de choisir le mode le plus adapté au macrobloc courant. Ces quatre modes sont déjà illustrés dans le chapitre précédant. Nous pouvons noter que trois de ces modes sont associés à des directions spécifiques (verticale, horizontale et plane). Le dernier mode c'est le mode (DC)

La prédiction Intra16x16 s'avère très efficace pour coder les régions où la luminance varie doucement.

Par exemple si on veut prédire le bloc ci-dessous d'une image source de taille 16x16 on applique l'algorithme de prédiction avec les quatre modes vus précédemment.

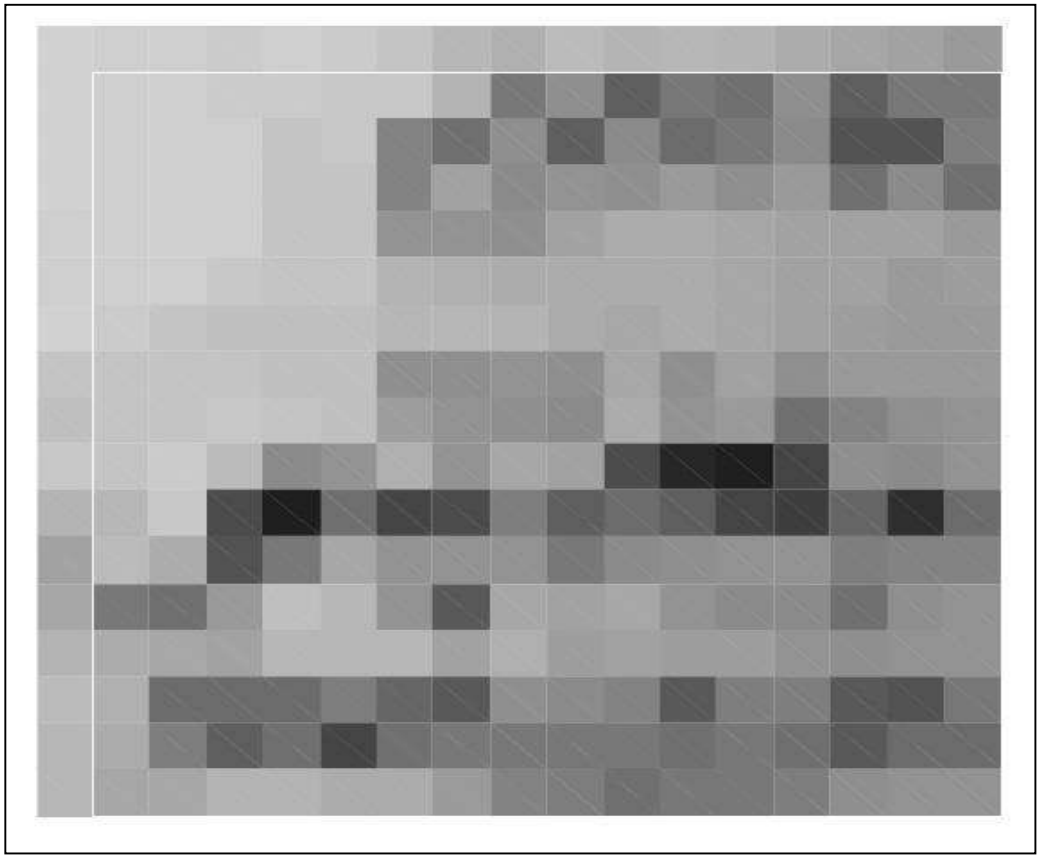


Figure III. 3. *Bloc initial de taille 16x16*

Les résultats de prédiction pour chaque mode sur un bloc 16x16 ainsi que la somme de l'erreur absolue (SAD) sont représentés sur la figure III.4. Le meilleur mode de prédiction est choisi selon le critère SAD, qui serait dans notre cas le mode plane (mode 4), car il offre le plus petit SAD, soit 2539.

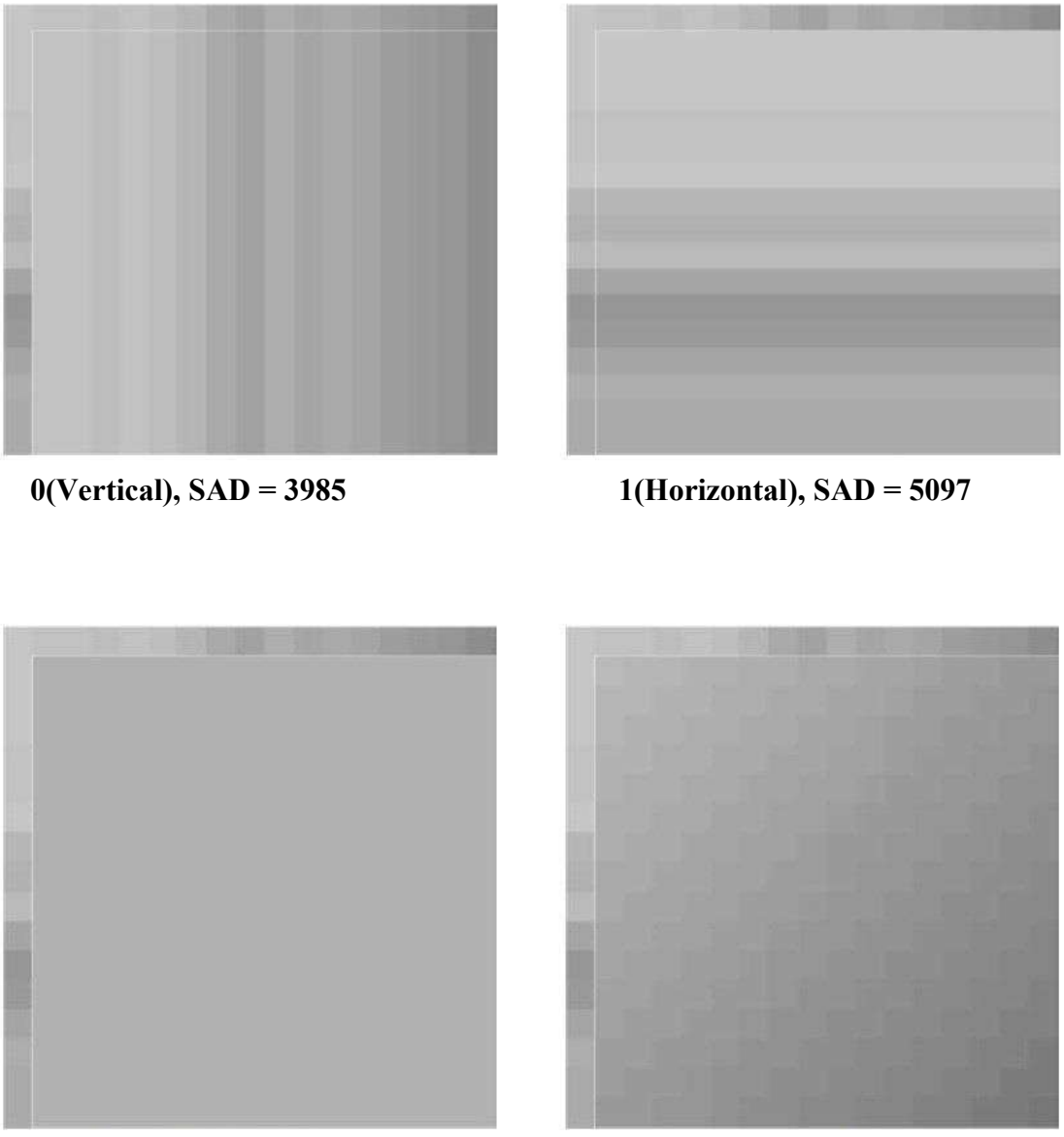


Figure III. 4. *La prédiction obtenue pour les quatre modes (16x16).*

Dans le type Intra4x4, le macrobloc est divisé en seize blocs de 4×4 pixels et chaque bloc est encodé individuellement. Neuf modes de prédiction sont fournis par la norme et l'objectif du codeur est de sélectionner le mode le plus adapté au bloc courant

Ces neuf modes sont représentés dans le chapitre précédent. Nous pouvons constater que huit de ces modes caractérisent des modèles de prédiction directionnels. Chaque valeur prédite du bloc courant est déterminée à partir d'une combinaison linéaire entre les pixels situés dans les blocs voisins. Ces combinaisons linéaires sont définies dans la norme en fonction d'une direction spécifique. Seul le mode DC n'est pas associé à une direction et consiste juste à calculer la moyenne des pixels contenus dans les deux blocs voisins.

Par exemple si on veut prédire le bloc de la figure (III.5) d'une image source de taille 4x4 on applique l'algorithme de prédiction avec les neuf modes vus précédemment.

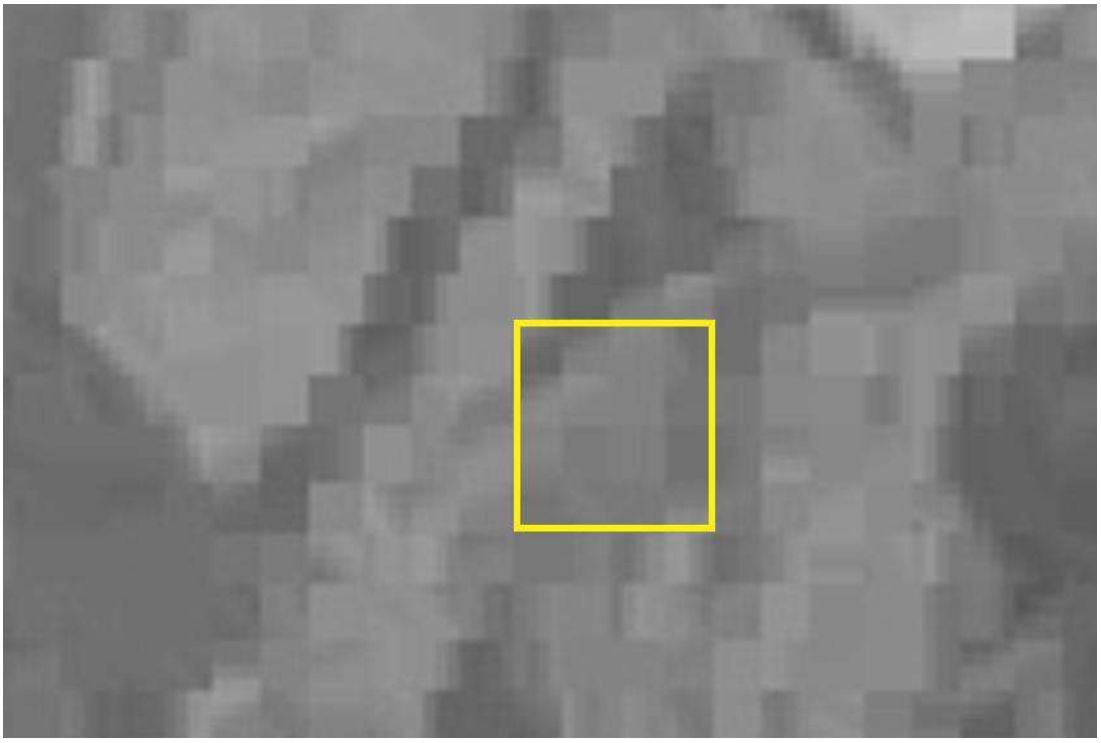
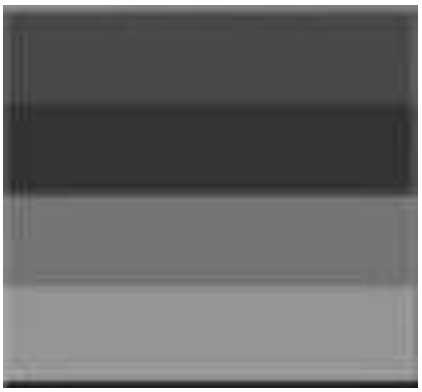


Figure III. 5. *Bloc initial de taille 4x4*

Les résultats de prédiction pour chaque mode sur un bloc 16x16 ainsi que la somme de l'erreur absolue (SAD) sont représentés sur la figure (III.6). Le meilleur mode de prédiction est choisi selon le critère SAD, qui serait dans notre cas le mode 8, car il offre le plus petit SAE, soit 203



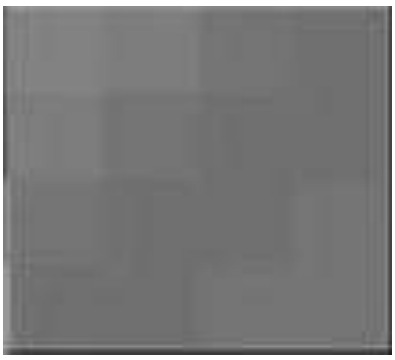
0 (Vertical), SAD = 317



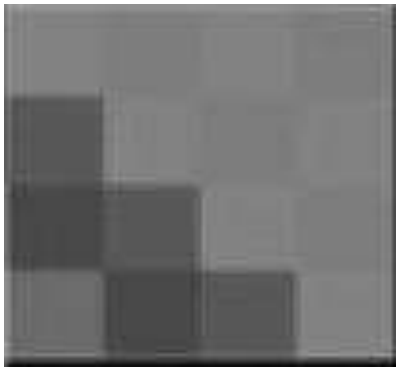
1 (Horizontal), SAD = 401



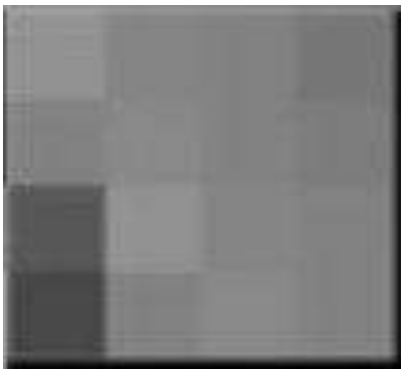
2 (DC), SAD = 317



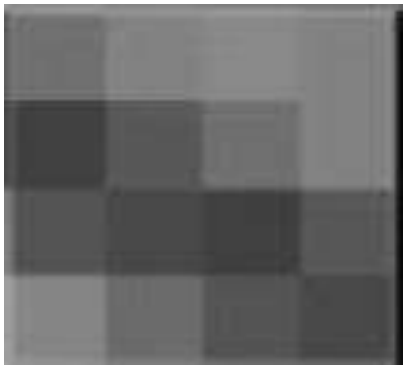
3 (diag bas-gauche), SAD = 350



4 (diag bas droit), SAD = 466



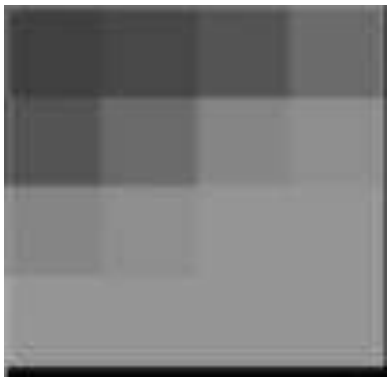
5 (Vertical droit), SAD = 419



6 (Horizontal bas), SAD = 530



7 (Vertical gouche), SAD = 351



8 (Horizontal haut), SAD = 203.

Figure III. 6. *La prédiction obtenue pour les neuf modes (4x4).*

5. Prédiction du mode de prédiction

Dans cette partie on va estimer le mode le plus probable (MPM) ; si on a les blocs adjacents A et B du bloc courant C comme nous montre la figure(III.7), on peut estimer le MPM.

Pour faire la prédiction du MPM, on fait appelle à une partie du code qui nous retourne une matrice contenant les modes les plus probables pour chaque bloc quel que soit la taille du bloc utilisée. (16x16 ou bien 4x4).

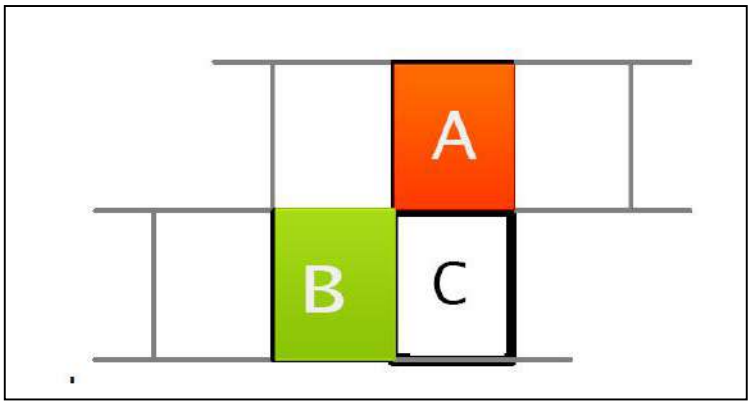


Figure III. 7. *Les blocs adjacents*

L'organigramme de la figure (III.8) suivant montre comment l'encodeur estime le mode le plus probable.

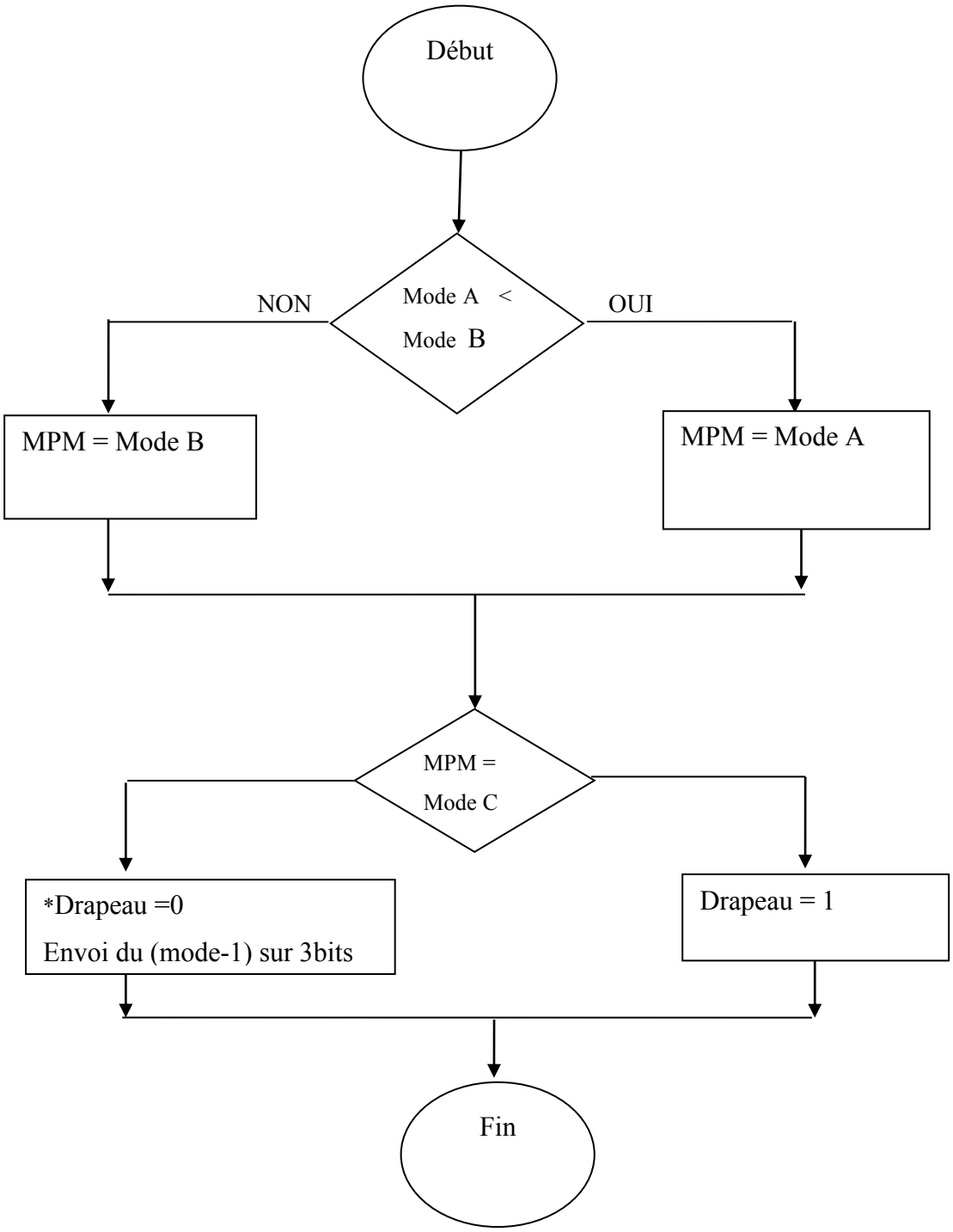


Figure III. 8. *Organigramme de prédiction de mode prédiction*

6. Tests et résultats

L'ensemble des organigrammes vus et d'autres sont réunis dans notre travail, et nous a permis d'avoir les résultats ci-après.

La figure (III.9) montre l'image originale à appliquer à l'entrée de l'encodeur.

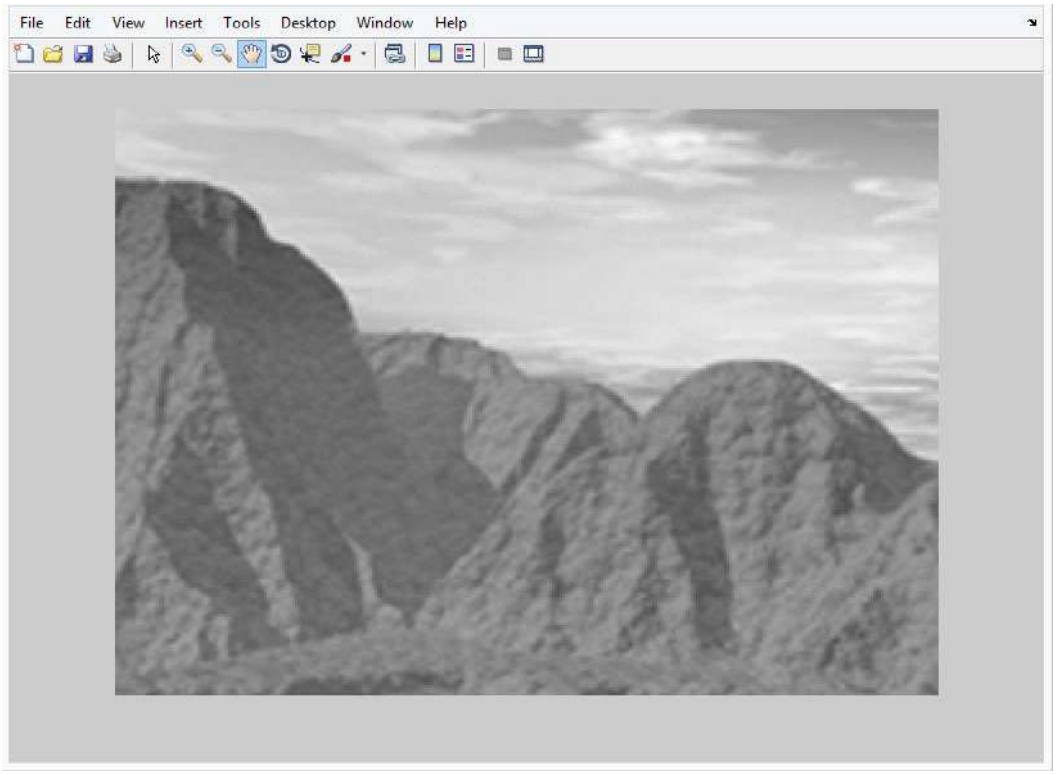


Figure III. 9. *Image originale*

Cette image est partitionnée en MB de 16x16, ensuite chaque MB est divisé en bloc de 4x4. Après le calcul des différents modes de prédiction et le choix optimal pour chaque bloc, les résultats sont regroupés dans l'image de la figure (III.10).

Dans cette image, on remarque l'effet des blocs très clairement à cause du partitionnement initial de l'image.

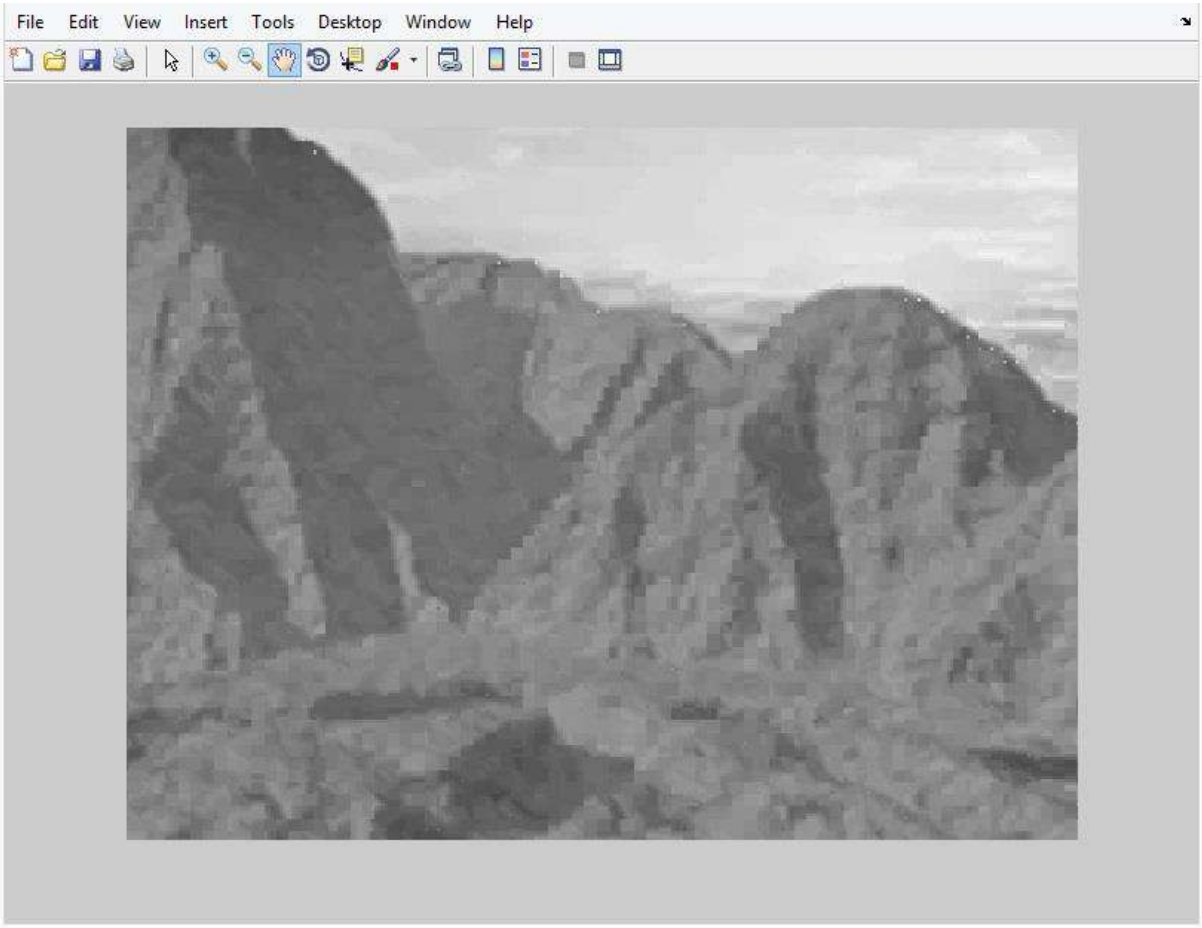


Figure III. 10 . *Image des blocs de prédiction (4x4)*

Après avoir fait la prédiction de l'image originale, l'image résiduelle n'est juste autre que la soustraction entre l'image originale et l'image prédite, comme le montre la figure (III.11)

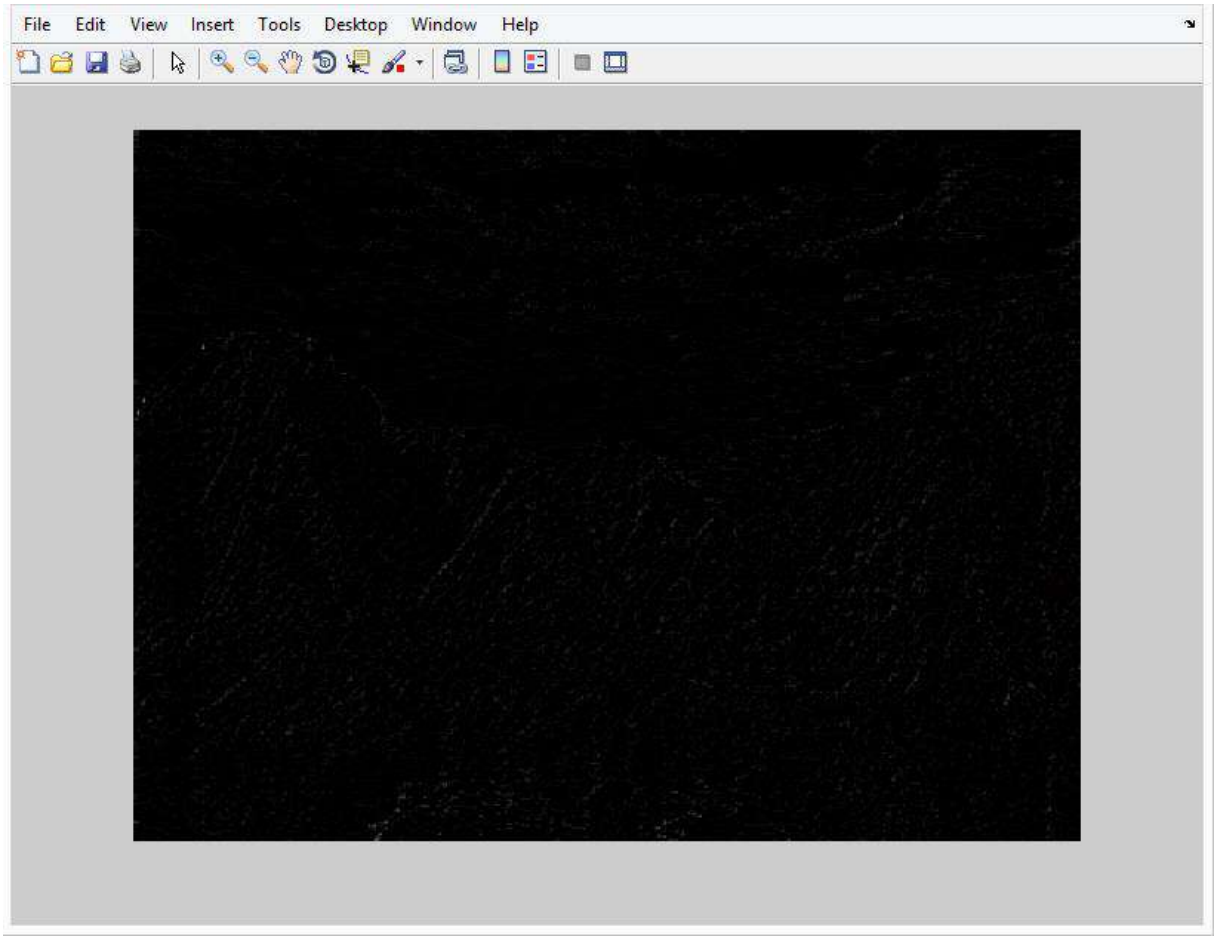


Figure III. 11. *Image résiduelle des blocs 4x4*

Les zones noires dans la figure III.11 indiquent qu'il n'y a pas une erreur entre l'image originale et l'ensemble des blocs de prédiction, tandis que certaines zones grises ou blanches indiquent là où il y a une différence significative.

La suite de la chaine de traitement comporte un cycle constitué de quatre parties (transformation, quantification, quantification inverse transformation inverse). La figure (III.12) montre un exemple de calcul sur un bloc 4x4.

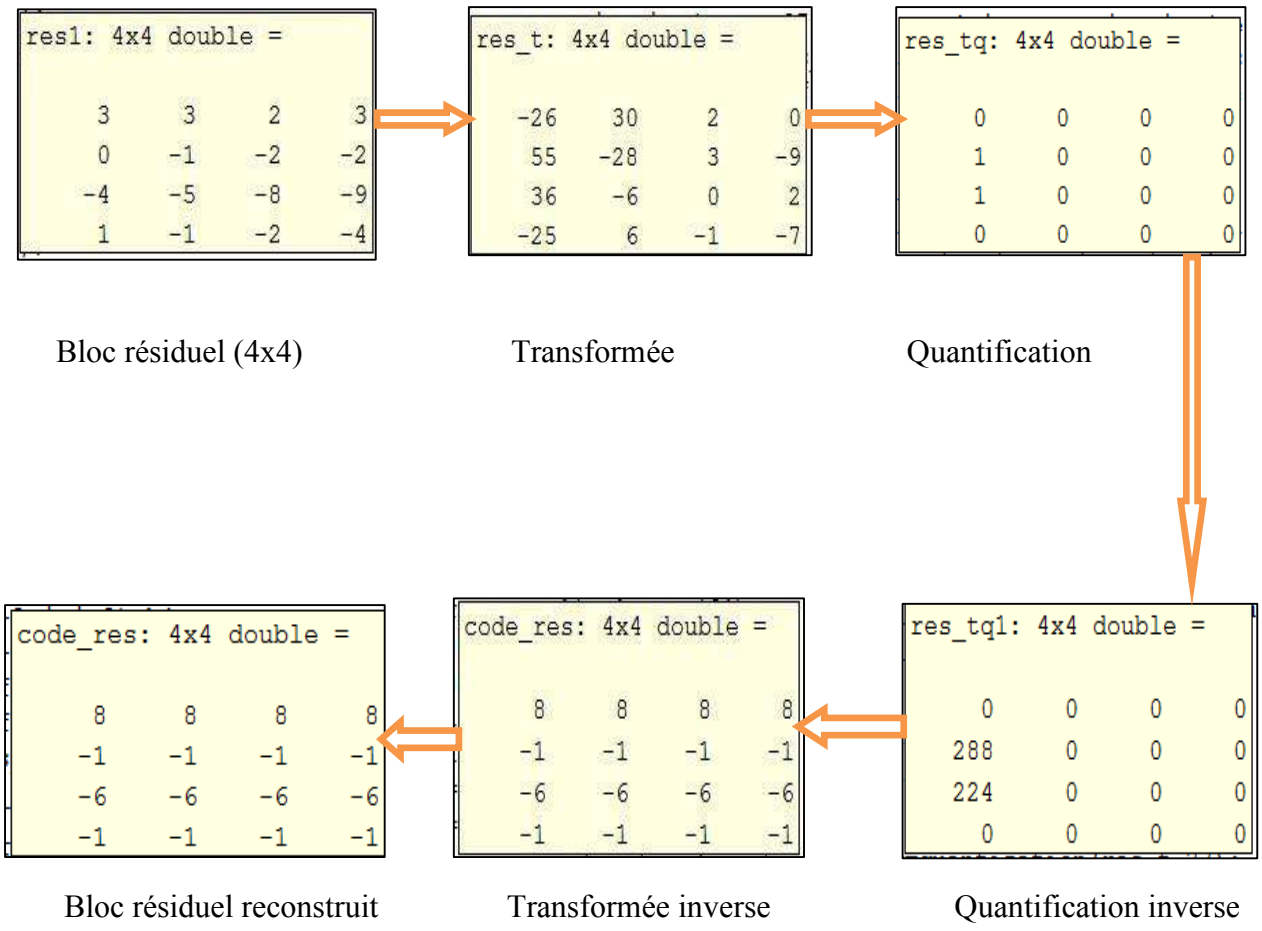


Figure III. 12. *Bloc résiduel (4x4) après cycle T,Q,Q^{-1},T^{-1}.*

Après la reconstruction du résidu (après cycle T,Q,Q^{-1},T^{-1}), on peut former une image en additionnant les blocs de prédiction et le résidu reconstruit, on obtient l'image de la figure (III.13).

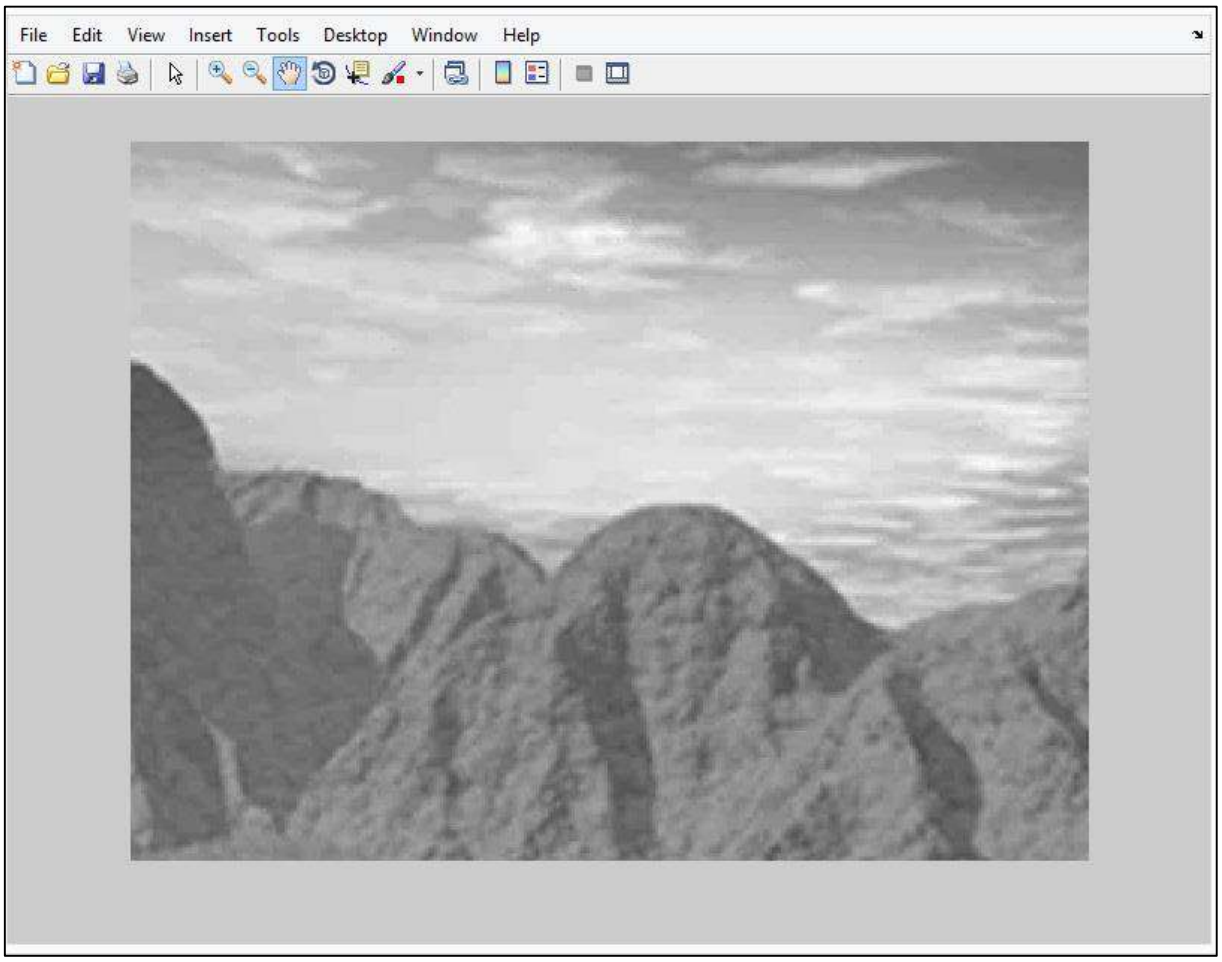


Figure III. 13. *Image reconstruite*

Cette image ressemble beaucoup à l'image originale, ceci est dû à la compensation de l'erreur par l'ajout du résidu.

Pour mesurer la qualité de reconstruction, on a utilisé une métrique basée sur le PSNR, l'exemple précédent possède un PSNR de l'ordre 39.3316.

Dans le cas de prédiction intra 16x16, on suit les mêmes étapes vues dans le cas 4x4, les différents résultats obtenus sont illustrés sur les figures III.14, III.15, III.16.

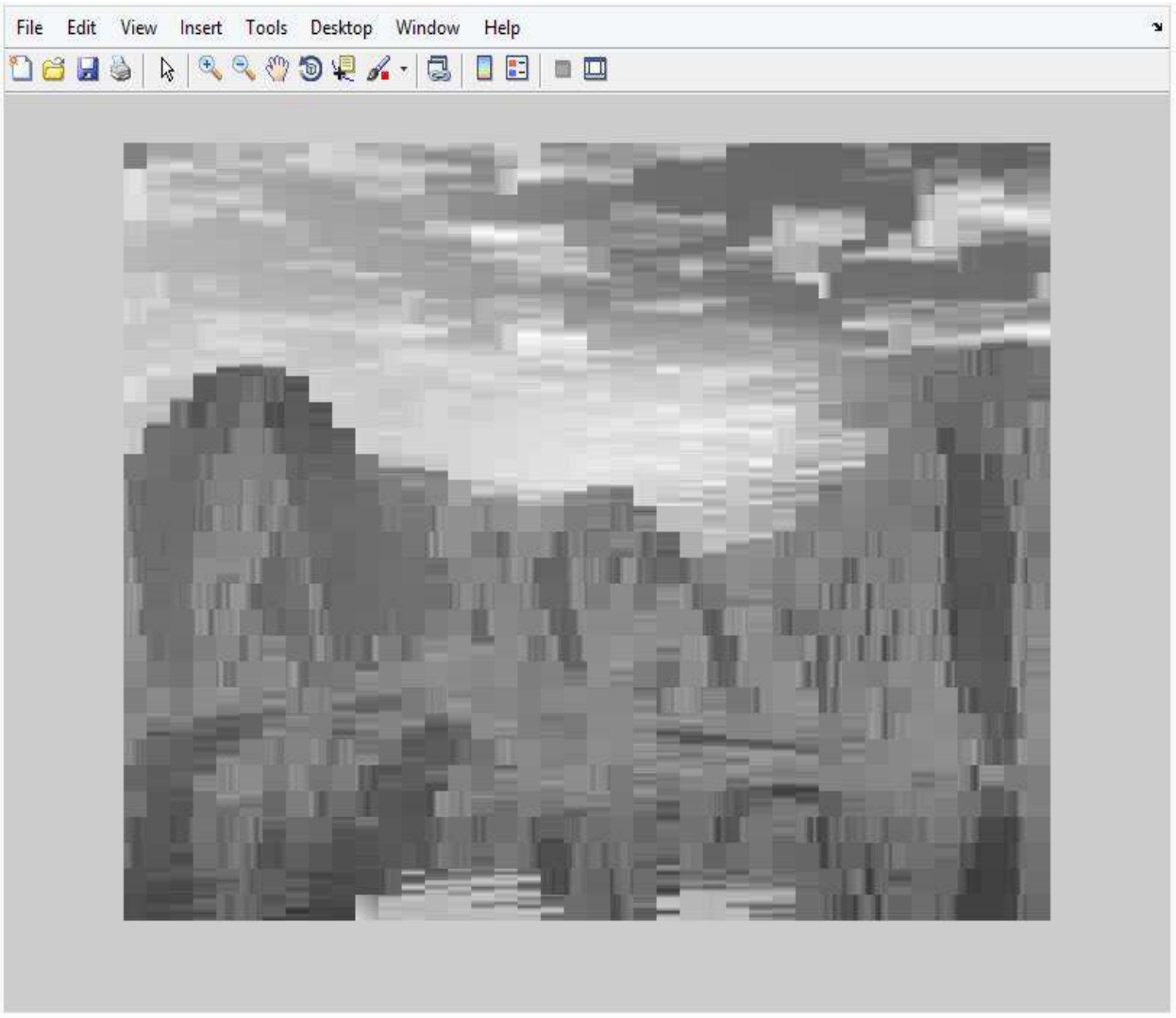


Figure III. 14.*image predite des blocs (16x16)*

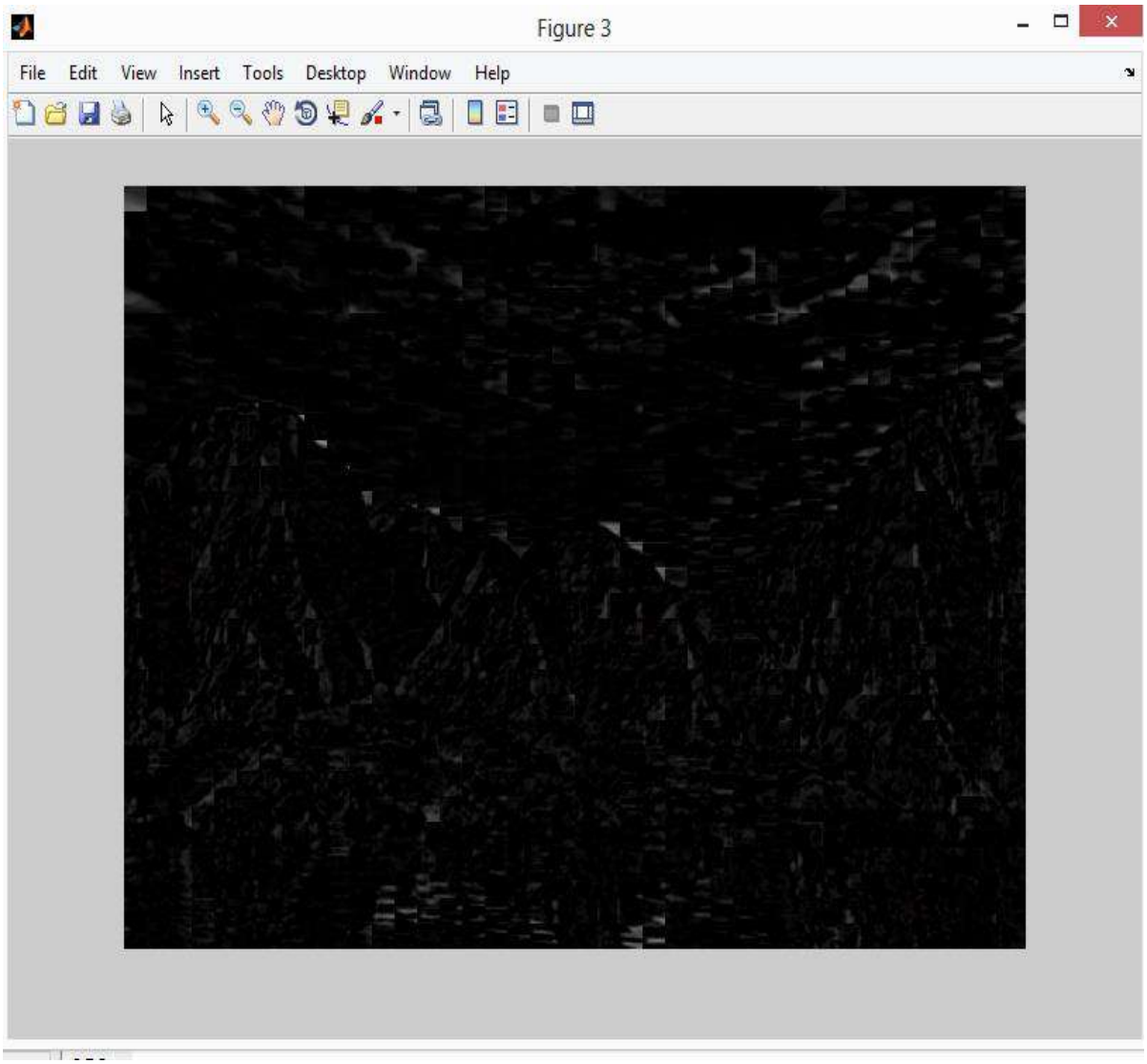


Figure III. 15. *Image résiduelle des blocs (16x16).*

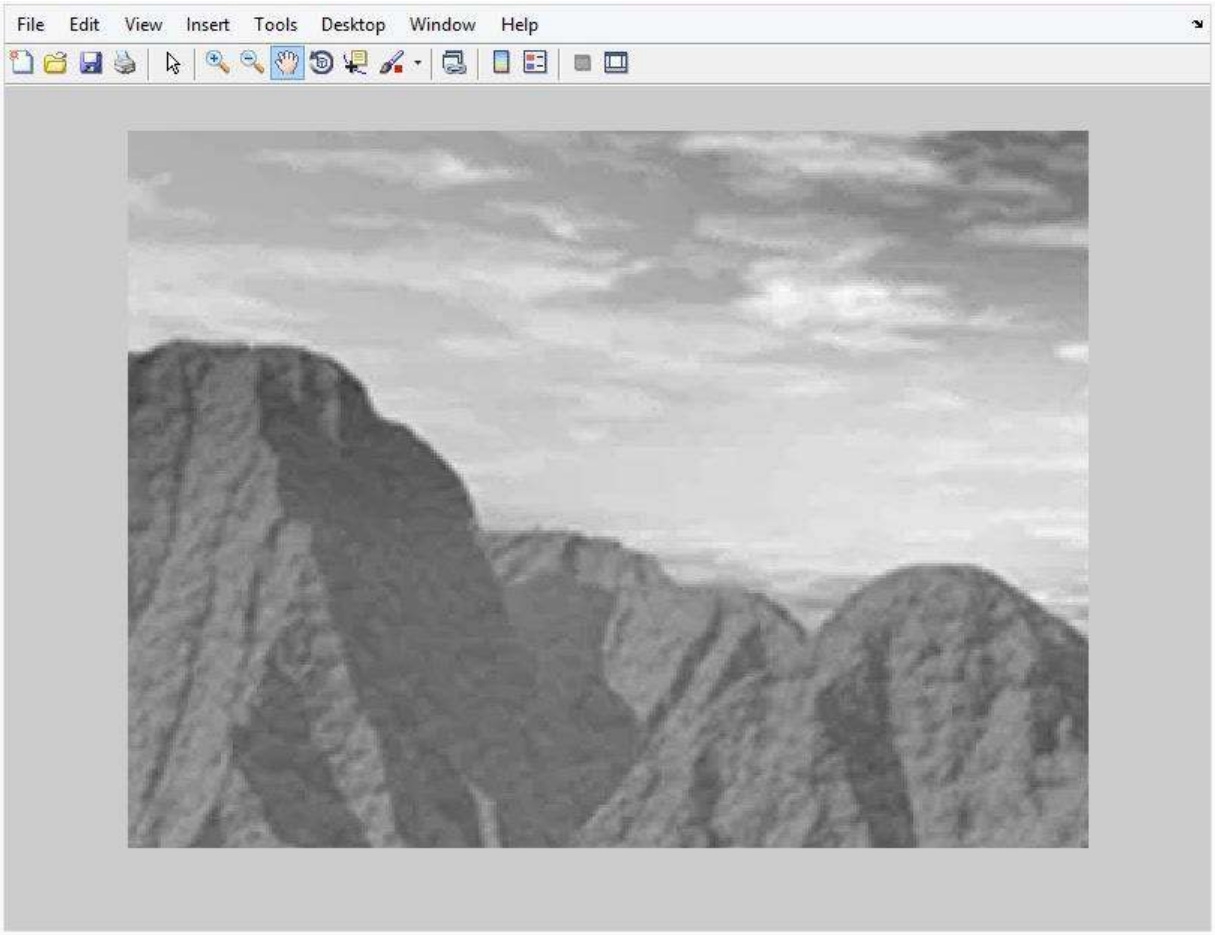


Figure III. 16. *Image reconstruite des blocs (16x16)*

Le PNSR pour cette image est de l'ordre de 39.4390, selon ce critère, la prédiction 16x16 est meilleure que la prédiction 4x4, ce qui affirme que La prédiction Intra16x16 est très efficace pour coder les régions où la luminance varie doucement (le cas de notre image).

Le tableau (III.1) récapitule les résultats de test sur un ensemble d'images.

TABLEAU III. 1. *Résultats de test*

Image	PSNR (blocs 16x16)	PSNR (blocs 4x4)
	39.3316	39.4390
	38.7697	38.9376
	43.1803	42.8189

7. Conclusion

Ce chapitre a été consacré pour la partie implémentation de l'algorithme de codage intra image. Nous avons appliqué la prédiction intra sur l'image source dans le but d'éliminer les redondances spatiales. Nous avons commencé par la prédiction 4x4 qui possède neuf modes différents (relativement complexe par rapport à la prédiction 16x16), ensuite, nous avons repris le même travail par des blocs de tailles 16x16. Les résultats trouvés montrent que le choix de la taille des blocs à l'intérieur d'un MacroBloc est relatif au contenu de l'image considérée. Nous avons également implémenté certains blocs de l'encodeur H.264 à savoir la transformation, la quantification, la quantification inverse et la transformation inverse.

Conclusion générale

Parmi les nouveautés de la norme H.264 est l'utilisation d'une nouvelle prédiction Intra appliquée dans le domaine spatial et non pas dans le domaine fréquentiel, par l'encodeur et le décodeur à la fois. Le temps d'exécution de ce module est remarquable par rapport aux normes précédentes, il est estimé autour de 20% pour le module Intra. Pour ces raisons, nous avons concentré nos efforts à la conception et l'implantation de ce module

Pour faire une telle implémentation, nous avons d'abord fait une étude générale sur l'encodeur H.264 particulièrement la partie qui concerne le codage intra. Ensuite, nous avons présenté les principes de la prédiction spatiale avec tous ses modes.

Nous avons implémenté le module Intra qui regroupe la prédiction, la transformation, la quantification, la quantification inverse et la transformation inverse. Une autre tâche qui semble importante consiste à la confirmation de l'efficacité de notre implémentation par l'utilisation d'une d'image réelle. Dans ce cas une étude comparative, des performances et des qualités des images (PSNR), serait nécessaire toute en comparant les résultats obtenus.

Enfin notre travail, n'est qu'un point de départ d'une intégration effective de l'implémentation de l'encodeur H.264. Et pourquoi pas l'implémentation en temps réel de la chaine de traitement sous un processeur dédié tel que le DSP (*Digital Signal Processor*).

Bibliographies

[1] H.264/MPEG-4 AVC
[En ligne]. http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC [Consulté le 3 /3/2014].

[2] Nouvelles possibilités dans le secteur de la vidéosurveillance.
2008 Axis Communications AB.

[3] Ahmed BEN ATITALLAH, 'Etude et Implantation d'Algorithmes de Compression d'Images dans un Environnement Mixte Matériel et Logiciel', Thèse doctorat, Université de Bordeaux1, Juillet 2007.

[4] Antoine ROBERT, 'Transformées orientées par blocs pour le codage vidéo hybride', Thèse doctorat, Ecole Nationale Supérieure des Télécommunications, France, Février 2008.

[5] video :definition et explications

[En ligne] http://www.techno-science.net/?onglet=glossaire&definition=7376. [Consulté le 3 /4/2014].

[6] Fawzi GHOZZI, 'Optimisation d'une bibliothèque de modules matériels de traitement d'images'.Thèse doctorat, Université de Bordeaux1, 2003.

[7] Aurélie Martin, 'Représentations parcimonieuses adaptées à la compression d'images', Thèse doctorat, IRISA Université De Rennes1, Avril 2010.

[8] planete numerique
[En ligne] http://www.planetenumerique.com/?Decouverte-du-MPEG-4-part-10-AVC.
. [Consulté le 3 /4/2014].

[9] Fabrice URBAN, 'Implantation optimisée d'estimateurs de mouvement pour la compression vidéo sur plates-formes hétérogènes multi-composants', Thèse doctorat, Institut National Des Sciences Appliquées DeRennes, Décembre 2007.

[10] Clément CHASTAGNOL, 'Stéganographie en domaine vidéo compressé', mémoire de projet de Fin d'études, Ecole Centrale de Lyon, THALES Communications, 2009.

[11] Site web http://www.compute-rs.com/fr/conseil-2037056.htm.

[12] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG. Draft of Version 4 of H.264/AVC

[13] Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding), 2003.

[14] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG. (ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding-Reference Software Manual), 2009.

[15] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, A. Luthra, 'Overview of the H.264/AVC Video Coding Standard', IEEE Transactions On Circuits And Systems For Video Technology, Vol. 13, No. 7, pp. 560-576, 2003.

[16] Iain E. G. RICHARDSON, 'The H.264 Advanced Video Compression Standard', Livre, Edition WILEY, Second Edition, 2010.

[17] K. Messaoudi, S. Toumi, E. Bourennane, 'Proposal and study for an architecture hardware/software for the implementation of the standard H264', CISA'08, in: American Institute of Physics (AIP), Vol. 1019 No. 1, pp. 536-540, 2008.

[18] Iain E. G. RICHARDSON, 'H.264 and MPEG-4 Video Compression - Video Coding for Next-generation Multimedia', The Robert Gordon University, Aberdeen, UK, Edition WILEY, 2003.

[19] Ralf Schäfer, Thomas Wiegand, Heiko Schwarz, 'H.264/AVC la norme qui monte', UER– RevueTechnique, pp. 1-10, 2003.

[20] Guillaume LAROCHE, 'Modules de codage par compétition et suppression de l'information de compétition pour le codage de séquences vidéo', Thèse doctorat, Ecole Nationale Supérieure des Télécommunications, France, 19 Mai 2009.

[21] M. Horowitz, A. Joch, F. Kossentini, S. Member, S. Hallapuro, 'H.264/AVC Baseline Profile Decoder'

[22] A. Elyousfi, A. Tamtaoui, E. Bouyakhf, 'Fast Mode Decision Algorithm for Intra prediction in H.264/AVC', International Journal of Computer Science and Network Security IJCSNS, Vol. 7 No. 1,pp. 356-364, 2007.

[23] J. S. Park, H. J. Song, 'Selective Intra Prediction Mode Decision for H.264/AVC Encoders', International Journal of Applied Science, Engineering and Technology, pp. 214-218, 2006.

[24] Improved Intra Prediction of H.264/AVC (Mohammed Golam Sarwer and Q. M. Jonathan Wu) *University of Windsor Canada*

ANNEXES

1. Codes source de l'implémentation:

Intra 4x4

```
%programme intra 4x4.
clear all
close all
clc
% lecture d'image
%I= imread('io.jpg');%
I= imread('images.jpeg');%
% Extraction des bandes d'image
R = I(:,:,1);
G = I(:,:,2);
B = I(:,:,3);
% les dimensions d'image
[x y]=size(R);
%transformation vers l'espace YUV
for i=1:x
    for j=1:y
        Y(i,j)   =   0.299*R(i,j)   +   0.587*G(i,j)   +
0.114*B(i,j)    ;
        U(i,j) = 0.564*(B(i,j) - Y(i,j) );
        V(i,j) = 0.713*(R(i,j) - Y(i,j) );
    end
end
%affichage des composants de YUV
figure(1)
imshow(Y)
%Y=Y1(1:40,1:40);
%[x y]=size(Y)
%figure
%imshow(U)
%figure
%imshow(V)
% sous echantillonnage de U V
```

```
c1=1;
for i=1:2:x
    c2=1;
    for j=1:2:y
       %
Ue(c1,c2)=round(U(i,j)+U(i,j+1)+U(i+1,j)+U(i+1,j+1)/4);
       Ue(c1,c2)=U(i,j);

%Ve(c1,c2)=round(V(i,j)+V(i,j+1)+V(i+1,j)+V(i+1,j+1)/4);
        Ve(c1,c2)=V(i,j);
         c2=c2+1;
    end
    c1=c1+1;

end
%------------l'étape de prédiction----------------------
file=fopen('file.txt','w');
[height,width]=size(Y);
fprintf(file,'%d\n',width/4);
fprintf(file,'%d\n',height/4);

Y=double(Y);
%prédication la prem mb
b=mode4x4DC([],[]);
res=Y(1:4,1:4)-b;
fprintf(file, '%d\n',2);
fprintf(file, '%d\n',int8(res));
fprintf(file, '%d\n',sum(sum(abs(res))));

Yb(1:4,1:4)=b;
res_t=integer_transform(res);
res_tq=quantization(res_t,27);
res_tq1=inv_quantization(res_tq,27);
 code_res=inv_integer_transform(res_tq1);
 code_res=round(code_res/64);
YS(1:4,1:4)=b+code_res;
%prédication de prémier ligne
for j=5:4:y
    V=YS(1:4,j-1);
```

```
    b1=mode4x4H(V);
    b2=mode4x4DC([],V);
    b3=mode84x4([0;V]);
    res1=Y(1:4,j:j+3)-b1;
    res2=Y(1:4,j:j+3)-b2;
    res3=Y(1:4,j:j+3)-b3;
    s1=sum(sum(abs(res1)));
    s2=sum(sum(abs(res2)));
    s3=sum(sum(abs(res3)));
    err=[ s1 s2 s3 ];
    terr=sort(err);
   switch terr(1)
     case s1,  res_t=integer_transform(res1);
               res_tq=quantization(res_t,27);
               res_tq1=inv_quantization(res_tq,27);
               code_res=inv_integer_transform(res_tq1);
               code_res=round(code_res/64);

YS(1:4,j:j+3)=b1+code_res;Yb(1:4,j:j+3)=b1;
               fprintf(file, '%d\n',1);
               fprintf(file, '%d\n',int8(res1));
               fprintf(file, '%d\n',sum(sum(abs(res1))));

     case s2,   res_t=integer_transform(res2);
                res_tq=quantization(res_t,27);
                res_tq1=inv_quantization(res_tq,27);
                code_res=inv_integer_transform(res_tq1);
                code_res=round(code_res/64);

YS(1:4,j:j+3)=b2+code_res;Yb(1:4,j:j+3)=b2;
                fprintf(file, '%d\n',2);
                fprintf(file, '%d\n',int8(res2));
                fprintf(file,
'%d\n',sum(sum(abs(res2))));
      case s3,  res_t=integer_transform(res3);
                res_tq=quantization(res_t,27);
                res_tq1=inv_quantization(res_tq,27);
                code_res=inv_integer_transform(res_tq1);
                code_res=round(code_res/64);
```

```
                YS(1:4,j:j+3)=b3+code_res;
                Yb(1:4,j:j+3)=b3;
                fprintf(file, '%d\n',8);
                fprintf(file, '%d\n',int8(res3));
                fprintf(file,
'%d\n',sum(sum(abs(res3))));
   end
end

%prédication de prémier colone
for i=5:4:x
    v=YS(i-1,1:7);
    b1=mode4x4V(v(1:4));
    b2=mode4x4DC(v(1:4),[]);
    b3=mode74x4(v);
    res1=Y(i:i+3,1:4)-b1;
    res2=Y(i:i+3,1:4)-b2;
    res3=Y(i:i+3,1:4)-b3;
    s1=sum(sum(abs(res1)));
    s2=sum(sum(abs(res2)));
    s3=sum(sum(abs(res3)));
    err=[ s1 s2 s3 ];
    terr=sort(err);
   switch terr(1)
     case s1,  res_t=integer_transform(res1);
               res_tq=quantization(res_t,27);
               res_tq1=inv_quantization(res_tq,27);
               code_res=inv_integer_transform(res_tq1);
               code_res=round(code_res/64);
               YS(i:i+3,1:4)=b1+code_res;
               Yb(i:i+3,1:4)=b1;
               fprintf(file, '%d\n',0);
               fprintf(file, '%d\n',int8(res1));

     case s2,   res_t=integer_transform(res2);
               res_tq=quantization(res_t,27);
               res_tq1=inv_quantization(res_tq,27);
               code_res=inv_integer_transform(res_tq1);
               code_res=round(code_res/64);
```

```
                YS(i:i+3,1:4)=b2+code_res;
                Yb(i:i+3,1:4)=b2;
                fprintf(file, '%d\n',2);
                fprintf(file, '%d\n',int8(res2));

        case s3,  res_t=integer_transform(res3);
                res_tq=quantization(res_t,27);
                res_tq1=inv_quantization(res_tq,27);
                code_res=inv_integer_transform(res_tq1);
                code_res=round(code_res/64);
                YS(i:i+3,1:4)=b3+code_res;
                Yb(i:i+3,1:4)=b3;
                fprintf(file, '%d\n',7);
                fprintf(file, '%d\n',int8(res3));
   end
end

%imshow(YS);

 for j=5:4:y
   for i=5:4:x
    v1=YS(i-1:i+3,j-1);
    if(j+7)<=y
    v2=YS(i-1,j:j+7);
    else v2=YS(i-1,j:j+3);
         v2(5:8)=[v(4),v(4),v(4),v(4)];
    end

    b0=floor(mode4x4V(v2(1:4)));
    b1=floor(mode4x4H(v1(2:5)));
    b2=floor(mode4x4DC(v1(2:5),v2(1:4)));
    b3=floor(mode34x4(v2));
    b4=floor(mode44x4(v2,v1));
    b5=floor(mode54x4(v1,v2));
    b6=floor(mode64x4(v1,v2));
    b7=floor(mode74x4(v2));
    b8=floor(mode84x4(v1));

      res0=Y(i:i+3,j:j+3)-b0;
```

```
        res1=Y(i:i+3,j:j+3)-b1;
        res2=Y(i:i+3,j:j+3)-b2;
        res3=Y(i:i+3,j:j+3)-b3;
        res4=Y(i:i+3,j:j+3)-b4;
        res5=Y(i:i+3,j:j+3)-b5;
        res6=Y(i:i+3,j:j+3)-b6;
        res7=Y(i:i+3,j:j+3)-b7;
        res8=Y(i:i+3,j:j+3)-b8;

        s0=sum(sum(abs(res0)));
        s1=sum(sum(abs(res1)));
        s2=sum(sum(abs(res2)));
        s3=sum(sum(abs(res3)));
        s4=sum(sum(abs(res4)));
        s5=sum(sum(abs(res5)));
        s6=sum(sum(abs(res6)));
        s7=sum(sum(abs(res7)));
        s8=sum(sum(abs(res8)));
        err=[s0 s1 s2 s3 s4 s5 s6 s7 s8];
        terr=sort(err);

          switch terr(1)
               case s0,res_t=integer_transform(res0);
                      res_tq=quantization(res_t,27);
                      res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                       code_res=round(code_res/64);
                   Yb(i:i+3,j:j+3)=b0;
                   YS(i:i+3,j:j+3)=b0+code_res;
                   fprintf(file, '%d\n',0);
                   fprintf(file, '%d\n',int8(res0));
               case s1,res_t=integer_transform(res1);
                      res_tq=quantization(res_t,27);
                      res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                       code_res=round(code_res/64);
                   Yb(i:i+3,j:j+3)=b1;
```

```
                YS(i:i+3,j:j+3)=b1+code_res;;
                fprintf(file, '%d\n',1);
                fprintf(file, '%d\n',int8(res1));
            case s2,res_t=integer_transform(res2);
                   res_tq=quantization(res_t,27);
                   res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                    code_res=round(code_res/64);
                Yb(i:i+3,j:j+3)=b2;
                YS(i:i+3,j:j+3)=b2+code_res;
                fprintf(file, '%d\n',2);
                fprintf(file, '%d\n',int8(res2));
            case s3,res_t=integer_transform(res3);
                   res_tq=quantization(res_t,27);
                   res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                    code_res=round(code_res/64);
                Yb(i:i+3,j:j+3)=b3;
                YS(i:i+3,j:j+3)=b3+code_res;
                fprintf(file, '%d\n',3);
                fprintf(file, '%d\n',int8(res3));
            case s4,res_t=integer_transform(res4);
                   res_tq=quantization(res_t,27);
                   res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                    code_res=round(code_res/64);
                Yb(i:i+3,j:j+3)=b4;
                YS(i:i+3,j:j+3)=b4+code_res;
                fprintf(file, '%d\n',4);
                fprintf(file, '%d\n',int8(res4));
            case s5,res_t=integer_transform(res5);
                   res_tq=quantization(res_t,27);
                   res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                    code_res=round(code_res/64);
```

```
                    Yb(i:i+3,j:j+3)=b5;
                    YS(i:i+3,j:j+3)=b5+code_res;
                    fprintf(file, '%d\n',5);
                    fprintf(file, '%d\n',int8(res5));
                case s6,res_t=integer_transform(res6);
                       res_tq=quantization(res_t,27);
                       res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                        code_res=round(code_res/64);
                    Yb(i:i+3,j:j+3)=b6;
                    YS(i:i+3,j:j+3)=b6+code_res;
                    fprintf(file, '%d\n',6);
                    fprintf(file, '%d\n',int8(res6));
                case s7,res_t=integer_transform(res7);
                       res_tq=quantization(res_t,27);
                       res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                        code_res=round(code_res/64);
                    Yb(i:i+3,j:j+3)=b7;
                    YS(i:i+3,j:j+3)=b7+code_res;
                    fprintf(file, '%d\n',7);
                    fprintf(file, '%d\n',int8(res7));
                case s8,res_t=integer_transform(res8);
                       res_tq=quantization(res_t,27);
                       res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                        code_res=round(code_res/64);
                    Yb(i:i+3,j:j+3)=b8;
                    YS(i:i+3,j:j+3)=b8+code_res;
                    fprintf(file, '%d\n',8);
                    fprintf(file, '%d\n',int8(res8));
            end
             YS(i:i+3,j:j+3)


        end
```

```
 end
fclose(file);
YS=uint8(YS);Yb=uint8(Yb);
figure(2)
imshow(YS);
figure(3)
imshow(Yb);
YR=uint8(Y)-uint8(Yb);
figure(4)
imshow(YR);
Y=double(Y);
YS=double(YS);

p=255;
MSE=0;
MSE=sum(sum((Y-YS).^2))/(x*y)
PNSR=10*log10(p*p/MSE)
```

Intra 16x16

```
%programme intra 16x16
clear all
close all
clc
% lecture d'image
I= imread('images.jpeg');%
% Extraction des bandes d'image
R = I(:,:,1);
G = I(:,:,2);
B = I(:,:,3);
% les dimensions d'image
[x y]=size(R);
%transformation vers l'espace YUV
for i=1:x
    for j=1:y
        Y(i,j)   =   0.299*R(i,j)   +   0.587*G(i,j)   +
0.114*B(i,j)    ;
        U(i,j) = 0.564*(B(i,j) - Y(i,j) );
```

```
        V(i,j) = 0.713*(R(i,j) - Y(i,j) );
    end
end
%affichage des composants de YUV
figure(1)
imshow(Y)
%Y=Y1(1:40,1:40);
%[x y]=size(Y)
%figure
%imshow(U)
%figure
%imshow(V)
% sous echantillonnage de U V
c1=1;
for i=1:2:x
    c2=1;
    for j=1:2:y
       %
Ue(c1,c2)=round(U(i,j)+U(i,j+1)+U(i+1,j)+U(i+1,j+1)/4);
       Ue(c1,c2)=U(i,j);

%Ve(c1,c2)=round(V(i,j)+V(i,j+1)+V(i+1,j)+V(i+1,j+1)/4);
        Ve(c1,c2)=V(i,j);
         c2=c2+1;
    end
    c1=c1+1;

end
%------------l'étape de prédiction----------------------
file=fopen('file.txt','w');
[height,width]=size(Y);
fprintf(file,'%d\n',width/16);
fprintf(file,'%d\n',height/16);

Y=double(Y);
%prédication du premier bloc
b=mode16x16DC([],[]);
res=Y(1:16,1:16)-b;
for i=1:4:16
```

```
    for j=1:4:16
        res2 = res(i:i+3,j:j+3);
        res_t=integer_transform(res2);
        res_tq=quantization(res_t,27);
        res_tq1=inv_quantization(res_tq,27);
        code_res=inv_integer_transform(res_tq1);
        code_res=round(code_res/64);
        out(i:i+3,j:j+3)=code_res;
    end
end
YS(1:16,1:16)= b + out;
Yb(1:16,1:16)= b;
fprintf(file, '%d\n',2);
fprintf(file, '%d\n',int8(res));
fprintf(file, '%d\n',sum(sum(abs(res))));
%prédication de la première ligne
for j=17:16:y
    V=YS(1:16,j-1);
    b1=mode16x16H(V);
    b2=mode16x16DC([],V);
    res1=Y(1:16,j:j+15)-b1;
    res2=Y(1:16,j:j+15)-b2;
    s1=sum(sum(abs(res1)));
    s2=sum(sum(abs(res2)));
    if (s1 > s2)
        for k=1:4:16
            for m=1:4:16
                res4 = res2(k:k+3,m:m+3);
                res_t=integer_transform(res4);
                res_tq=quantization(res_t,27);
                res_tq1=inv_quantization(res_tq,27);
                code_res=inv_integer_transform(res_tq1);
                code_res=round(code_res/64);
                out(k:k+3,m:m+3)=code_res;
            end
        end
        Yb(1:16,j:j+15)=b2;
        YS(1:16,j:j+15)=b2+out;
        fprintf(file, '%d\n',2);
```

```
        fprintf(file, '%d\n',int8(res2));
    else
        for k=1:4:16
            for m=1:4:16
                res5 = res1(k:k+3,m:m+3);
                res_t=integer_transform(res5);
                res_tq=quantization(res_t,27);
                res_tq1=inv_quantization(res_tq,27);
                code_res=inv_integer_transform(res_tq1);
                code_res=round(code_res/64);
                out(k:k+3,m:m+3)=code_res;
            end
        end
        Yb(1:16,j:j+15)=b1;
        YS(1:16,j:j+15)=b1+out;
        fprintf(file, '%d\n',1);
        fprintf(file, '%d\n',int8(res1));
    end
end
%prédication de prémier colone
for i=17:16:x
    v=YS(i-1,1:16);
    b1=mode16x16V(v);
    b2=mode16x16DC(v,[]);
    res1=Y(i:i+15,1:16)-b1;
    res2=Y(i:i+15,1:16)-b2;
    s1=sum(sum(abs(res1)));
    s2=sum(sum(abs(res2)));
    if (s1 > s2)
        for k=1:4:16
            for m=1:4:16
                res4 = res2(k:k+3,m:m+3);
                res_t=integer_transform(res4);
                res_tq=quantization(res_t,27);
                res_tq1=inv_quantization(res_tq,27);
                code_res=inv_integer_transform(res_tq1);
                code_res=round(code_res/64);
                out(k:k+3,m:m+3)=code_res;
            end
```

```
            end
            Yb(i:i+15,1:16)=b2;
            YS(i:i+15,1:16)=b2+out;
            fprintf(file, '%d\n',2);
            fprintf(file, '%d\n',int8(res2));
        else
            for k=1:4:16
                for m=1:4:16
                    res5 = res1(k:k+3,m:m+3);
                    res_t=integer_transform(res5);
                    res_tq=quantization(res_t,27);
                    res_tq1=inv_quantization(res_tq,27);
                    code_res=inv_integer_transform(res_tq1);
                    code_res=round(code_res/64);
                    out(k:k+3,m:m+3)=code_res;
                end
            end
            Yb(i:i+15,1:16)=b1;
            YS(i:i+15,1:16)=b1+out;
            fprintf(file, '%d\n',0);
            fprintf(file, '%d\n',int8(res1));
        end
    end
    %les blocs restants
     for j=17:16:y
        for i=17:16:x
            v1=YS(i-1:i+15,j-1);
            if (j+15) <= y
                v2=YS(i-1,j:j+15);
            else
                v2=YS(i-1,j:j+15);
                v2(17:32)=[v(16),v(16),v(16),v(16)];
            end
            b0 = floor(mode16x16V(v2(1:16)));
            b1 = floor(mode16x16H(v1(2:17)));
            b2 = floor(mode16x16DC(v1(2:17),v2(1:16)));
            b3 = floor(modeplane16x16(v2,v1));
            res0 = Y(i:i+15,j:j+15)-b0;
            res1 = Y(i:i+15,j:j+15)-b1;
```

```
        res2 = Y(i:i+15,j:j+15)-b2;
        res3 = Y(i:i+15,j:j+15)-b3;
        s0=sum(sum(abs(res0)));
        s1=sum(sum(abs(res1)));
        s2=sum(sum(abs(res2)));
        s3=sum(sum(abs(res3)));
        err=[s0 s1 s2 s3 ];
        terr=sort(err);
        switch terr(1)
             case s0,
                    for k=1:4:16
                        for m=1:4:16
                            res4 = res0(k:k+3,m:m+3);

res_t=integer_transform(res4);

res_tq=quantization(res_t,27);

res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                            code_res=round(code_res/64);
                            out(k:k+3,m:m+3)=code_res;
                        end
                    end
                    Yb(i:i+15,j:j+15) = b0;
                    YS(i:i+15,j:j+15) = b0 + out;
                    fprintf(file, '%d\n',0);
                    fprintf(file, '%d\n',int8(res0));
             case s1
                    for k=1:4:16
                        for m=1:4:16
                            res4 = res1(k:k+3,m:m+3);

res_t=integer_transform(res4);

res_tq=quantization(res_t,27);

res_tq1=inv_quantization(res_tq,27);
```

```
code_res=inv_integer_transform(res_tq1);
                            code_res=round(code_res/64);
                            out(k:k+3,m:m+3)=code_res;
                        end
                    end
                    Yb(i:i+15,j:j+15) = b1;
                    YS(i:i+15,j:j+15) = b1 + out;
                    fprintf(file, '%d\n',1);
                    fprintf(file, '%d\n',int8(res1));
             case s2
                    for k=1:4:16
                        for m=1:4:16
                            res4 = res2(k:k+3,m:m+3);

res_t=integer_transform(res4);

res_tq=quantization(res_t,27);

res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                            code_res=round(code_res/64);
                            out(k:k+3,m:m+3)=code_res;
                        end
                    end
                    Yb(i:i+15,j:j+15) = b2;
                    YS(i:i+15,j:j+15) = b2 + out;
                    fprintf(file, '%d\n',2);
                    fprintf(file, '%d\n',int8(res2));
            case s3
                    for k=1:4:16
                        for m=1:4:16
                            res4 = res3(k:k+3,m:m+3);

res_t=integer_transform(res4);

res_tq=quantization(res_t,27);
```

```
res_tq1=inv_quantization(res_tq,27);

code_res=inv_integer_transform(res_tq1);
                                code_res=round(code_res/64);
                                out(k:k+3,m:m+3)=code_res;
                            end
                        end
                        Yb(i:i+15,j:j+15) = b3;
                        YS(i:i+15,j:j+15) = b3 + out;
                        fprintf(file, '%d\n',3);
                        fprintf(file, '%d\n',int8(res3));
            end
        end
    end
fclose(file);
YS = uint8(YS);
figure(2)
imshow(YS);
YR = uint8(Y) - uint8(Yb);
figure(3)
imshow(YR);
Yb = uint8(Yb);
figure(4)
imshow(Yb);

YS=double(YS);
p=255;
MSE=0;
MSE=sum(sum((Y-YS).^2))/(x*y)
PNSR=10*log10(p*p/MSE)
```

2. Images de test:



01



02



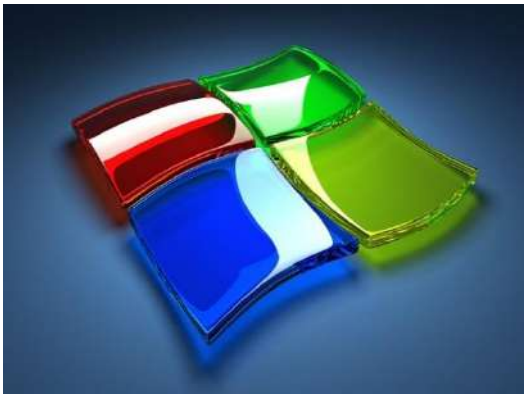
03



04



05



06