الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Études

présenté par

ALLOUTI Djahida

&

TOUTAOUI Lina

pour l'obtention du diplôme master 2 option traitement de l'information

et systèmes électroniques

Thème

Codage hybride pour la compression d'images

Proposé par : Mlle BENBLIDIA Nadjia & Mlle REGUIEG F. Zohra

Année Universitaire 2011-2012

Nous tenons à remercier dieu le tout puissant qui nous a donné le courage et la

Patience qui a éclairé non chemins pour achever ce travail.

Nous tenons à remercier profondément notre promotrice Mme N.Benblidia et notre co-promotrice Mme F.Zohra Reguieg, pour leurs patiences, leurs conseils et leurs encouragements.

Nous témoignons également notre gratitude à l'ensemble des membres de juré.

Nous adressons aussi nos sincères remerciements à nos parents à qui nous devons tout et que dieu les gardent pour nous.

Résumé : Comme l'utilisation de l'image numérique est à la hauteur. La compression de données d'images numériques acquises est devient de plus en plus important pour faire face aux besoins de stockage. L'un des défis de la compression est de compresser les images à haute efficacité tout en préservant la qualité de l'image reconstruite.
Dans cette recherche, nous présentons un algorithme hybride proposé pour la compression d'images couleurs basé sur deux transformée discrète à deux dimension, la transformée en ondelette DWT et la transformée en cosinus DCT, L'algorithme proposé commence par l'utilisation en une seule étape la transformée en ondelette discrète pour décomposer une image en quatre sous-bandes; basse fréquence et hautes fréquences, chaque bloc de "n × n" de la sous-bande basse fréquence (LF) sont transformés à l'aide DCT bidimensionnelle, en suite en va faire la quantification et le codage arithmétique pour obtenir l'image compressée Notre algorithme de compression s'est révélé bon taux de compression, et comparées en fonction du PSNR.

Mots clés : compression d'image, ondelette, taux de compression, PSNR

Abstract: As the use of digital images is up Data compression of digital images acquired is becoming increasingly important to meet storage needs. One of the challenges of compression is to compress the images at high efficiency while maintaining the quality of the reconstructed image. In this research, we present a hybrid algorithm proposed for compression of color images based on two two-dimensional discrete transform, the wavelet transform DWT and the cosine transform DCT, the proposed algorithm starts by using a single step the discrete wavelet transform for decomposing an image into four sub-bands; low frequency and high frequencies, each block of "n × n" of the sub-band low frequency (LF) are transformed two-dimensional DCT using, in following applies to the quantification and arithmetic coding for the compressed image Our compression algorithm proved good compression ratio, and compared as a function of PSNR**.**

Keywords : Image compression, wavelete, compression ratio,PSNR

Table des matières

Chapitre 3: Description du logiciel élaboré

Liste des figures

Liste des tableaux

Introduction générale

Avec le développement de l'informatique, des appareils très performants sont apparus sur le marché et de puissantes méthodes ont été mises en œuvre pour faire face aux besoins d'acquisition, de stockage et de traitement de l'information. Un tel développement a engendré des applications qui mettent en jeu la transmission et l'archivage de données de plus en plus volumineuses. Il est alors devenu primordial de compresser ces données pour les ramener à des tailles plus facilement exploitables.

De façon générale, les fichiers numériques à traiter se distinguent par leur format qui est identifiable grâce à l'extension du nom du fichier. Dans le cas d'images brutes, le fichier bitmap reste le plus utilisé. En fait, près de soixante-dix formats différents sont actuellement disponibles pour stocker des images bitmap, tout en étant conformes aux standards les plus usités tels que Windows ou Unix. Ceci implique la compatibilité des programmes avec de telles normes, lors de leur conception. Si l'on veut augmenter la vitesse de consultation des fichiers ou minimiser l'espace de stockage des documents numérisés, surtout dans le cas des images, il est nécessaire de recourir à des systèmes de compression appropriés. Deux sortes de techniques permettent la compression des images : les méthodes réversibles, c'est a dire sans pertes, qui conduisent à de faibles taux de compression et celles appelées irréversibles qui permettent de compresser fortement les images mais au prix de certaines distorsions.

Parmi les méthodes dites sans pertes, citons la technique RLC (*Run Length Coding*), le codage de Huffman et la compression LZW (*Lempel Ziv Welch*). Ces méthodes opèrent avec des facteurs de compression allant de 1,2 à 2,5. Quant aux méthodes à fort taux de compression, les plus efficaces sont basées sur de puissants outils tels que

la transformée discrète en cosinus (DCT) et la transformée en Ondelettes, Avec de tels outils, le fichier image subit une forte

Compression mais avec des pertes se traduisant par des dégradations plus ou moins perceptibles à l'œil nu lors de la décompression. Parmi les méthodes à fort taux de compression,

Parmi les méthodes à fort taux de compression, on distingue le format JPEG (*Joint Photographic Expert Group*) et la méthode GIF (*Graphic Interchange Format*). JPEG reste néanmoins le plus couramment employé actuellement. Son principe est de diviser l'image en blocs carrés de 8 x 8 pixels et de coder les valeurs les plus proches dans chaque bloc sur quelques bits. En procédant de la sorte, nous pouvons atteindre des taux de compression supérieurs à 90 % tout en restituant des images avec une perte d'informations à peine perceptible. Mais, à l'instar d'autres méthodes irréversibles, le temps d'exécution est très lent surtout dans le cas d'images de grande taille. De plus, des artefacts apparaissent dans celles-ci sous forme de mosaïques. Pour réduire les distorsions causées par les méthodes à fort taux de compression, nous avons élaboré dans notre étude une technique hybride basé sur la transformée en ondelette discrète DWT et la transformée en cosinus discrète DCT, Cette technique appliquée à des images de référence comme l'image Lena et à d'autres images couleurs En effet, elle a permis de réaliser un bon compromis entre le taux de compression et la qualité de restitution des images. Grâce à cette méthode, le taux de compression et le PSNR atteignent respectivement 98.44% et 36.10dB

Notons que la méthode élaborée, a permis après décompression, de reproduire fidèlement les images considérées et de réduire l'espace mémoire requis pour leur stockage de plus de 97 %.

Le travail que nous présentons dans cette thèse a été organisé en trois chapitres :

- Dans le premier chapitre, nous donnons des notions essentielles sur la compression, la Structure générale des algorithmes utilisés ainsi que les paramètres permettant d'évaluer leurs performances.

- Dans le deuxième chapitre, nous présentons une méthode de compression d'images Hybride basé sur deux méthode de transformées, la transformée en ondelette discrète DWT et la transformée en cosinus discrète DCT.
- Le troisième chapitre comporte les résultats obtenus présentés et analysés sur plusieurs images ainsi l'application réalisée pour la simulation des résultats.
- Finalement, nous présenterons les conclusions de la thèse et Les perspectives du travail.

Techniques de compression d'images

Préambule

La compression de données est une activité ancienne, l'utilisation d'abréviations est une preuve. Les langues elles-mêmes utilisent des mots de longueurs variées, les plus fréquents étant les plus courts, afin de réduire la taille des phrases. La compression des images numériques, C'est elle qui a permis la diffusion des images sur Internet ou encore la propagation des appareils photos numériques. Elle constitue également la base de la compression vidéo.

L'image devient vite lourde de données, donc excessivement longue à transmettre et à traiter. Voilà pourquoi depuis quelques années, les centres de recherche en informatique dépensent de nombreuses heures sur des algorithmes de compression. Afin de limiter la taille, ou le poids d'une image, nous devons la compresser, c'est-à-dire éliminer les informations inintéressantes ou redondantes. Il existe de nos jours plus d'une vingtaine de formats de compression, spécifiquement dans la compression d'image (.gif, .jpeg, .bmp…).

Dans ce chapitre, nous donnerons quelques notions essentielles sur la compression des images, la structure générale des algorithmes utilisés ainsi que les paramètres permettant d'évaluer leurs performances.

Généralités

Notion de pixel

En informatique, et ce pour des raisons de gain de place, une image est composée d'un ensemble de points, appelés pixel (abréviation de Picture Elément). Ces pixels sont regroupés en lignes et en colonnes afin de former un espace à deux dimensions. Chaque point sera représenté par ses coordonnées (X, Y), avec X l'axe orienté de gauche à droite, et Y l'axe orienté de haut en bas.

Représentation de la couleur

En plus de ses coordonnées planaires, un pixel se caractérise par sa pondération, appelée aussi profondeur de codage, qui représente sa couleur ou son intensité. Cette valeur peut être codée sur un nombre n différent de bits (ou octet) selon les méthodes de codage de la couleur utilisées. Les standards les plus répandus sont n=1 bit (noir ou blanc), n=4 (16 couleurs ou 16 niveaux de gris), n=8 bits (256 couleurs ou 256 niveaux de gris), On appelle la palette de couleur, l'ensemble des couleurs que peut contenir une image. Il est fréquent de voir des images qui n'utilisent jamais certaines couleurs, il devient dès lors intéressant de limiter la palette de couleur en ne sélectionnant que la ou les couleurs utilisées réellement par l'image.il existe Plusieurs types d'images les plus connue sont :

Les images binaires (noir ou blanc)

Images la plus simples, un pixel peut prendre uniquement les valeurs noir ou blanc. C'est typiquement le type d'image que l'on utilise pour scanner du texte quand celui-ci est composé d'une seule couleur.

Les images en niveaux de gris

En général, les images en niveaux de gris renferment 256 teintes de gris. Par convention la valeur zéro représente le noir (intensité lumineuse nulle) et la valeur 255 le blanc (intensité lumineuse maximale). Le nombre 256 est lié à la quantification de l'image. En effet chaque entier représentant un niveau de gris

est codé sur 8 bits. Il est donc compris entre 0 et 28 – 1 = 255. C'est la quantification la plus courante. On peut coder une image en niveaux de gris sur 16 bits (0 ≤ n ≤ 216 - 1) ou sur 2 bits : dans ce dernier cas le « niveau de gris » vaut 0 ou 1 : il s'agit alors d'une image binaire (Noir et Blanc).

Les images couleurs

L'espace couleur est basé sur la synthèse additive des couleurs, c'est à dire que le mélange de trois composantes (par exemple (R, V, B)) donne une couleur. Un pixel est codé par trois valeurs numériques. La signification de ces valeurs dépend du type de codage choisi. Le plus utilisé pour la manipulation des images numériques est l'espace couleur Rouge, Vert, Bleu (R, V, B) (RGB en anglais). La restitution des couleurs sur écran utilise cette représentation (synthèse additive).

Taille et définition d'une image

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

Prenons l'exemple d'une image 1024 x 768, dont la couleur est codée sur 24 bits (1octet pour les nuances de rouge, 1 pour le bleu et 1 octet pour le vert, codage True color ou RGB

- *Nombre de pixels :* 1024 x 768 = 786432 pixels

- *Poids de l'image :* 786432 x 3 = 2359296 octets

- soit une image de 2359296 / 1024 = 2304 Ko, ou 2304 / 1024 = 2,25 Mo, ce qui est assez conséquent, surtout lorsqu'on veut transmettre l'image…

Compression de données

L'enjeu de la recherche sur la compression d'image est de trouver un moyen de diminuer la taille d'une image, tout en essayant de limiter la dégradation due à la compression.

Nécessite de la compression de données

Aujourd'hui l'imagerie numérique (Internet, CD vidéo, télévision numérique) remplacent de plus en plus les techniques analogiques (télévision classique, vidéo VHS). Les images numériques partagent les propriétés des signaux numériques. C'est-à-dire que leur support est discret (deux coordonnées spatiales plus une coordonnée temporelle pour les séquences d'images), de même que l'ensemble des valeurs possibles. De plus, ces deux ensembles sont finis en pratique. Les images numériques sont en fait des matrices de points, ces points étant appelés pixels de l'image. Un pixel est décrit par un ou plusieurs nombres, indiquant les propriétés visuelles du point (intensité, couleur). On parle des images binaires si un seul bit décrit chaque point, 0 représentant un point noir, 1 un point blanc. En attribuant plusieurs bits aux pixels, un nombre plus élevé de niveaux de gris peut être distingué. La plupart du temps, les images monochromes sont codées sur huit bits, puisque cela devient exactement un octet et les 256 différentes intensités ainsi représentables sont largement suffisantes pour la perception humaine. Cependant, cela peut aller jusqu'à 16 bits lorsque les appareils de mesure ont une meilleur résolution dans les applications exigeantes, comme par exemple l'imagerie médicale. Les images en couleurs utilisent plusieurs valeurs pour décrire chaque pixel. Une représentation très répandue est celle de RGB où, en général, un octet est réservé pour indiquer la proportion de chacune des composantes rouge, vert et bleu.
Une image contenant des milliers de points, sa gestion consiste à traiter – pour stocker et/ou transmettre – cette quantité importante de données. Une seule image noir et blanc de télévision (de taille standard 720×575) représente 400 Ko sous forme brute, non comprimée, cela devenant trois fois plus important dans le cas d'une image en couleur.

L'intérêt de la compression est donc évident. Les images comprimées occupent moins de place sur une unité de stockage. Elles prennent moins de temps de transmission sous forme comprimée sur le même canal, ou bien, elles ont besoin d'une bande passante plus petite pour arriver à destination en même temps que la même image non comprimée.

Le but de la compression d'image est donc de modifier la représentation initiale des données, pour qu'elles occupent moins de place. Cette nouvelle représentation sera décodée durant une procédure de décompression pour reconstruire l'image.

Classification des techniques de compression de données

Les techniques de compression de données sont principalement classées en deux groupes comme suit :

- Technique de compression sans perte.
- Technique de compression avec perte.

Les Technique de compression sans pe*rte*

Les techniques de compression sans pertes présentent l'avantage d'une restitution exacte des pixels de l'image originale il n'y a donc aucune perte, et l'inconvénient d'un faible taux de compression, d'information. Cette famille d'algorithme est essentielle dans nos ordinateurs, la totalité des programmes d'archivage sont bâtis sur cette technique de compression sans perte d'information. Le principe courant de ces programmes est de réduire les séquences d'information redondantes [1].

En pratique, les méthodes sans pertes basées sur les codeurs entropiques restent les plus utilisées. Ces codeurs ont été mis en œuvre pour coder les pixels de l'image avec la contrainte d'obtenir des mots de code de longueur aussi proche que possible de l'entropie de l'image [7]. Les codeurs les plus utilisés en compression d'images sont le codeur prédictif, les codeurs à longueurs variables, ceux à base de dictionnaires et les codeurs par longueur de plages.

- ***Codeur prédictif linéaire*** Les algorithmes qui utilisent le codage par prédiction exploitent la redondance spatiale. Il s'agit de prédire la valeur d'un pixel en fonction de la valeur des pixels voisins et de ne coder que l'erreur de prédiction. Le voisinage peut être défini selon sa connexité (4-connexité ou 8-connexité) ou selon l'ordre du parcours choisi pour accéder aux pixels voisins. La technique de prédiction la plus utilisée est la DPCM (Differential Pulse Code Modulation) [6].

Cette technique effectue une prédiction à base d'une combinaison linéaire des valeurs des pixels voisins

- ***Codeurs à longueur variable*** Le principe de ces codeurs est d'allouer les mots binaires les plus courts possibles aux caractères les plus courants, et réserver les codes les plus longs à ceux les moins fréquents. Le but est d'avoir un nombre de bits après codage qui soit le plus faible possible. Différents algorithmes permettent de déterminer des codes à longueurs variables. Le codage de Huffman et celui de Shannon Fano restent les plus utilisés
- **Codage de Huffman** Le codage de Huffman est certainement le plus utilisé en raison de sa gratuité, de sa simplicité et de son efficacité. Il est notamment utilisé pour le codage d'images binaires, telles que les fax, ainsi que dans toutes les modalités du standard JPEG. Le principe de ce codage est de créer des codes de longueurs variables [614]. Pour ce faire, les pixels sont d'abord triés et classés en fonction de leur fréquence (occurrence). Un graphe est alors construit de la manière suivante :

  A partir des deux pixels présentant la fréquence la plus faible, un nœud est créé. Il lui est affecté un poids égal à la somme des fréquences des deux symboles. Le nœud créé remplace désormais les deux symboles dans la suite du processus. A ces derniers sont affectés respectivement les chiffres binaires 0 pour le plus fréquent et 1 pour le plus rare ou l'inverse.

  La même démarche est reprise en considérant les deux symboles ou nœuds de poids le plus faible.

  Cette procédure est renouvelée tant qu'il reste plus d'un nœud libre. La figure 1.1 donne un exemple de codage d'un message de 36 caractères par l'algorithme de Huffman.

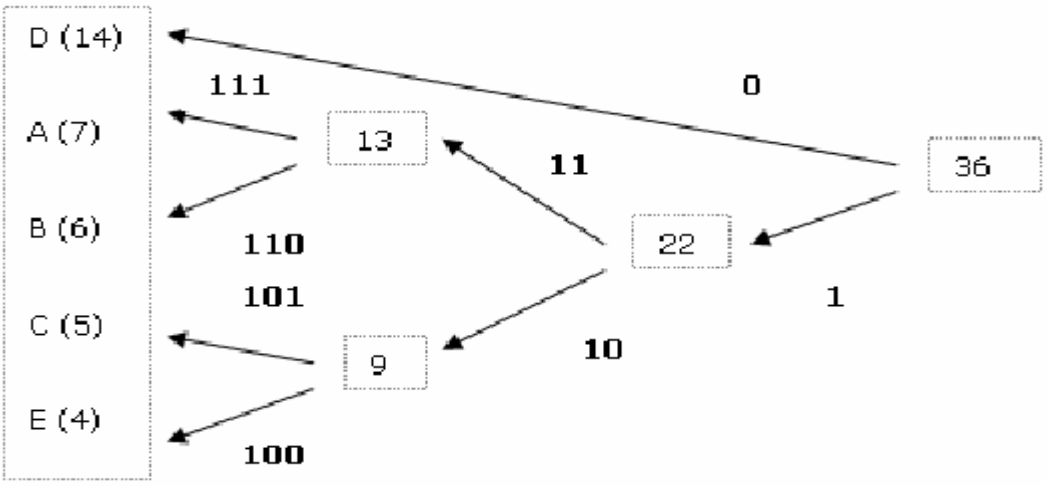

Figure 1. 1 Exemple d'application du codage de Huffman

- **Codage de Shannon Fano** Le codage de Shannon Fano repose sur le même principe que celui d'Huffman. Cet algorithme part des racines et aboutit aux feuilles par divisions successives [6,7]. Pour ce faire, on classe dans un premier temps les probabilités d'apparition des pixels de l'image par ordre décroissant. L'ensemble de ces probabilités est ensuite divisé en deux sous-ensembles de probabilités aussi proches que possibles. A chacun de ces deux ensembles est affecté le code 0 ou 1. Puis, le processus est réitéré jusqu'à ce que chaque ensemble ne soit composé que d'un seul élément. En prenant le même exemple que celui de Huffman, on obtient le graphe de la figure 1.2.

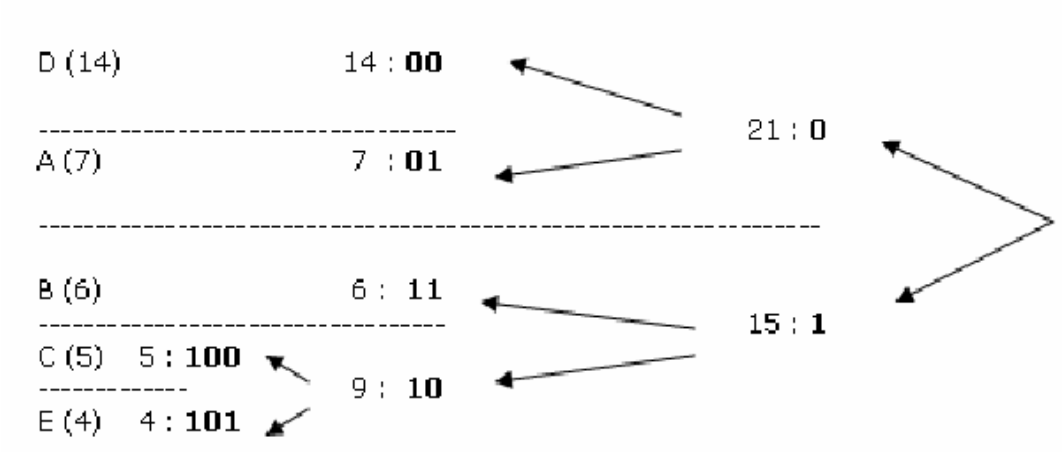

Figure 1. 2 Exemple d'application du codage de Shannon Fano

- ***Codage Arithmétique*** Dans les schémas de compression, les codeurs arithmétiques remplacent de plus en plus les codeurs usuels. Ils sont notamment utilisés dans le standard JPEG2000. Le codage arithmétique permet, à partir de la probabilité d'apparition des symboles d'une source, de créer un seul mot 'code' qui soit associé à une séquence de longueur arbitraire de symboles. Ceci diffère du code de Huffman qui attribue des mots 'codes' de longueurs variables à chaque symbole de la source. Le code associé à une séquence est un nombre réel de l'intervalle [0, 1[. Ce code est construit par subdivisions récursives d'intervalles. Un intervalle est subdivisé pour chaque nouveau symbole qui appartient à la séquence. On obtient, en définitive, un sous intervalle de l'intervalle [0, 1[, tel que tout nombre réel appartenant à cet intervalle représente la séquence à coder [10,11].
- **Codeurs par longueur de plages (Le *format RLE)*** Le principe du codage RLE (Run Length Encoding) est de regrouper dans une image les pixels voisins ayant le même niveau de gris. Chaque groupement définit de façon unique un couple de valeurs Pi(p, n), où p est le nombre de pixels voisins ayant un même niveau de gris n, et i le numéro de code dans la séquence .Le principe de compression de RLE est assez simple à mettre en œuvre. Il repose sur le fait que dans une image, il existe de nombreuses répétitions d'un même pixel, ou d'une même séquence de pixel, tous juxtaposés. Ainsi, au lieu de coder chaque pixel d'une image, le RLE propose de coder d'une part le nombre de répétitions, et d'autre part la séquence ou l'octet à répéter. Chaque groupement définit de façon unique un couple de valeurs Pi(p, n), où p est le nombre de pixels voisins ayant un même niveau de gris n, et i le numéro de code dans la séquence [5,11,12] .
  L'exemple ci-dessous illustre un exemple de codage par RLE pour une séquence de niveaux de gris représentant une ligne d'une image.

….	19	13	13	13	13	109	131	131	162	162	162	162	162	88	…..

...; Pi = (4, 13); Pi+1 = (1, 109); Pi+2 = (2, 131); Pi+3 = (5, 162); ... Ce système de compression est assez bon dans certain cas (images monochromes ou 256 couleurs), mais il s'avère très coûteux pour des images dont la couleur est codée sur 16 ou 24 bits… Il est à noter qu'il existe plusieurs variantes de ce codage,

certains types encodent l'image en décelant des séquences redondantes selon les lignes, les colonnes, ou Même en zigzag [1].

- ***Codeurs à base de dictionnaires*** Le principal codeur à base de dictionnaire est le codeur « Lempel Ziv » (LZ). Il permet de coder des chaînes de longueurs variables par des mots de longueurs fixes. Ainsi, dans le cas d'une image, une suite de pixels est codée à l'aide d'un dictionnaire (ou d'une table de codage) construit au fur et à mesure de la lecture des données. Ce codeur est optimal pour des images de faible entropie telles que les images de synthèse, mais reste inefficace dans le cas des images réelles [13].

  Son principal inconvénient réside dans le temps de calcul requis pour la comparaison des données à coder avec les mots du dictionnaire. Notons que LZ est à la base de nombreux algorithmes de compression de données tels que LZSS, LZ78 et LZW.

Technique de compression avec perte

Afin d'atteindre de forts taux de compression, on utilise les méthodes avec pertes, ils délivrent après compression une image différente, elle contient beaucoup moins d'information que l'original, certains détails auront été éliminés lors du codage. Cette modification est plus ou moins visible, selon le degré de compression. L'attrait de cette famille est qu'elle peut obtenir un rendement formidablement grand. Cependant, on ne peut utiliser ce genre de compression que pour des sons, vidéos et images, mais pas sur des fichiers ou sur du texte puisque ceux-ci ne doivent subir aucune dégradation [1].

Quelques exemples de techniques de compression avec perte sont les suivantes

- JPEG
- JPEG 2000
- ***la compression JPEG*** Le Groupe Joint Photographic Expert (JPEG) norme a été élaborée en 1992, basée sur la transformée en cosinus discrète (DCT). Il a été l'un des plus populaire et largement utilisé dans les méthodes de compression [3, 4]. Bien que la mise en œuvre du matériel pour le JPEG à

l'aide de la DCT est simple, les notables "d'artefacts de blocs" à travers les limites des blocs ne peut être négligé pour taux de compression élevé. En outre, la qualité des images reconstruites est dégradée par les "Faux contours" effet pour les images comportant des zones spécifiques progressivement ombragées [5]. Les faux contours se produit lorsque la zone de douceur classés d'une image est déformée par une aberration due au quantification lourde des coefficients de transformation. L'effet ressemble à une carte de contours. Le processus de compression et de décompression JPEG irréversibles comporte six étapes principales représentées ci-dessous :

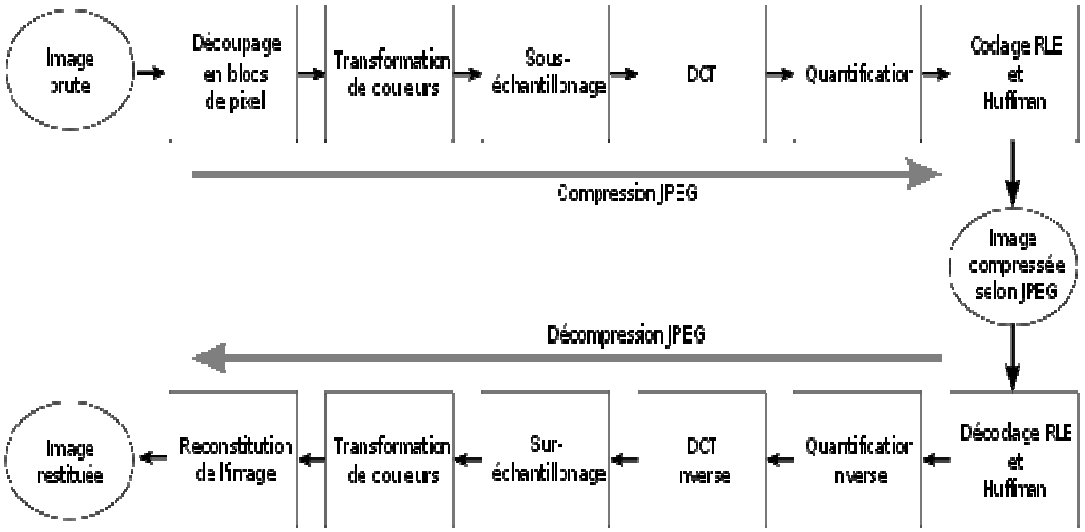

Figure 1. 3 Organigramme de compression/décompression JPEG.

JPEG est capable de coder les couleurs sous n'importe quel format, toutefois les meilleurs taux de compression sont obtenus avec des codages de couleur de type luminance/chrominance car l'œil humain est assez sensible à la luminance (la luminosité) mais peu à la chrominance (la teinte) d'une image. Afin de pouvoir exploiter cette propriété, - l'algorithme convertit l'image d'origine depuis son modèle colorimétrique initial (en général RVB) vers le modèle de type chrominance/luminance YCbCr. Dans ce modèle, Y est l'information de luminance, et Cb et Cr sont deux informations de chrominance, respectivement le bleu moins Y et le rouge moins Y.

L'algorithme de compression découpe premièrement les matrices en blocs de 8x8 pixels, et leur applique ensuite la fonction DCT (Discrete Cosinus Transform). Cette fonction DCT est une transformation en série (Fourier), qui délivre une représentation non plus spatiale, mais dans le domaine fréquentiel. En effet, la ligne et la colonne de la matrice représentent les axes X et Y dans le domaine spatial, et la valeur d'une case particulière représente la valeur de Z. Nous raisonnons donc en 3 dimensions. La transformation de la matrice donne une nouvelle matrice contenant cette fois ci, les différentes puissances spectrales pour chaque fréquence. L'élément (0,0) représente la valeur moyenne du signal [3,4].

La quantification réside dans le faite que l'algorithme attribue à chaque fréquence, à chaque cellule de la matrice 8x8 pixels, un coefficient de perte. L'algorithme annulera ou diminuera les hautes fréquences qui, représentent les plus petits détails de l'image.

L'atténuation de l'amplitude des différentes fréquences est déterminée par le ratio demandé par l'utilisateur.

Enfin, la matrice obtenue est linéarités en zigzag, selon le codage RLE, et est compressée avec la méthode de Huffman.

Les étapes de la décompression s'effectuent dans l'ordre inverse de la compression suivant les méthodes définies précédemment (en même temps que la compression).

- **la compression JPEG *2002*** JPEG est le sigle de Joint Photographic Experts Group qui est le nom du groupe de travail qui a créé cette norme reconnue par l'ISO. La norme JPEG 2000 définit le codage des images numériques et est destinée à supplanter la norme JPEG. L'extension des images au format JPEG 2000 est *.jp2* [2]. L'originalité du JPEG 2000 est qu'il permet de compresser les images **avec ou sans perte(s)** d'informations. La norme ne décrit cependant que la méthode de décompression qui doit être utilisée pour les images de ce format. Dès lors, les développeurs qui désirent implémenter un algorithme de compression conforme à JPEG 2000 procèdent à leur guise tant que leur méthode répond aux exigences de la norme.

La compression JPEG 2000 peut débuter par la transformation des trois composants colorimétriques de l'image en un coefficient de luminance et deux coefficients pour exprimer la couleur. Contrairement au JPEG, cette première étape de compression n'est pas obligatoire.

Contrairement au JPEG, le JPEG 2000 utilise une transformation par ondelettes des pixels de l'image. Il s'agit d'une transformation des pixels de l'image en fréquences où chaque pixel correspond à une et une seule fréquence. Cette opération produit plusieurs sous-images par divisions successives de l'image source [2].Ces sous-images rassemblent chacune un intervalle de fréquences. Pour la majorité des images, les fréquences hautes sont moins nombreuses que les fréquences basses car les fréquences hautes signifient que les pixels de l'image sont très différents les uns des autres, un phénomène rare dans une image. Ensuite vient l'étape de quantification où se produit la destruction des fréquences les plus hautes sont éliminées en fonction d'un taux de compression donné. Néanmoins, si la compression est non destructrice, l'étape de quantification n'est pas effectuée.

Enfin, le JPEG 2000 effectue un codage arithmétique adaptatif des octets résultant de l'opération de compression

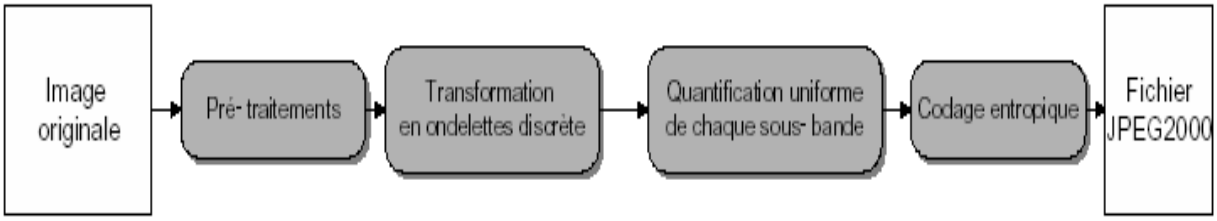

Figure 1. 4 Organigramme de compression JPEG 2000

Récapitulation des résultats obtenus par les différentes méthodes

Les tableaux 1.1, 1.2 et regroupent les principaux résultats obtenus respectivement par les méthodes sans pertes, et avec pertes.

Méthode / Paramètres	RLE	Huffman	LZ77
Taux de compression (%)	30 ~ 40	25 ~ 35	30 ~ 40
Observations	Efficace avec des images constituées de zones uniformes.	Dépend de la statistique des données de l'image.	Très efficace pour les images possédant de fortes redondances

Tableau 1. 1 Caractéristiques des principales méthodes de compression sans pertes [2]

Méthode / Paramètres	QV	DCT (JPEG)	Ondelettes	fractal
Décomposition	Spatiale	Fréquentielle	-Fréquentielle et spatiale -Multirésolution	Spatiale
Temps de compression (s)	10	5	3	180
Temps de décompression (s)	1	1	1	1
Taux pour une image restituée de bonne qualité (%)	90	87,5	91,66	88,88

Tableau 1. 2 Caractéristiques des principales méthodes de compression avec pertes [2]

Paramètres de performance des méthodes de compression d'images

Les principaux paramètres qui permettent d'évaluer une méthode de compression sont : le taux de compression, le débit, le temps d'exécution et les mesures de distorsions. En pratique, il s'agit de réaliser le meilleur compromis possible entre le taux de compression et la qualité de l'image décompressée tout en minimisant le temps d'exécution [Lahdir *et al.*, 2006].De plus, l'algorithme permettant d'implémenter ces méthodes sur ordinateur, doit être robuste aux erreurs de transmission, flexible, et permettre une transmission progressive, c'est à dire transmettre une image dont la qualité à la réception s'affine progressivement.

Taux de compression

Le ***taux de compression*** est une mesure de la performance d'un algorithme de compression de données informatiques. Il est généralement exprimé en pourcentage, et noté τ. Deux définitions sont communément admises :

- L'une définit le taux de compression comme le rapport du volume des données après compression sur le volume initial des données. De ce fait, plus le taux de compression est faible, plus la taille du fichier compressé résultant est faible. Le taux de compression ainsi défini est donné par la formule [1] :

  $$\tau = [\text{Volume final}] / [\text{Volume initial}].$$

  C'est aussi l'inverse du quotient de compression.

- L'autre définition exprime le taux de compression comme le gain en volume rapporté au volume initial des données. *Cette définition est en fait complémentaire de la première*. Plus le taux de compression est élevé, plus la taille du fichier compressé résultant est faible. La formule correspondante s'écrit : $\tau = 1 - ([\text{Volume final}]/ [\text{Volume initial}])$. Dans ce cas, le taux de compression est relié au quotient de compression *q* par l'équation $\tau = 1 - 1/q$.

Débit

Le ***débit*** désigne la quantité d'informations transférée en l'espace d'une seconde.

L'unité de mesure est le bit par seconde (bit/s) et ses différents multiples (Kilo-bits par seconde, Mega-bits par seconde..).

Le choix du débit aura un impact direct sur la taille occupée par fichier final et sur sa qualité (plus le débit est important, plus le fichier final sera « lourd » et sa qualité meilleure). A partir d'un certain débit, l'amélioration de la qualité n'est plus perceptible.

On estime qu'un débit de 1 500 Kbit/s (150 Ko/s) est un bon compromis pour obtenir une vidéo de qualité VHS à partir d'un codec dérivé du MPEG 4 (DIVX, WMV, XVID…), en 25 images par seconde[1].

Temps de compression

Le ***temps de compression*** dépend de plusieurs facteurs :

- facteurs matériels : type du micro-processeur équipant l'ordinateur, quantité de mémoire installée…
- facteurs logiciels : codec et logiciel utilisés, options choisies…
- qualité du rendu final : nombre d'images par seconde, taille de l'image, résolution…

Mesure de la distorsion

Quelques mesures de distorsion ou de qualité permettent, de manière assez fruste, de comparer les algorithmes de compression. Leur utilisation est pratique puisqu'elles ne nécessitent que des calculs automatiques qui sont simples et rapides. La mesure la plus utilisée est l'**erreur quadratique moyenne** (*Mean Square Error*, MSE) calculée entre les pixels originaux et reconstruits [1]:

$$\mathrm{MSE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{p} \in \mathrm{P}} (\mathrm{x_p} - \hat{\mathrm{x}}_\mathrm{p})^2$$

où P désigne l'ensemble des N pixels de l'image, et x et _x sont respectivement les amplitudes de pixel sur les images originale et reconstruite. L'utilisation de cette mesure d'erreur est préférée dans le traitement des données, puisqu'il est possible de développer des algorithmes qui la minimisent. Cependant, la distorsion des images

comprimées est parfois mesurée par d'autres valeurs, comme l'erreur absolue moyenne ou l'erreur absolue maximale. Cette dernière semble plutôt trop sensible aux imperfections occasionnelles. Il est aussi vraisemblable que l'œil tient beaucoup plus compte des erreurs à grandes amplitudes, ce qui favorise la mesure quadratique. Cependant, l'ordonnance des images basée sur ces mesures est en général le même [1]

Au lieu de communiquer les valeurs de MSE, les études donnent en général le **rapport crête signal sur bruit** (*Peak Signal to Noise Ratio*, PSNR). Au lieu de mesurer la distorsion, cette valeur mesure la fidélité de la compression, puisqu'elle est proportionnelle à la qualité. Tout de même, elle est une fonction de MSE ; sa définition et son utilisation proviennent du domaine

Des signaux audio/vidéo [1] :

$$\mathrm{PSNR} = 10 \log_{10} \frac{\mathrm{x}_{\max}^2}{\mathrm{MSE}}$$

Où Xmax désigne la luminance maximale possible. Une valeur de PSNR égale à ∞ correspond à une image parfaitement reconstruite, et elle décroît en fonction de la distorsion. Le PSNR relie donc l'erreur quadratique moyenne à l'énergie maximale de l'image [1].

Conclusion

Les images numériques, qu'elles soient fixes ou animées, ont vocation à être partagées, donc Transmises sur des réseaux de communication. Il faut donc voir plus loin que le simple codage Source. La compression n'est en général qu'une tâche particulière qui s'intègre dans des systèmes où plus de contraintes sont présentes, que l'on doit considérer pendant le développement. Ainsi, on ne doit pas forcément mettre en place un algorithme qui – en tant que méthode indépendante – produit un taux de compression maximal, mais qui dans un système global conduit à la performance optimale. Les critères de performance incluent bien sûr le taux de compression. Mais si la transmission fait partie des services, un faible délai de transmission, la robustesse aux erreurs, la possibilité d'une transmission progressive contribuent à la valeur de l'approche et peuvent justifier un taux moins élevé.

Pour développer de nouvelles méthodes de compression ou des versions améliorées de celles existantes, il est important d'exploiter les avantages de ces différentes méthodes. En effet, les performances d'une méthode de compression dépendent des techniques utilisées dans les étapes de transformation, de quantification et de codage.

Tenant compte de toutes ces constatations, et fixant comme objectif, l'obtention du meilleur compromis possible entre le taux de compression, la qualité de l'image restituée, nous avons élaborée une méthode de compression hybride basé sur la transformée en ondelette discrète DWT et la transformée en cosinus discrète DCT. Cette méthode sera décrite dans le chapitre suivant.

Compression par fusion de techniques

Introduction

L'image est représentée numériquement dans un ensemble de pixels. En général, les pixels voisins sont corrélée avec d'autre et sont redondants. La redondance occupe inutilement l'espace de stockage qui accroît le coût de transmission et la largeur de la bande du système. Par conséquent, il est nécessaire de réduire la redondance d'une image. La réduction peut être obtenue au moyen de techniques de compression de données. L'idée principale derrière la technique de compression est d'utiliser la transformation orthonormale rendant la valeur de pixel plus petit que l'original.

Représentation du schéma de codage avec pertes

La figure II.1 illustre les différentes phases qui permettent d'obtenir une image compressée (et un taux de compression intéressant). L'image (ou une partition de celle-ci) peut être dans un premier temps transformée, les transformations les plus couramment utilisées en compression d'images sont la DCT (transformation en cosinus discrète), la DWT (transformation en ondelette), la DFT (transformation de fourrier discrète) ou parfois la transformation de Karhunen Loevel. Le but de ces transformations et de compacter au mieux l'information contenue Dans l'image, c'est à dire d'avoir un nombre de cœfficients représentatifs aussi faible de possible.

L'étape suivante est la quantification des coefficients transformes, elle permet de coder au mieux des coefficients souvent réels ou complexes en introduisant une erreur

de quantification. La quantification est donc la phase qui provoque une dégradation dans l'image reconstruite mais c'est aussi cette opération qui permet d'obtenir des taux de compressions beaucoup plus importants que dans le cas d'une compression sans perte.

Enfin la dernière étape est le codage des coefficients quantifies, les méthodes utilisées sont les mêmes méthodes que celles qui sont utilisées dans le cas de la compression sans perte. C'est cette étape qui va exploiter au mieux la redondance d'information qui s'est accumulée après les étapes de transformation et de quantification.

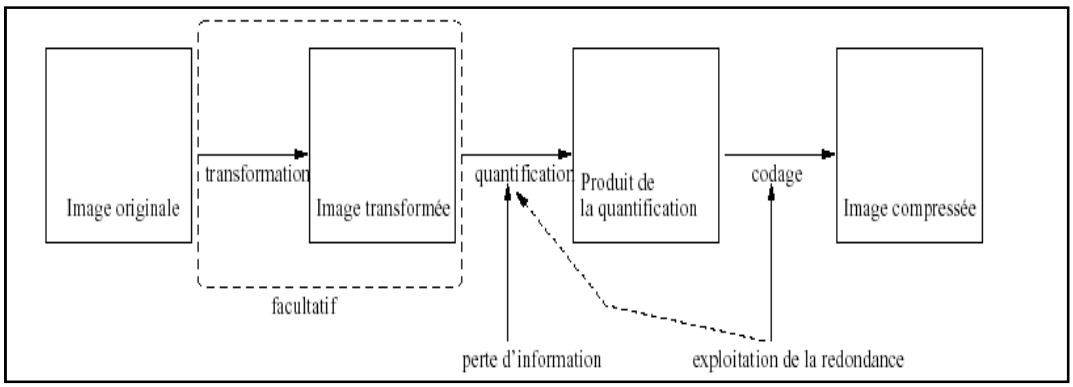

Figure 2. 1 Schéma de codage avec perte

Méthodes de transformation pour la compression des données

Transformée on cosinus discrète (DCT)

Une DCT représente les points de données d'entrée sous la forme d'une somme de fonctions cosinus qui sont oscillantes à des fréquences différentes et des grandeurs. Il y a principalement deux types de DCT: DCT unidimensionnelle (1-D) et DCT bidimensionnelle (2-D). Comme une image est représentée comme une double matrice bidimensionnelle, pour ce travail de recherche, 2-D DCT est considéré. La DCT 2-D pour un N X N séquence d'entrée peut être définie comme suit [3]:

$$D_{DCT}(i,j) = \frac{1}{\sqrt{2N}} B(i)B(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} M(x,y) \cos\left[\frac{2x+1}{2N} i\pi\right] \cos\left[\frac{2y+1}{2N} j\pi\right] \quad (2.1)$$

Où

$$B(u) = \begin{cases} \frac{1}{\sqrt{2}} & if \quad u = 0 \\ 1 & if \quad u > 0 \end{cases} \quad (2.2)$$

M (x, y) est la donnée d'entrée de taille *x ** y dans la compression DCT, la quasi-totalité de l'information est concentrée dans un petit nombre des coefficients de basse fréquence. Ces basses fréquences coefficients sont également connues comme composantes continues et le reste des composants sont AC composants. Les coefficients DCT sont ensuite quantifiés en utilisant une table de quantification en utilisant une table de quantification Q de 8X8 [9], tel que décrit dans le Joint Photographic Expert Group (JPEG) standard. La quantification est Obtenue en divisant chaque élément des données transformées par l'élément correspondant dans la matrice de quantification Q et arrondi au plus proche valeur entière comme indiqué dans l'équation. (2,3).

$$D_{quant}(i,j) = round\left(\frac{D_{DCT}(i,j)}{Q(i,j)}\right) \tag{2.3}$$

La compression complémentaire peut être obtenue en appliquant le facteur d'échelle appropriée. Afin de, reconstruire les données de sortie, le rééchelonnement et la dé-quantification doit être effectué comme indiqué dans l'équation 2.4

$$D_{dequant}(i,j) = Q(i,j) \times D_{quant}(i,j) \tag{2.4}$$

La matrice de dé-quantification est ensuite transformée à nouveau en utilisant le 2D-DCT inverse [3]. L'équation de la 2-D transformée inverse DCT est donnée dans l'Equation. (2.5).

$$D_{IDCT}(i,j) = \frac{1}{\sqrt{2N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} B(i)B(j)D_{dequant}(i,j) \cos\left[\frac{2x+1}{2N} i\pi\right] \cos\left[\frac{2y+1}{2N} j\pi\right] \tag{2.5}$$

Où

$$B(u) = \begin{cases} \frac{1}{\sqrt{2}} & if \quad u = 0 \\ 1 & if \quad u > 0 \end{cases} \tag{2.6}$$

Limites de DCT

Les figures 2.2 et 2.3 montrent l'image originale et les images reconstruites à deux différents niveaux de compression à l'aide 2 - D DCT [19]. Pour le plus faible ratio de compression, la distorsion est inaperçue par la perception visuelle humaine, qui peut être vu dans la figure 2.2-(b). Afin d'atteindre une compression plus élevée il est nécessaire d'appliquer la quantification suivie d'une mise à

l'échelle des coefficients transformés, Pour un tel taux de compression la DCT à la suite de deux limites :

- **Blocage des artefacts**: blocage des Artefacts est une distorsion qui apparaît en raison de lourdes compression et apparaît comme des blocs de pixels de grande taille. Pour un ratio de compression élevé, les notables "d'artefacts de blocs" à travers les limites des blocs ne peut être négligé. L'exemple de l'apparence d'un artefact de blocage dû à la compression élevée est indiqué dans Figure 2.2-(c)[19].

- **faux contours :** Les faux contours se produit lorsque la zone de douceur classée d'une image est déformée par une aberration qui ressemble à une carte de contours pour les images spécifiques ayant les zones ombragées progressivement [5]. La principale cause de l'effet de faux contours est la lourde quantification des coefficients de transformation.

Un exemple de faux contours peuvent être observée dans la figure 2.3-(c).

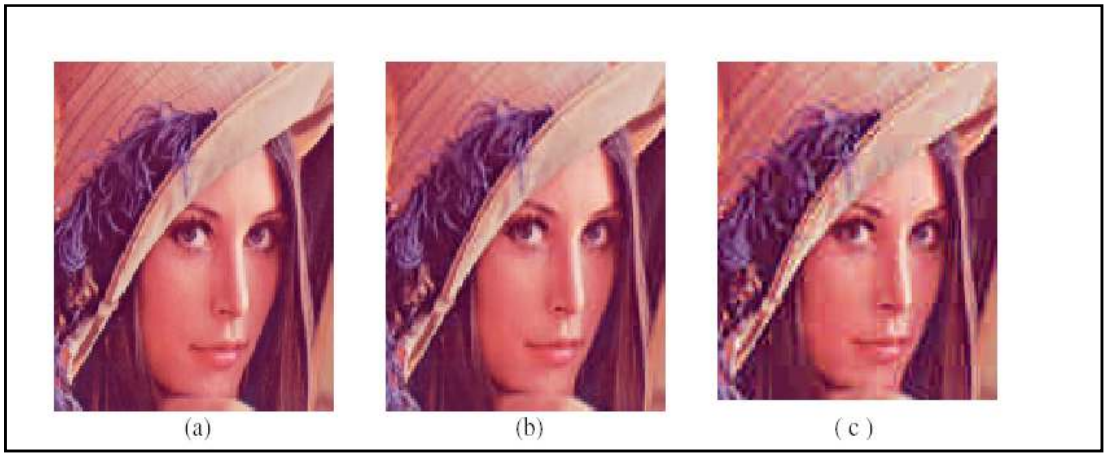

Figure 2. 2 ***Illustration de la compression en utilisant la DCT: (a) Image[1] original, RC : (b) 88%, (c) 96%.***

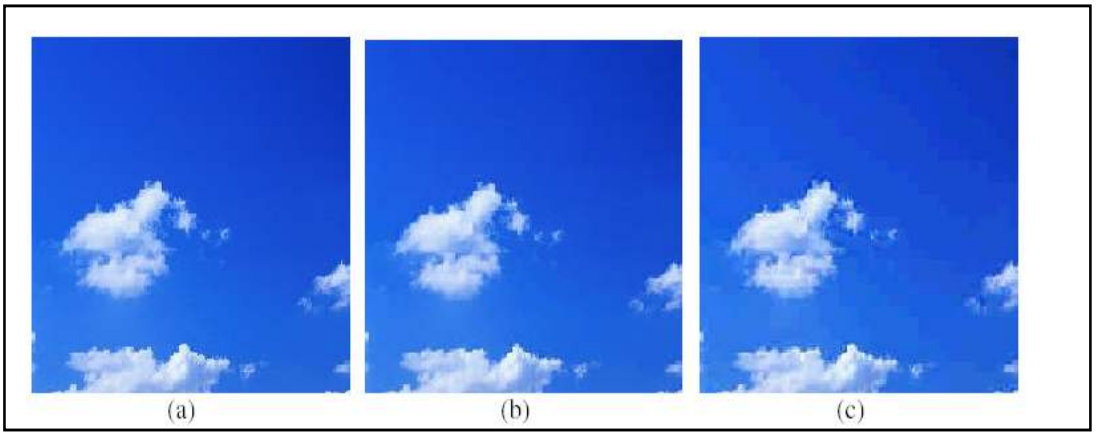

Figure 2. 3 Illustration de la compression en utilisant la DCT:
(a) Image[2] original, RC : (b) 88%, (c) 96%.

Transformée en ondelette discrète (DWT)

La transformée en Ondelettes discrète (DWT) [18] est une opération qui décompose un signal (une série d'échantillons numériques) en deux parties par projection sur un filtre passe-bas L et un filtre passe-haut H. La partie résultant du filtrage passe-bas représente une approximation du signal d'origine à la nouvelle résolution (ou échelle), celle résultant du filtrage passe-haut représentant les détails perdus entre les deux résolutions.

Puisqu'une image est typiquement un signal en deux dimensions, une TO dyadique est réalisée, en appliquant d'abord les filtres L et H sur les échantillons ligne par ligne, puis en réappliquant les mêmes filtres sur les échantillons résultants, mais colonne par colonne cette fois-ci. Au final, l'image est divisée en 4 parties, les sous-images LL, LH, HL et HH comme présenté sur la figure 2.4 (a). La sous-image LL fournit une version à l'échelle ½ de l'image d'origine, LH, HL et HH représentant les détails perdus respectivement dans les directions horizontale, verticale et diagonale. La TOD peut être répétée sur LL pour obtenir plusieurs niveaux de résolution.

La figure 2.4 (b) présente une image décomposée en trois niveaux de résolution [18].

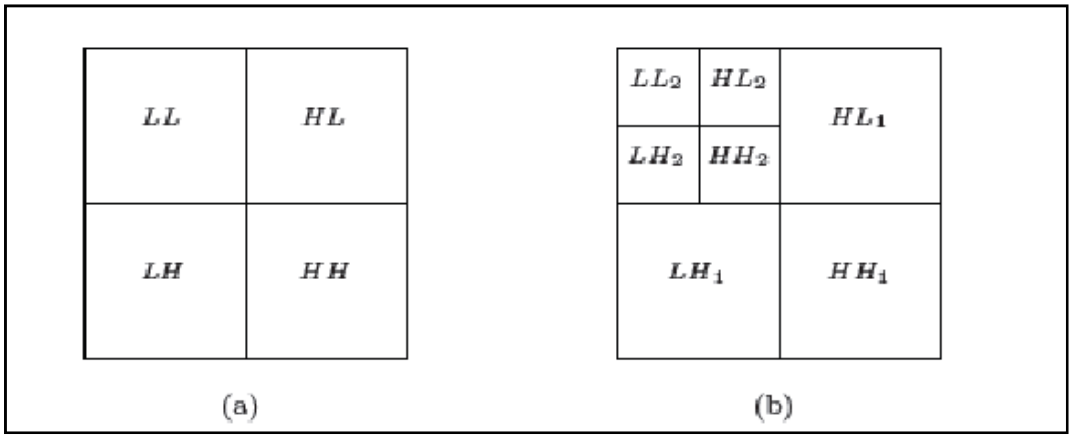

Figure 2. 4 La TO dyadique appliquée une fois (a) et deux fois (b).

- **Discussion**

Pour un ratio de compression plus faible, l'algorithme DCT présente plus de caractéristiques de compactage d'énergie et nécessite moins de complexité de calcul par rapport à d'autres méthodes de compression (DFT, DST, WHT, et DWT).[19] Cependant, il introduit des artefacts de blocage et un effet de faux contours, D'autre part, la DWT est la méthode de compression multi-résolution, une image peut être obtenue dans des résolutions différentes en éliminant les coefficients de détail et en ne prenant que le coefficient approximatif. Mais, la caractéristique de compactage de l'énergie de la DWT est inférieure à celle de la DCT et nécessite plus de calcul. Par conséquent, une transformation multiple peut être mis-en œuvre afin de compenser l'inconvénient d'une méthode par rapport à l'autre. Dans ce travail, un algorithme hybride DWT-DCT a été proposé.

Algorithme de compression proposée

Il est admis que les méthodes de compression par transformation sont les plus efficaces en termes de taux de compression. Parmi celles-ci, la méthode JPEG largement utilisée dans les applications existantes montre ses limites pour les forts taux de compression avec l'apparition de l'effet de bloc (importante dégradation de l'image originale). La méthode de transformation par les Ondelettes quant à elle est une méthode globale qui travaille sur l'ensemble de L'image sans découpage et fait apparaître un flou pour les forts taux de compression ; c'est en l'occurrence la technique utilisée dans le format Jpeg2000.

Notre algorithme de compression d'image utilise deux transformations. Première transformation est DWT en plusieurs niveaux, convertit les images en sous-bandes, et

la seconde transformation est DCT bidimensionnelle appliquée sur chaque des blocs de données des sous bandes de basse fréquence. La DCT est très important pour notre approche elle se partage avec la DWT pour obtenir les domaines de plus hautes fréquences. DCT joue le rôle principal pour convertir la sous-bande en courant continue alternatif coefficients, comme toute technique de compression d'image, après cette étape de transformation, on entame la quantification et le codage pour finir notre algorithme

Description des étapes de l'algorithme

Étape de transformation

Utilisation de la Transformée en Ondelettes Discrète (DWT)

La transformée en Ondelettes a la capacité d'offrir une certaine information sur la fréquence du domaine temporel simultanément. Dans cette transformée, du domaine du temps est passé à travers des filtres passe-bas et passe-haut pour extraire des fréquences basses et hautes, respectivement. Ce processus est répété plusieurs fois et à chaque fois une section du signal est tirée. L'analyse du signal DWT divise en deux classes (approximation et détail) par décomposition du signal pour différentes bandes de fréquences et échelles [11]. Figure 2.5. Montre trois niveaux de décomposition de l'image.

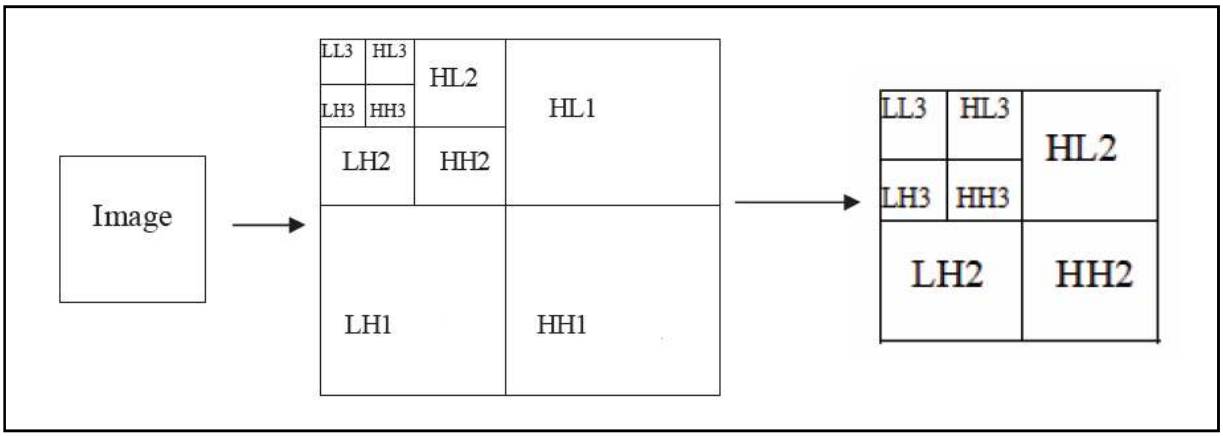

Figure 2. 5 Première étape de l'algorithme hybride DWT-DCT (transformation de l'image en utilisant la transformée en ondelette discrète DWT)

La décomposition en Ondelettes a quelques caractéristiques importantes ; la plupart des coefficients pour les composants haute fréquence (LH, HL et HH) sont nulles ou

négligeables [12]. Cela reflète le fait que la majeure partie de l'information importante est en "LL2" sous-bande. Le DWT est un bon choix de transformation accomplit une dé-corrélation pour les pixels, tout en fournissant simultanément une représentation dans laquelle la plupart de l'énergie est généralement limitée à quelques coefficients [13], tandis que d'autres sous-bandes (LH1, HL1 et HH1) sont ignorés. C'est la clé pour parvenir à un taux de compression élevé.

Utilisation transformée en cosinus discrète (DCT)

La DCT est la deuxième transformation utilisée dans notre algorithme, qui est applicable sur chaque bloc de la sous-bande "LL2" " comme il est montré dans la figure2.6.

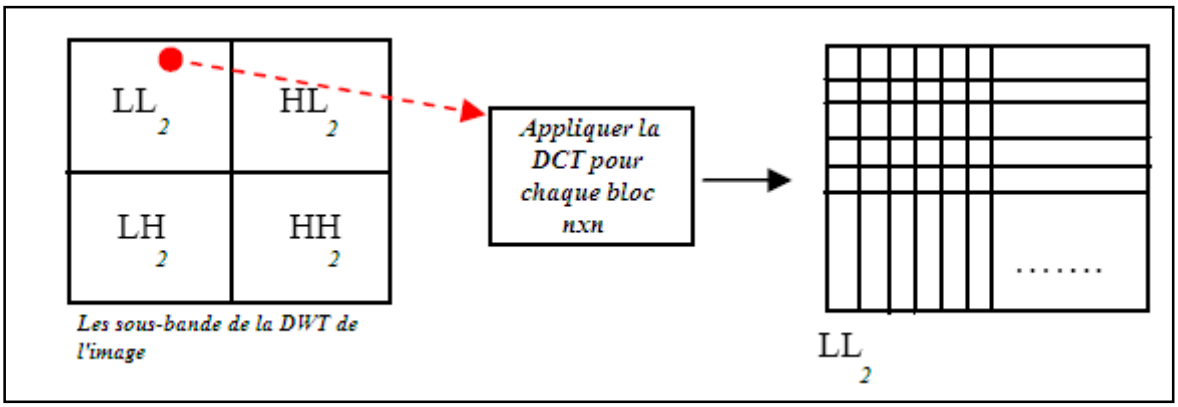

Figure 2. 6 Transformé en cosinus discrète DCT appliqué pour chaque bloc nxn de la sous-bande LL2

L'énergie dans les coefficients transformés est concentrée sur le coin supérieur gauche de la matrice de coefficients. Les coefficients du haut à gauche correspondent à des fréquences basses: il y a un «pic» de l'énergie dans ce domaine et les valeurs des coefficients diminuent rapidement en bas à droite de la matrice, [2,3]. Les coefficients DCT sont décorrélé, ce qui signifie que beaucoup de coefficients avec les petites valeurs peuvent être éliminés sans affecter considérablement la qualité de l'image [2,3].L'une des différences clés entre les applications de la transformée en cosinus discrète DCT et la transformée en Ondelettes discrète (DWT),est que cette dernière s'appliquent généralement à une image comme un seul bloc ou une grande région rectangulaire de l'image, tandis que pour la DCT [6,8,9], elles est de plus en plus compliquée à calculer pour les blocs de grande taille, pour cette raison on a utilisé des

blocs de "n × n" pour la transformée en cosinus discrète DCT, alors que pour la DWT elle sera plus efficace à appliquer aux images complètes pour obtenir un bon taux de compression [10].

Étape de quantification

Quantification Level1

Dans la Figure 2.6 ci-dessus, "Quantification level1" est utilisé pour réduire la taille des données pour LL2, en sélectionnant le ratio de la valeur maximale pour LL2 comme indiqué dans l'équation suivante (2.7):

$$Q1 = Quality \times \max(LL_2) \tag{2.7}$$

$$LL_2 = \text{round}\left(\frac{LL_2}{Q1}\right) \tag{2.8}$$

"Qualité" valeur est utilisée dans l'équation (2.7), elle affecte la qualité d'image. Par exemple, «la qualité = 0,005", cela signifie calculer 0,5% de la valeur maximale en LV2 à diviser par toutes les valeurs de LL2. Ce processus aide les valeurs de la LL2 à être plus convergente (c'est à dire plus corrélés). Par ailleurs ce processus de quantification conduit à obtenir une faible qualité d'image, si la qualité > 1%.

Quantification Level2

Pour chaque block de "2 × 2" les coefficients de LL2 sont transformés par la DCT, puis divisé par "Q2", en utilisant un processus qui supprime les coefficients non significatifs et remplis de zéros la LL2 transformée. La "Q2" peut être calculée comme suit [25]:

$$Q2(i,j) = \begin{cases} 1, & \text{if}(i=1 \text{ and } j=1) \\ i+j+R, & \text{if}(i\neq 1 \text{ and } j\neq 1) \end{cases} \tag{2.9}$$

En utilisant deux dimensions DCT

L'énergie dans les coefficients transformés est concentrée sur le coin supérieur gauche de la matrice de coefficients. Les coefficients du haut à gauche correspondent à basses fréquences :

Il y a un "pic" de l'énergie dans ce domaine et les coefficients les valeurs diminuent rapidement en bas à droite de la matrice, ce qui signifie que les fréquences plus élevées coefficients [24].

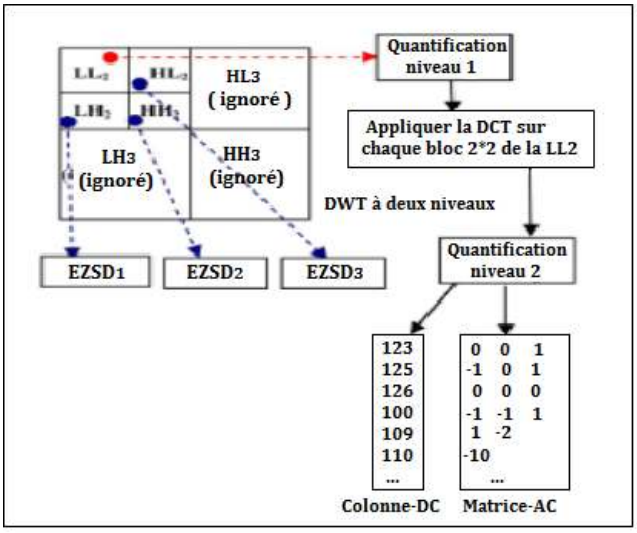

Figure 2. 7 sous-bande *LL2* quantifiés et transformés par DCT pour chaque 2 × 2 blocs

Les coefficients DCT sont dé-corrélé, ce qui signifie que bon nombre des coefficients à petites valeurs peuvent être éliminés sans affecter considérablement la qualité de l'image. Les coefficients dé-corrélés d'une matrice compacte peuvent être compressée beaucoup plus efficacement qu'une matrice à pixels d'image fortement corrélées [25]. Les équations suivantes illustre la fonction de la DCT et la DCT inverse pour des matrices bidimensionnelles "2 × 2":

$$C(u,v) = a(u)a(v) \cdot \sum_{x=0}^{3}\sum_{y=0}^{3} f(x,y)\cos\left[\frac{(2x+1)u\pi}{4}\right]\cos\left[\frac{(2y+1)v\pi}{4}\right] \qquad (2.10)$$

Ou :

$$a(u) = \sqrt{\frac{1}{2}},\ for\ u=0,\ \ a(u)=1,\ \ for\ \ u\neq 0 \qquad (2.11)$$

Avec :

$$f(x,y) = \sum_{u=0}^{3}\sum_{v=0}^{3} a(u)a(v)C(u,v)\cos\left[\frac{(2x+1)\ u\pi}{4}\right]\cos\left[\frac{(2y+1)\ v\pi}{4}\right] \qquad (2.12)$$

Appliquer l'algorithme qui minimise la taille de la matrice

La DWT et la DCT sont utilisées pour un nombre croissant de coefficients de haute fréquence, mais ceci n'est pas suffisant pour notre approche, pour cette raison nous avons introduit un nouvel algorithme qui minimise la taille de l'AC-Matrice dans un tableau. Cet algorithme basé sur le poids aléatoire des valeurs. L'équation suivante représente la conversion AC-Matrice dans un tableau :

$$Arr(L) = W(1) * AC(L,1) + W(2) * AC(L,2) + W(3) * AC(L,3) \quad (2.13)$$

Remarque: "AC" représente l'AC-Matrix.
L = 1, 2, 3, · · · taille de la colonne de l'AC-Matrix.

Le "W" dans l'équation (2.13) représente les valeurs aléatoires du poids, générés par la fonction aléatoire dans le langage MATLAB, les valeurs de "W" sont comprises entre {0 · · · 1}, par exemple : W = {0, 1, 0,65, 0,423}. Les valeurs aléatoires du poids sont multiplier avec chaque ligne de l'AC-Matrix, pour produire une seule valeur en virgule flottante "Arr", cette idée est similaire aux équations de réseaux de neurones, ces dernières multiplie les valeurs mises à jour de poids avec les neurones d'entrée pour produire une seule sortie neurone [35]. L'algorithme de la minimisation de la taille de la matrice est présenté dans la liste suivante **(Liste1)** :

Let *Pos*=1

***W*=** Génération_de_valeurs_de_poids_aléatoires **();**

I=1;

Tant que (I<taille de ligne LL2)

J=1;

Tant que (J<taille de colonne LL2)

%% Appliquer la DCT pour chaque bloc 2 × 2 de la LL2

Block2x2=**Appliquer_DCT**(LL2[I, J]);

%% convertit immédiatement un bloc 2*2 en un vecteur 1*4

L1x4=**Convertion_Block_en_tableau**(Bloc2x2);

DC[*Pos*]=L1x4[1]; %% stocker la première valeur dans la Colonne-DC

Arr[*Pos*]=0; %% initialisation avant l'addition

pour K=2 à 4

Arr[*Pos*]= Arr[*Pos*] + L1x4[*K*] * *W*(*K*-1); %% appliquer Eq(2.13)

FIN**; %%** *fin pour*

Pos++;

J=J+2;

Fin**;** *%% fin tant que*

I=I+2;

Fin; *%% fin tant que*

Liste1 : Algorithme de minimisation de la taille de la matrice

Avant d'appliquer l'algorithme de minimisation de la taille de la matrice, notre algorithme de compression calcule la probabilité de la Matrice-AC. Ces probabilités sont appelées : données limitées, qui sont utilisées plus tard dans l'algorithme de décompression, les données limitées sont stockées dans l'en-tête pour le fichier compressé, à cause des données incompressibles.

L'étape finale dans cette section est le codage arithmétique qui consiste à convertir le flux de données en une seule valeur à virgule flottante. Ces valeurs de sortie dans la gamme inférieure à un et supérieure à zéro, lorsque cette valeur est décodée, on obtient le flux exact de données. Le codage arithmétique besoin de calculer la probabilité de toutes les données et d'attribuer à chaque donnée une plage, cette dernière comporte la minima et la maxima de la valeur [24,25].

T-matrice de codage

La colonne-DC contient des valeurs entières, ces valeurs sont obtenues à partir de l'article perméable. Le codage arithmétique est incapable de compresser cette colonne parce que ces valeurs sont corrélées, pour cette raison colonne-DC est divisée en 128 parties, et chaque partition est transformée par la DCT (Voir équation (2.13)) en tableau à une dimension en utilisant (u = 0 et x = 0) dans l'équation (2.14) pour que ça soit une DCT unidimensionnel, puis on utilise un processus de quantification pour chaque ligne, comme indiqué dans l'équation suivante :

$$Q'(n) = Q(n-1)+2 \tag{2.14}$$

Note: n = 2, 3, · · ·, 128.
La valeur initiale de Q (1) = 12,8.

La première valeur dans l'équation ci-dessus (2.14) est Q (1) = 12,8, en raison de la longueur de chaque ligne qui est de 128. Chaque 128 valeurs sont transformées et stockés sous forme d'une ligne dans une matrice appelée « Transformée-matrice » (T-Matrice). Cette partie de notre algorithme de compression ressemble à la technique JPEG, comme le montre la Figure (2.9). L'idée d'utiliser la T-Matrice, est que nous avons besoin d'augmenter le nombre de zéros pour obtenir des valeurs plus dé-corrélées, ce processus augmente le taux de compression. Chaque ligne de la T-Matrice se compose de la valeur DC et le flux des coefficients AC, maintenant nous avons besoin de numériser la T-matrice, colonne par colonne pour la convertir dans un tableau unidimensionnel, faire ensuite une compression par Run Length Encoding (RLE) et enfin un codage arithmétique. RLE compte les coefficients répétées, et fait ensuite référence à des coefficients de répétition d'une valeur, ce processus de réduction de la longueur des données répétées, puis on applique un codage arithmétique pour convertir ces données réduites en morceaux.

Elimination des zéros et stockage des données (EZSD)

Cette technique a été utilisée pour éliminer le bloc des zéros, et ensuite l'emmagasiné dans un tableau. Cette méthode est particulièrement utile pour réduire la taille des hautes-fréquences des sous-bandes (i.e HL2, LH2 et HH2) ou les matrices qui contiennent plus de zéros, EZSD appliquée sur chaque sous-bande indépendamment. Avant d'appliquer l'algorithme EZSD sur chaque haute-fréquence des sous-bandes (i.e HL2, LH2 et HH2) on applique une quantification level1 (voir équation (2.8)), avec une valeur de qualité > = 0,01. Ceci pour la suppression des coefficients non significatifs et l'augmentation du nombre des zéros.

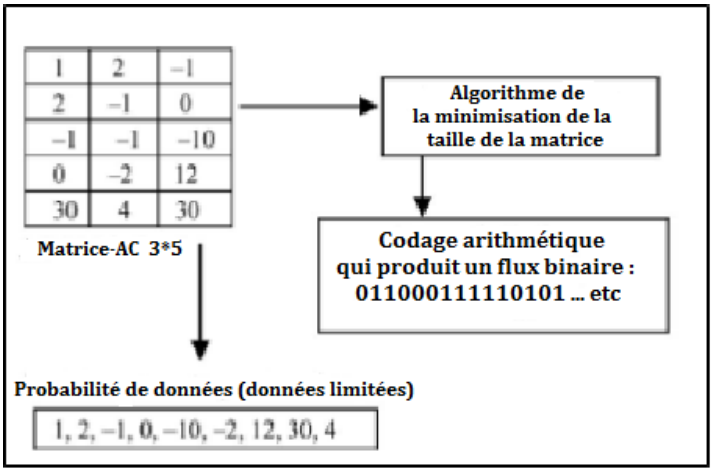

Figure 2. 8 Illustre la probabilité de données (données limitées) pour la Matrice-AC 3 × 5.

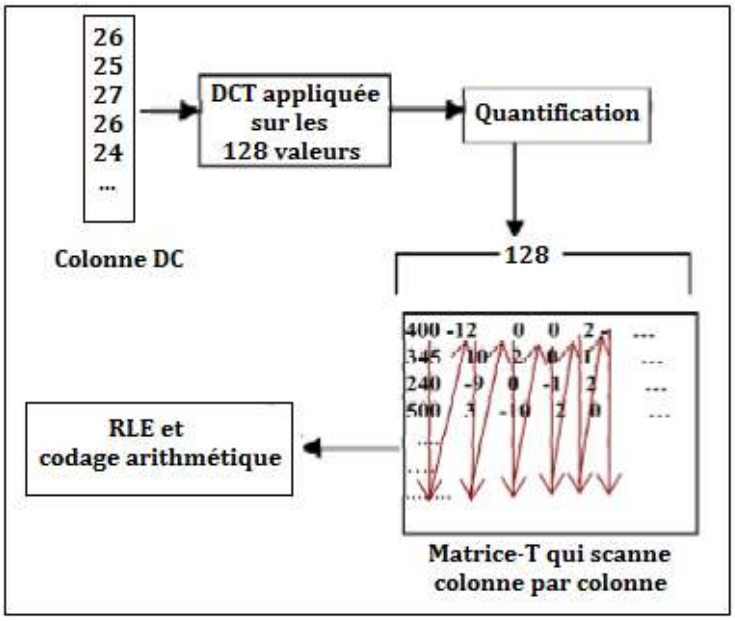

Figure 2. 9 Technique de codage de la matrice-T.

L'algorithme EZSD commence à partitionner les hautes fréquences sous-bandes dans le non-chevauchement des blocs 8 × 8, puis commence à chercher pour le bloc non nulle (i.e recherche des données à l'intérieur d'un bloc)si ce bloc contient des données, ce bloc sera stocké dans le tableau appelé « matrice réduite », et la position de ce bloc est stocké dans un nouveau tableau appelé « Position ». Si le bloc ne contient que des zéros, ce bloc sera ignoré, et on saute au bloc suivant, l'algorithme de résumé de la liste 2.

Taille bloc =8; %% Bloc de taille 8x8

I=1; *LOC*=1

Tant que (I< taille de la colonne de LH2)

J=1;

While (J< taille de la ligne de LH2)

%% lire un bloc 8x8 des hautes frequencies de la sous-bande

*Block[1..taille bloc*taille bloc]*=**Lire_Bloc_de_la_Matrice**(I,J);

%% cette fonction coche le contenu de la case "Bloquer", a-t-il une valeur non nulle?

Si (**Cocher_Bloc**(*Bloc*, *taille bloc*) = = 'NO')

*POSTION [LOC] =I; POSTION [LOC+*1*] =J;*

%% Enregistrer le localisation d'origine du bloc contenant les données non nulles

*LOC=LOC+*2;

%% Transférer le contenu du bloc à un tableau unidimensionnelle

For *K=1: taille_bloc*taille_bloc*

Tableau_réduit[P]= Bloc[k];

++P;

Fin*;*

Fin;

J=J+ taille_bloc;

Fin;

I=I+ taille_bloc;

End;

Liste 2. Algorithme EZSD.

La figure 2.8 montre que les haute-fréquences de la sous-bande partitionnée en bloc 8x8 sont codées par l'algorithme EZSD. L'algorithme EZSD appliquée sur chaque haute-fréquence des sous-bandes. Le problème de la technique EZSD est que le tableau réduit encore les zéros, c'est parce que certains blocs contiennent peu de données dans certains endroits, tandis que d'autres contiennent des zéros. Ce problème a été résolu en supprimant les zéros de gamme réduite. Figure 2.9 illustre l'élimination des zéros de la matrice réduite. L'idée d'éliminer tout juste huit zéros ou moins du

«Tableau réduit », ceci pour répéter le nombre des zéros. Pour l'exemple ci-dessus "0, 8" répété 4 fois, "0, 1" et "0, 4" sont répétées 2 fois, ce procédé augmente le taux de compression avec un codage arithmétique.

Notre algorithme de décompression

Notre algorithme de décompression représente l'inverse de notre algorithme de compression, qui est composé de quatre phases illustrées dans les points suivants:

1) Première phase; décodage du domaine des haute-fréquences pour la DWT à deux niveau (i.e HL2, LH2 et HH2).

2) Deuxième phase; décodage de la Colonne-DC.
3) Troisième phase; décodage de la Matrice-AC en utilisant LSS-algorithme.
4) Quatrième phase; combiner la Colonne-DC avec la Matrice-AC pour reconstruire environ les coefficients de LL2, et finalement la DWT inverse s'applique pour obtenir une image décompressée.

Décodage des haute-fréquence des sous-bandes

Dans cette section les hautes fréquences sont décodées par l'expansion des données réduites. L'expansion signifiant la recherche d'un zéro suivi d'un numéro, ce nombre est nul comte, l'algorithme répète les zéros suivant le nombre dans le nouveau tableau appelé « Tableau d'expansion ». Enfin remplacer chaque 64-données d'un bloc dans une matrice vide, selon lui est la position dans la matrice (Voir la liste 2). Cet algorithme de décodage est appliqué à chaque matrice réduite pour chaque sous-bande de manière indépendante. La figure 2.9 illustre simplement le décodage de la LH2.

Décodage de la Colonne-DC

Run Length décodage (RLD) est un décodage arithmétique qui produit un tableau unidimensionnel et qui contient les données d'origine pour la T-Matrice. Ce tableau est lu élément par élément, et placés dans les colonnes de la T-Matrice. Les lignes

sont complètes. Après avoir génère la T-Matrice, on applique la Quantification inverse, maintenant on fait une dot-multiplication des coefficients de lignes avec "Q (n)" (Voir l'équation (2.14)), puis on applique la DCT inverse à une dimension sur chaque ligne pour produire 128 valeurs (voir équation (2.12)), ce processus sera répètera jusqu'à ce que toutes les lignes soient complètes. La figure 2.10 illustre la Colonne-DC de codage.

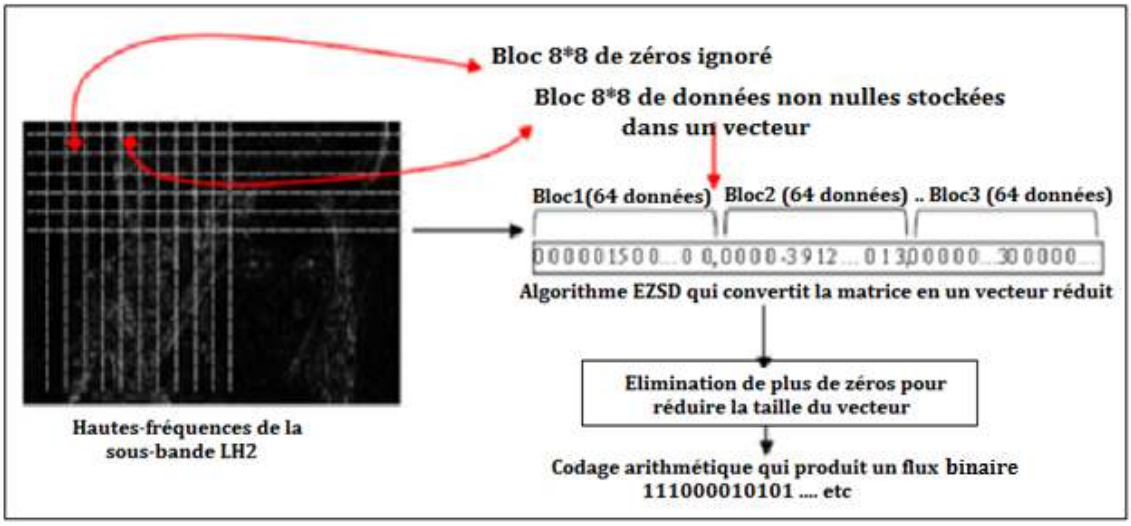

Figure 2. 10 : Algorithme EZSD appliquée sur LH2 sous-bande.

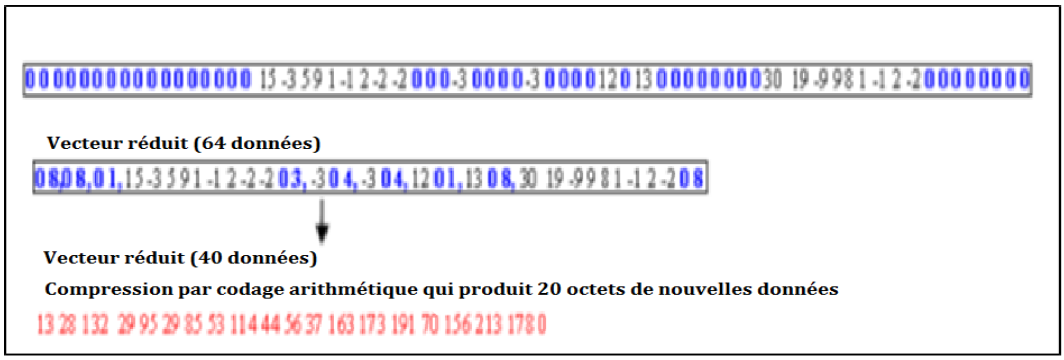

Figure 2. 11 Illustre l'élimination de plus de zéros du tableau réduit.

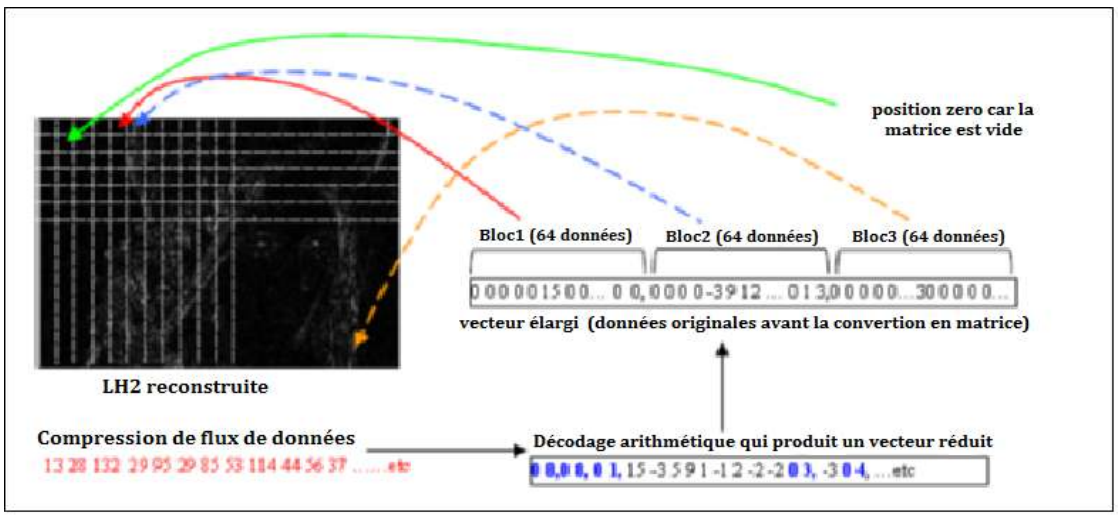

Figure 2. 12 Illustre la première phase de décodage pour LH2 reconstruite.

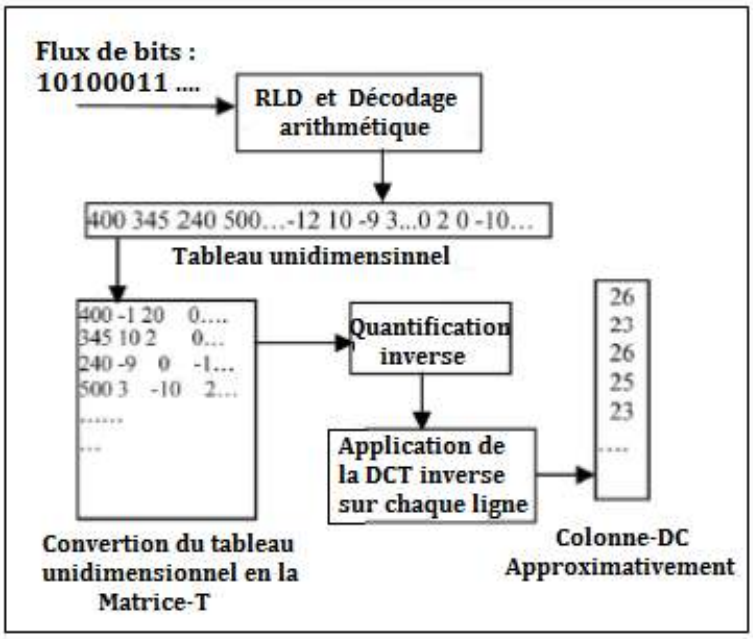

Figure 2. 13 Deuxième phase de notre algorithme de décompression "décodage de la Colonne-DC".

Algorithme de recherche séquentielle limitée de décodage

Cet algorithme est utilisé pour reconstruire Matrice-AC, en utilisant les données réduites au minimum (voir équation (2.13)), et au hasard des poids-Values. Cet algorithme dépend des pointeurs pour trouver les données manquantes à l'intérieur des commanditaires des données pour la Matrice-AC. Si la limite de données est interrompue ou détruite la Matrice-AC ne pourra pas revenir en arrière. La figure 2.12 montre la Matrice-AC décodée par l'algorithme LSS. L'algorithme LSS est conçu pour trouver les données manquantes à l'intérieur des données Limitées, à l'aide de trois pointeurs. Ces pointeurs sont réfère à l'index qui se trouve dans les données Limitées, la valeur initiale de ces pointeurs est "1" (première localisation dans un donnée Limitée). Ces trois pointeurs sont appelés S1, S2 et S3; ces pointeurs sont incrémentés à chaque itération (autrement dit, ces trois pointeurs travail comme une horloge numérique, où S3 S1, S2 représentent les heures, les minutes et les secondes respectivement). Pour illustrer l'algorithme LSS que nous avons la matrice 2*3 suivante

:

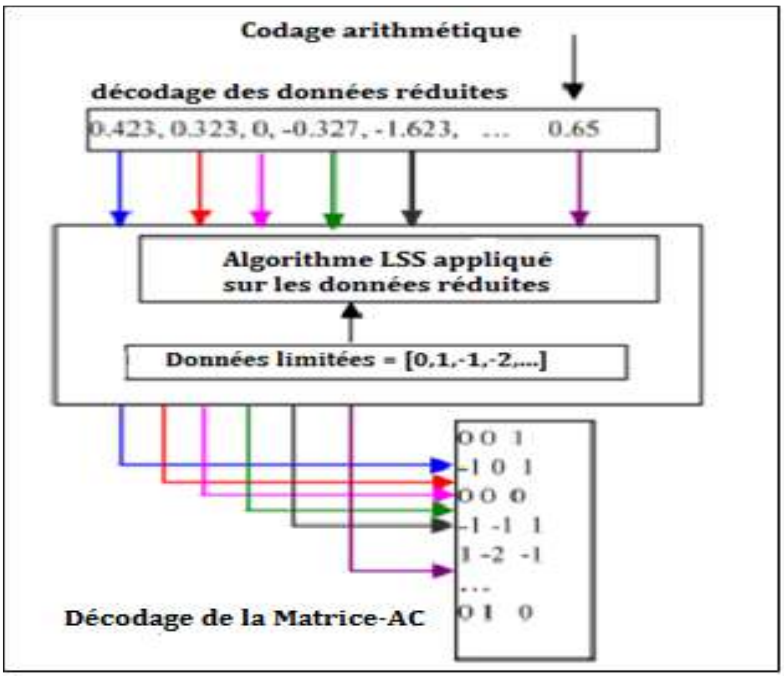

Figure 2. 14 Troisième phase de notre algorithme de décompression "décodage de la Matrice-AC".

Exemple

30	1	0
19	1	**1**

Dans l'exemple ci-dessus la matrice va comprimer en utilisant l'algorithme qui minimise la taille de la matrice afin de générer les données minimisées; M (1) = 3,65 et M (2) = 2,973, les commanditaires des données = {30, 19, 1, 0}. Maintenant, l'algorithme LSS estime les valeurs de la matrice 2 * 3, en utilisant les données limitées et les valeurs du poids aléatoire= {0.1, 0.65, 0.423 et}. La première étape de notre algorithme de décompression, assigner S1 = S2 = S3 = 1, puis le calcul du résultat par l'équation suivante:

$$Estimated = \sum_{i=1}^{3} W(i) \times Limited\left(S(i)\right) \tag{2.15}$$

W: poids généré.

Limitée: des données limitées.

S (i): pointeurs 1,2 et 3.

LSS-algorithme calcule une «estimation» à chaque itération et la compare avec "M (i)". Si elle est nulle, les valeurs estimées sont situées à des emplacements = {S1, S2 et S3}. Si elle est différente de zéro, l'algorithme va continuer à trouver les valeurs originales

à l'intérieur des données limitées. la figure 2.14 illustre l'algorithme LSS qui estime les valeurs de la matrice pour l'exemple ci-dessus.
Lors de la quatrième phase, notre algorithme de décompression combine la Colonne-DC et la Matrice-AC, puis il applique la DCT inverse (voir équation (2.12)) pour reconstruire la sous-bande LL2 , puis on appliquant la DWT inverse au premier niveau sur LL2, HL2, LH2 et HH2 pour produire approximativement LL1. La DWT inverse au deuxième niveau est appliquée sur la LL1 (les autres sous-bandes sont à zéros) pour une reconstruction approximative de l'image originale.

Conclusion

Cette recherche présente une nouvelle approche de compression de l'image et des algorithmes de décompression, basé sur les deux transformées importantes (DCT et DWT), Les algorithmes en question sont : l'algorithme qui minimise la taille de la matrice et l'Algorithme LSS. Cette recherche présente certains avantages illustrés dans les étapes suivantes:

1) L'utilisation de deux transformées, ce qui apporte à notre algorithme de compression une augmentation du nombre de coefficients de haute-fréquence qui conduit à une augmentation du taux de compression.
2) La quantification des niveaux de la sous-bande LL2 et un niveau de quantification des sous-bandes de hautes fréquences par la DWT au second niveau. Ce processus apporte un nombre croissant de zéros dans la matrice.
3) L'algorithme qui minimise la taille de la matrice est utilisé pour convertir les trois coefficients de la Matrice-AC en une seule valeur de point flottant. Ce processus augmente le taux de compression d'une part, et d'autre part la qualité des coefficients de haute fréquence.
4) L'algorithme LSS représente le noyau de notre algorithme de décompression, il convertit le tableau à une dimension dans la Matrice-AC. Cet algorithme est en fonction sur les valeurs des poids aléatoires.
5) Notre approche donne une bonne qualité d'image visuelle, parce qu'elle élimine le flou et les artefacts causés par niveaux multiples de la DWT.

6) dans cette recherche l'algorithme EZSD est utilisé pour supprimer un grand nombre de zéros, et en même temps convertir les hautes fréquences de la sous-bande en matrice, ce processus augmente le taux de compression.

Description du logiciel élaboré

Introduction

Pour la réalisation d'une interface d'analyse, deux solutions s'offraient :

- Utilisation d'un langage de programmation qui offre une richesse graphique conséquente.
- Utilisation d'un langage dédie au calcul scientifique et qui offre un interprète évolué.

Cette ouverture permet l'ajout de fonction à ce même, qui sont assemblées sous forme de boite à outils, ce qui étend le champ d'action à des domaines aussi varie le contrôle, l'optimisation, le traitement d'image et bien évidemment le traitement du signal.

Notre choix s'est porté sur la deuxième catégorie qui inclue des logiciels tels que : MATHEMATICA, MAPLE, MATHCAD et MATLAB. Du fait qu'il correspond exactement à nos besoins logiciels, nous nous sommes fixés finalement sur MATLAB.

Environnement de travail

Matériel utilisé

L'implémentation de notre application « CIC » a été réalisée sur un micro-portable fonctionnant sous le système d'exploitation *Microsoft Windows vista* dont les performances sont les suivantes:

- Processeur Intel Core2 Duo T6400.
- Fréquence de 2.0 GHz.
- Mémoire RAM de 2 Go DDR2.
- Disque 250 Go SATA 5400 tours/mn.
- Carte graphique NVIDIA GeForce 9200M GS.

Langage de programmation

Aujourd'hui de nombreux logiciels de programmation permettent un développement aisé sous Windows. Notre choix a été porté sur le langage de programmation « **MATLAB R2010a 7.10.0.499 (Windows 32/64 bits)**».

MATLAB se présente sous un environnement interactif pour le développement d'algorithmes d'analyse des données. Comparé à des langages de programmation classiques (C / C + +, Java, Pascal, Fortran) MATLAB, permet de réduire le temps de calcul pour des tâches typiques et simplifie grandement le développement d'algorithmes complexes. MATLAB est le fondement de la famille de produits Math Works [21]; il permet de résoudre un large éventail de problèmes d'ordre scientifique tels que: le développement des systèmes de contrôle, la conception de systèmes de communication, le traitement du signal et de l'image, la modélisation financière, la biologie computationnelle etc...

MATLAB est un outil très efficace, largement utilisé pour le calcul numérique et la visualisation graphique. Dans cet environnement, les variables et les scalaires sont manipulés comme des matrices de "n" colonnes par "m" rangées. Par exemple, un scalaire serait une matrice de 1 x 1. À l'exécution, MATLAB affiche plusieurs fenêtres sur l'écran. Les trois types de fenêtres les plus importants sont :

- "Command window" (cf. Figure 4.1), où toutes les commandes sont intégrées ;

Figure 3. 1 fenêtre de commande

Une fenêtre appelée "Command Window" apparaît sur l'écran. L'utilisateur peut intégrer de multiples commandes ou équations mathématiques après le signe ">>" qui apparaît au côté gauche de la fenêtre.

Pour exécuter une opération, il faut toujours appuyer sur la touche "entrée" du clavier. De plus, il faut terminer l'opération par un point-virgule ";"sinon, toutes les étapes du calcul seront affichées sur l'écran.

- "Edit Windows", où l'utilisateur peut modifier ou créer des programmes MATLAB ("M-files") ;

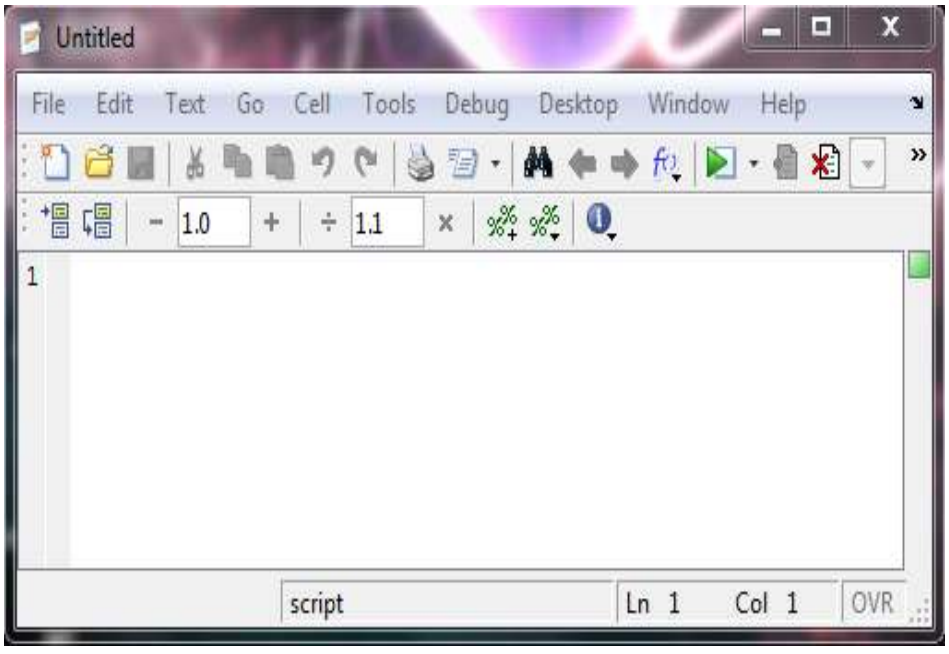

Figure 3. 2 fenêtre d'édition

Au lieu de taper les commandes individuellement et directement dans la fenêtre de commande, il est possible de créer un fichier appelé "m-file" qui contient toutes les fonctions et commandes nécessaires et qui, peut être rapidement exécuté en tapant le nom du fichier dans la fenêtre de commande. Ces fichiers sont appelés "script files" et se terminent avec l'extension ".m". La fenêtre "Edit Window" est utilisée pour créer ou modifier les "m-files". Pour créer un nouveau fichier, il faut aller dans le menu de sélection (À noter que les "m-files" sont exécutés en tapant le nom du fichier dans la fenêtre de commande).

On peut se rendre compte de la puissance du logiciel en lançant la commande *demo.* De plus, des **" toolboxes "** (boîtes à outils) sont disponibles dans de nombreux domaines (traitement du signal, traitement d'image, optimisation, contrôle ...). MATLAB s'élargit sur la librairie des fonctions mathématiques, l'environnement graphique, ainsi que sur une interface de développement.

Aperçu du logiciel réalisé

Le logiciel que nous avons implémenté est une mise en œuvre facile : pas de mot clés à connaître ni de programme à écrire, l'utilisation est constamment guide en cliquent sur les buttons selon notre choix.

Hiérarchie

Notre interface présente une structure arborescente qui offre à l'utilisateur un bon suivi des applications effectuée et une meilleure représentation de ses données. Toutes les applications sont utilisées automatiquement à la fin de chaque session. La figure 3.3 illustre l'organigramme du logiciel élaboré.

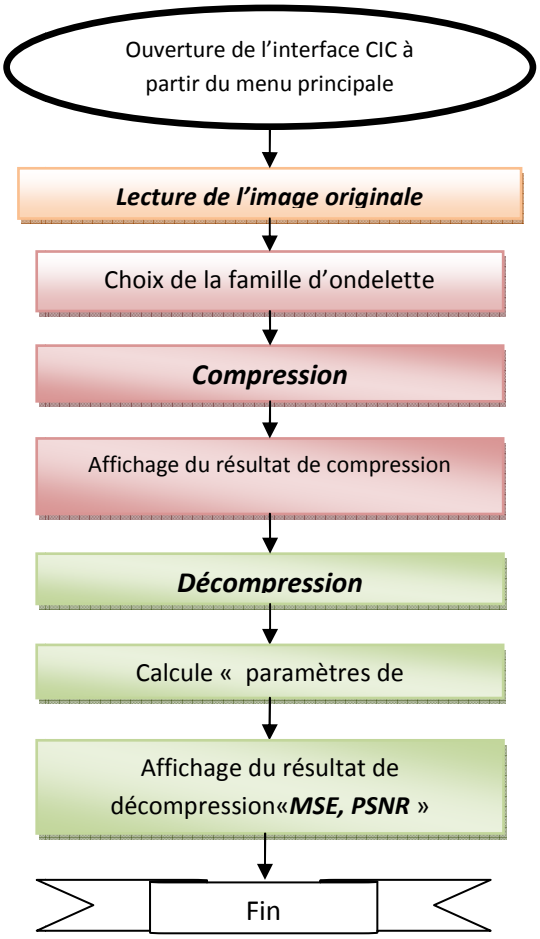

Figure 3. 3 Organigramme du logiciel élaboré

Le Lancement de l'application fait à partir de l'éditeur MATLAB avec la commande :

>>**MENU**, l'interface principale apparaît comme le montre la Figure 3.4

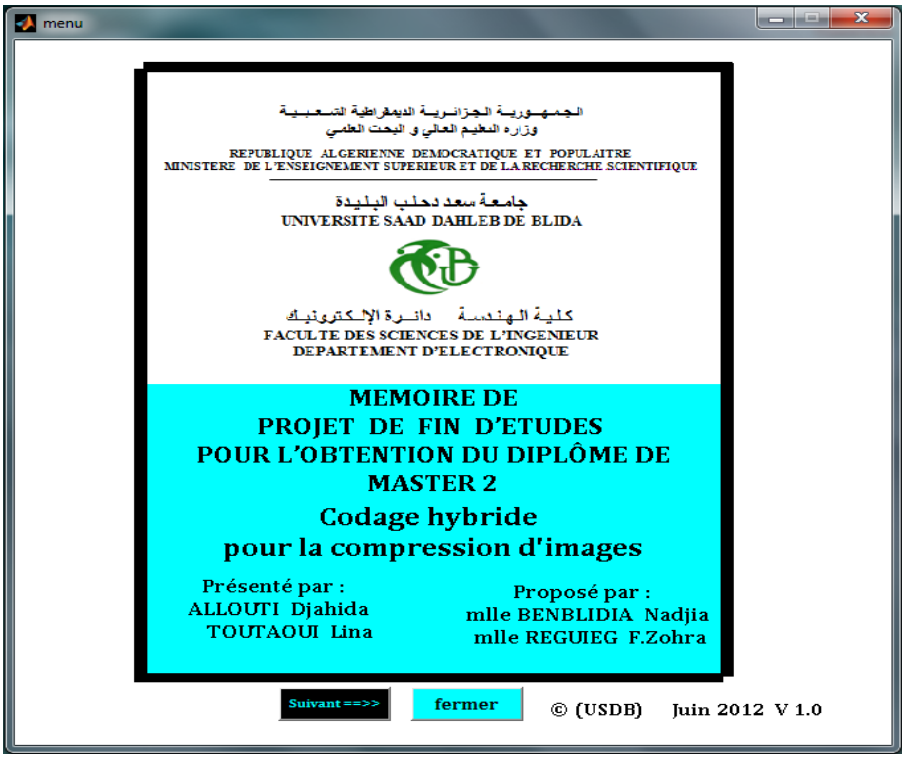

Figure 3. 4 menu principale

Il s'agit de décrire brièvement la première version de l'application 'CIC(0)'

Dès qu'on appuie sur le bouton ***« suivant »***, l'interface de notre application 'CIC(0)' apparaît comme le montre la figure IV.5.

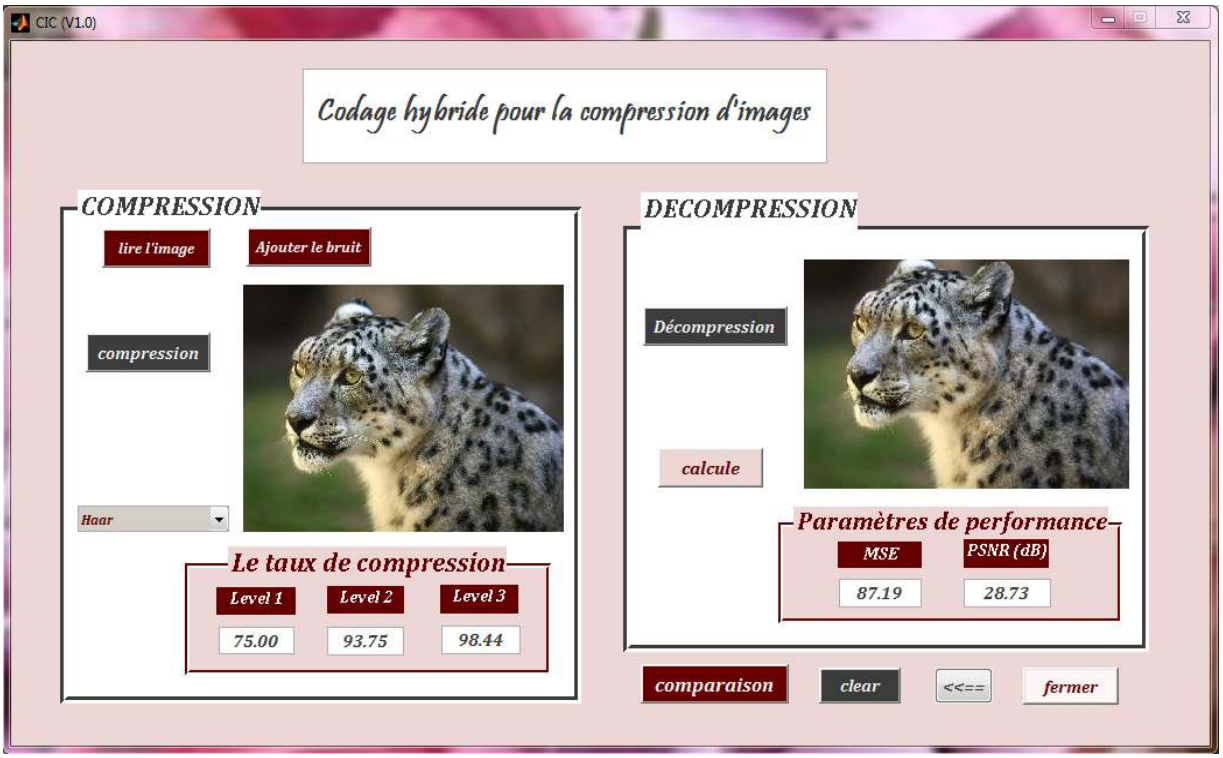

Figure 3. 5 Interface de l'application CIC(V1.0).

L'application 'CIC(V1.0)', développée sous environnement MATLAB, consacre la première partie à la compression des images suivant l'algorithme hybride basé sur les deux méthodes de transformée DCT et DWT. Elle exploite les résultats obtenus en calculant le taux de compression pour différent niveau de décomposition par ondelette, en deuxième partie nous faisant la décompression pour bien validé la qualité de l'image reconstruite on calcule les paramètres de distorsion à savoir le PSNR et MSE

Pour bien évaluer notre technique, on va la comparer avec d'autre méthodes de compression, Pour cela voici l'interface de comparaison, elle s'ouvre en appuyant sur le bouton « comparaison » dans l'interface « CIC »

La figure 3.6 représente cette interface

Figure 3. 6 Interface de comparaison

Dans ce qui suit on va bien détaillée chaque partie avec ces modules

Description des modules

- ***Module 'lecture'*** **:** Permet de charger une image à partir de n'importe qu'elle endroit du PC.
- ***Module 'Ajouter le bruit' :*** ce module permet d'ajouter 3 types de bruit, gaussien , salt , pepper et speckle à l'image originale
- ***Module 'Supprimer le Bruit' :*** permet de supprimer le bruit dans l'image
- ***Module du choix de la famille d'ondelette*** **:** ce module nous permet de choisir la famille de l'ondelette que l'on veut pour la compression
- ***Module 'compression'*** **:** ce module est le plus important dans notre travail ; il contient l'algorithme hybride « DWT, DCT », il permet de faire la compression de l'image originale en choisissant la famille d'ondelette
- ***Module d'affichage des résultats de compression*** **:** après la compression de l'image,on peut voir le taux de compression « Cr » pour chaque niveau de décomposition qui sera afficher dans trois case ,respectivement Cr level 1,Cr level 2 et Cr level 3.

- ***Module 'décompression' :*** il permet de décompresser l'image et afficher l'image reconstruite

- **Module du résultat de décompression :** ce module est très important pour tester les performances de notre algorithme en basant sur deux paramètres, l'erreur quadratique moyenne ***(Mean Square Error, MSE)*** et rapport crête signal sur bruit ***(Peak Signal to Noise Ratio, PSNR)***
- **Module de comparaison :** au niveau de ce module on fait la comparaison de notre aproche (CIC) avec d'autres techniques de compression existe dans le logiciel Matlab

Bibliothèque d'images

Les différentes images employées pour tester notre application de compression sont divisée en trois type d'image, Nous retrouvons ainsi, des images synthétiques, texturées et réelles (cf. Figures 3.7).

- L'image 'Lena' qui est un portrait de référence composé de diverses textures, dont des Textures très fines comme les yeux et les cheveux. Cette image de taille 500 x 500 pixels codés sur 256 niveaux de gris, est représentée sur la figure 3.7 (a).

(a) Image lena

(b) Image œil

(d) Image Léopard

(c) Image Perrocket

Figure 3. 7 les différentes images utilisées pour la compression/décompression

Tests expérimentaux

Nous présentons dans ce qui suit, les résultats issus de notre application, sur chacune des images abordées

Résultats de la compression

Les images de la collection étudiée, sont compressées, suivant l'algorithme hybride DWT-DCT .En modifiants deux paramètres, la famille de l'ondelette appliquer dans l'étape de transformation, et le niveau de décomposition des ondelettes

- les familles d'ondelette utilisée :
    - L'ondelette de Haar, L'ondelette de daubechies(3.5.7.9……..)

- les niveaux (levels) de décomposition sont :

    - Level 1, Level 2 et Level 3

Dans cette étape le paramètre d'évaluation de notre technique de compression est le taux de compression noté CR

Les résultats obtenus sont regroupée dans le tableau si dessous :

Nom de l'image	Famille d'ondelette	Taux de compression pour chaque Niveau de décomposition		
		Level 1	Level 2	Level 3
Lena Dimension (500X500) Taille de l'image originale 732 Kbytes	Haar	75.00	93.75	98.41
	db3	74.60	93.45	98.26
	Db5	74.19	93.14	98.04
	Db7	73.79	92.82	97.87
	Db9	73.37	92.49	97.63
Œil Dimension (512X512) Taille de l'image originale 768 Kbytes	Haar	75.00	93.75	98.44
	Db3	74.61	93.45	98.24
	Db5	74.21	93.15	98.08
	Db7	73.81	92.84	97.85
	Db9	73.41	92.52	97.68
Perocket Dimension (384X256) Taille de l'image originale 232 Kbytes	Haar	75.00	93.75	98.44
	Db3	74.34	93.25	98.10
	Db5	73.68	92.74	97.82
	Db7	73.01	92.20	97.42
	Db9	72.33	91.65	97.10

Tableau 3. 1 Résultats de la compression sur différentes images

Résultat de la décompression

Cette étape comme on a dit précédemment permet de reconstruire l'image originale avant la compression, pour bien évaluer la qualité de l'image reconstruite, deux paramètres seront discutées, à savoir *l'erreur quadratique moyenne* noté ***MSE*** et le *rapport crête signal sur bruit* noté ***PSNR.*** Ces deux valeurs sont aussi calculées pour

différentes familles d'ondelette. La figure 3.8 illustre la reconstruction de l'image originale en modifiant la famille d'ondelettes :

Image originale

Lena

Ondelettes de Haar

Cr=98.41, PSNR=29.28

Ondelettes Daubechies (db5)

CR=98.04, PSNR=31.48

Image originale

perroquet

Ondelettes Daubechies (db5)

CR=98.10%,PSNR=30.62dB

Image originale

Œil

Ondelettes Daubechies (db5)

CR= 97.68 ,PSNR= 36

Figure 3. 8 résultats de la reconstruction d'images pour différentes familles d'ondelettes

Le tableau ci-dessous englobe tous les résultats obtenus :

Nom de l'image	Famille d'ondelette	*Erreur quadratique moyenne « **MSE** »*	*rapport crête signal sur bruit « **PSNR** »*
Lena Dimension (500X500) Taille de l'image originale 732 Kbytes	Haar	76.75	29.28
	db3	48.99	31.23
	Db5	46.26	31.48
	Db7	45.12	31.59
	Db9	43.74	31.72
Œil Dimension (512X512) Taille de l'image originale 768 Kbytes	Haar	30.63	33.27
	Db3	17.26	35.76
	Db5	16.36	35.99
	Db7	16.10	36.06
	Db9	15.97	36.10
Perroquet Dimension (384X256) Taille de l'image originale 232 Kbytes	Haar	75.24	29.37
	Db3	56.32	30.62
	Db5	56.32	30.62
	Db7	52.41	30.94
	Db9	51.59	31.01

Tableau 3. 2 Résultats de la décompression sur différentes images.

Cas d'image bruitée

Pour évaluer notre algorithme de compression on a pensée à l'effet de bruit sur la reconstruction de l'image original. Ici on a pris l'image œil comme un exemple, on va ajouter le bruit gaussien en pourcentage, et on va voire la qualité de l'image reconstruite en calculer les paramètres de distorsion.

Image originale

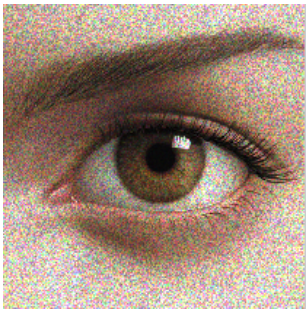

Image bruitée (Bruit speckle de 4%)

Image reconstruite MSE=794.07, PSNR=19.13 dB

Figure 3. 9 L'effet du bruit sur la reconstruction d'images

Pour plus de détaille sur la décompression d'une image bruité, voici les résultats affichée dans le tableau suivant

Nom de l'image	***Bruit en pourcentage***	*Erreur quadratique moyenne « **MSE** »*	*rapport crête signal sur bruit « **PSNR** »*
Lena Dimension (500X500) Taille de l'image originale 732 Kbytes	4%	571.47	20.56
	10%	1551.69	16.22
	30%	3170.07	13.12
Œil Dimension (512X512) Taille de l'image originale 768 Kbytes	5%	785.08	19.18
	10%	1418.03	16.61
	30%	3531.87	12.65

***Tableau 3. 3* Résultats de compression sur des images bruitées.**

Interprétation des résultats

- Nous remarquons que le critère du choix de familles d'ondelette joue un rôle crucial dans la compression et la décompression (reconstruction) des images,

l'ondelette de haar c'est elle qui nous a données un taux de compression plus élevée(98.44%) par rapport à l'ondelette de daubechies, tant dis que pour la décompression, on remarque que l'ondelette de Haar donne la plus grade valeurs de l'erreur quadratique moyenne ***MSE,*** donc l'image reconstruite est dégradé par rapport à l'image reconstruite en utilisant l'ondelette de daubechies ,plus précisément db9 qui a présenté le meilleur image reconstruite.

- Dans la compression, le niveau de décomposition choisi au niveau de l'étape de transformation aussi à un impact sur le taux de compression, pour notre algorithme on constate que le niveau (Level3) c'est lui qui nous a données le meilleur taux de compression pour toutes les images utilisée dans notre étude, mais il faut aussi savoir que en augmentant beaucoup plus le niveau de décomposition l'image reconstruite se dégrade de plus en plus.
- ***Cas d'une image avec bruit :***

  Pour la compression, l'ajoute du bruit dans l'image originale n'influe pas sur le taux de compression, l'influence du bruit est remarquable au niveau de la décompression, on remarque que l'erreur quadratique moyenne augmente avec l'augmentation du bruit, donc la valeur du PSNR diminue par rapport a une image sans bruit, l'image reconstruite est beaucoup dégradé.

  Pour compresser une image bruite avec notre algorithme il est nécessaire de faire un prétraitement de l'image avant de commencer à exécuter les étapes de l'algorithme hybride.

Comparaison de notre algorithme avec d'autres techniques de compression

Pour voir la robustesse de notre application de compression, on va comparer les résultats obtenus avec d'autres techniques de compression, en termes de terme de taux de compression et le rapport de crête signale bruit les méthodes choisi pour la comparaison sont les suivantes :

'gbl_mmc_h' : Global thresholding of coefficients and Huffman encoding

'ezw' : Embedded Zerotree Wavelet

(a) Image originale

(b) Reconstruite par 'gbl_mmc_h'
Cr= 82.91%, PSNR= 28.24dB

(c) Reconstruite par 'ezw'
Cr= 95.97%, PSNR=16.53 dB

(b) Reconstruite par Cr=97.63 %,
PSNR=31.72dB

Figure 3. 10 Comparaison de la décompression de l'image lena par notre approche avec 'gbl_mmc_h' et 'ezw'

(a)Image originale

(b) Reconstruite par
'gbl_mmc_h'
Cr=54.02 %, PSNR= 31.13d B

(c) Reconstruite par 'ezw'
Cr= 97.72 %, PSNR=19.86 dB

(b) Reconstruite par
Cr=97.68%, PSNR= 36.10dB

Figure 3. 11 Comparaison de la décompression de l'image Œil par notre approche avec 'gbl_mmc_h' et 'ezw'

Tableau de comparaison

Nom de l'image	***Global thresholding of coefficients and Huffman encoding*** ***'gbl_mmc_h'***		**Embedded Zerotree Wavelet** ***'ezw'***		***Notre approche*** ***CIC***	
	CR (%)	***PSNR (dB)***	**CR (%)**	**PSNR (dB)**	***CR (%)***	***PSNR (dB)***
Lena Dimension (500X500) Taille de l'image originale 732 Kbytes	82.91	28.24	95.97	16.53	97.63	31.72
Oeil Dimension (512X512) Taille de l'image originale 768 Kbytes	54.02	31.13	97.72	19.86	97.68	36.10

Tableau 3. 4 comparaison entre 'gbl_mmc_h' , 'ezw' et notre approche (image 'lena' et oeil)

Discussion

D'après les résultats présenter dans le tableau 3.4 on remarque bien la robustesse de notre technique de compression qui donne des meilleurs taux de compression avec une bonne reconstruction de l'image originale par rapport aux deux méthodes présenter dans le tableau qui se base uniquement sur la transformée en ondelette,

Donc l'amélioration vient d'être confirmée en ajoutant une transformée en cosinus discrète en termes de paramètres de distorsion ainsi le taux de compression.

Conclusion

L'algorithme de compression et de décompression des images que nous avons développé dans ce chapitre, est capable de compresser toute image avec de forts taux de compression et de la restituer avec une qualité appréciable. Ainsi, les tests effectués sur l'image Lena prise pour référence et sur des images visibles ont montré la supériorité de cet algorithme vis à vis d'autres méthodes de compression telles que celles basées sur la DCT, ou les Ondelettes.

De telles performances ont pu être atteintes grâce à la combinaison d'une transformée en ondelette DWT à 2D et de transformée en cosinus DCT que nous avons adoptée pour réaliser le processus de compression et de décompression des images, car L'utilisation de deux transformations a contribuer d'augmenter le nombre des cœfficients à haute fréquence, ce qui conduit à l'augmentation du taux de compression

Les critères de choix qui ont été utilisés dans notre algorithme dont le niveau de compression et la famille d'ondelette ont donnés un impact sur la qualité de l'image reconstruite.

Conclusion générale

Dans ce mémoire, nous avons élaboré une technique de compression d'images pour faciliter l'archivage d'images avec de forts taux de compression et le minimum de distorsions. Pour concevoir notre méthode, nous avons dans un premier temps Procédé à une comparaison des principales techniques de compression d'images publiées dans la Littérature. Nous avons ainsi montré que les méthodes avec pertes étaient les plus adaptées à la Compression des images et que celles basées sur les transformations linéaires donnaient le meilleur compromis possible entre le taux de compression et la qualité de restitution des images. Parmi ces techniques, celles utilisant les Ondelettes, la transformée discrète en Cosinus (DCT), semblent convenir à cette exigence.

Dans ce travail, un système d'algorithme hybride combine la DWT et la DCT dans la contrainte est d'atteindre un taux de compression élevé a été présenté.

Les résultats des simulations exhaustives montrent une certaine uniformité des performances améliorées pour le système hybride par rapport aux méthodes basées uniquement sur la DCT ou la DWT

Le nouveau système a également réduit les effets de faux contours et les artefacts de blocage significatif, ce qui se produit dans les images reconstruites en utilisant l'algorithme DCT

On a aussi vu que le niveau de décomposition dans la transformée en ondelette DWT à un rôle crucial sur le taux de compression, car on atteint un taux de compression élevée pour le troisième niveau de décomposition (Level 3) Différentes Types

d'Ondelettes sont utilisées, l'Ondelettes de Daubechies db9 donne le meilleur résultat par rapport aux autres familles d'Ondelettes.

Perspectives

- L'algorithme proposé peut aussi être implémenté sur d'autres espaces d'image couleur que l'espace RVB.
- Ce travail peut être approfondit pour explorer de nouveaux filtres que les Daubechies ou Haar.
- ça sera intéressant d'évaluer les résultats de l'algorithme en utilisant d'autres codeurs, par exemple codeur LZW a base du dictionnaire

Bibliographie

[1] Cziho A : ”Quantification vectorielle et compression d'image, application à l'imagerie médicale ”, Thèse de doctorat, école doctorale en informatique, traitement du signal et d'image, 1999.

[2] Lahdir Mourad : ”nouvelle approche de compression d'images basé sur les ondelettes et les fractales : application aux images météosat ”, diplôme de doctorat en électronique option : télédétection, université mouloud Mammeri, tizi-ouzou ,2010.

[3] Patrick Bas : ”Compression d'Images Fixes et de Séquences Vidéo ”, cours ENSERG/INPG, Laboratoire des Images et des Signaux de Grenoble ,2006.

[4] R. K. Rao ET P. Yip:” Discrete Cosine Transform: Algorithms”, Advantages and Applications. NY: Academic, 1992.

[5] W. B. Pennebaker et J. L. Mitchell:” *JPEG Still Image Data Compression Standard*”, 3rd ed. New York, Springer, 1993.

[6] G. Joy et Z. Xiang:*” Reducing false contours in quantized color images,” Computer and Graphics, Elsevier*, vol. 20, no. 2, pp. 231–242, 1996.

[7] Delgorge, C : ” *Proposition et Evaluation de techniques de compression d'images Ultrasonores dans le cadre d'une télé échographie robotisée*”. Thèse de doctorat, Sciences et Technologies industrielles université d'Orléans, 2005.

[8] Shannon, C. E:” A *mathematical theory of communication*”, Bell System Tech. J., vol. 27, p.379-423 et 623-656, 1948.

[9] L. Chen:” *VLSI Design of Wavelet Transform: Analysis, Architecture and Design Examples*”. Imp. College press, 2007.

[10] J. D. Kornblum:” *Using JPEG quantization tables to identify imagery processed by software*”, Digital Forensic Workshop, Elsevier, pp. 21–25, 2008.

[11] Ngono J.M : ”*Compression des images de Radar a synthèse d'ouverture dans le cadre de leur utilisation dans les systèmes d'information géographique*”, Thèse de doctorat, école Nationale Supérieure Polytechnique de Yaoundé de l'Université de Yaoundé I, cameroun, 172 p, 2001.

[12] Rodrigues, J. M : " *Transfert sécurisé d'images par combinaison de techniques de Compression, cryptage et marquage*", Thèse de doctorat, sciences et techniques du Languedoc, université Montpellier II, France, 143 p, 2006.

[13] Ammar, M : " *Optimisation d'un schéma de codage d'image a base d'une TCD. Application a un codeur JPEG pour l'enregistrement numérique à bas débit*", Thèse de doctorat, Ecole nationale supérieure des télécommunications, signal et images, Paris, France, 143 p, 2002.

[14] Ziv, J. et Lempel, A:" *A universal algorithm for sequential data compression*", IEEE Transactions on information theory, vol. 23, no. 3, p 337-343, 1977.

[15] Huffman, D. A:" *A method for the construction of minimum redundancy codes*", IRE Proc.,vol. 40, p. 1098-1101, 1952.

[16] R.J.Clarke:*" Digital Compression of Still Images and Video "*, Academic Press, 1995.

[17] D.J.Granrath:" *The Role of Human Visual Models in Image Processing*", Proc.of the IEEE, Vol. 69, pp. 552-561, 1981.

[18] U. S. Mohammed et W. M. Abd-elhafiez : " *Image coding scheme based on object extraction and hybrid transformation technique*", Int. J. of Engineering Science and Technology, vol. 2, no. 5, pp. 1375–1383, 2010

[19] Vincent Lecuire, Cristian Duran-Faundez et Nicolas Krommenacker:" *Energy-Efficient Transmission of Wavelet-Based Images in Wireless Sensor Networks*", Centre de Recherche en Automatique de Nancy (CRAN - UMR 7039), EURASIP Journal on Image and Video Processing, 2007.

[20] Suchitra Shrestha:" HYBRID DWT-DCT ALGORITHM FOR IMAGE AND VIDEO COMPRESSION APPLICATIONS" ,memoire pour l'obtention du diplôme en Master of Science, University Saskatchewan,canada, 2010.

[21] G. Sadashivappa et K. V. S. Ananda Babu : " Performance Analysis of Image Coding Using Wavelets", *IJCSNS International Journal of Computer Science and Network Security*, Vol. 8 No. 10, pp. 144-151,2008 .

[22] A. Al-Haj : " Combined DWT-DCT Digital Image Water- marking", *Journal of Computer Science*, Vol. 3, No. 9, pp. 740-746,2007 .

[23] K. Sayood:" Introduction to Data Compression" 2nd Edition, Academic Press, San Diego, 2000.

[24] R. C. Gonzalez et R. E. Woods:" Digital Image Proc-essing" Addison Wesley Publishing Company, Reading, 2001.

[25] M. Tsai et H. Hung : " DCT and DWT Based Image Watermarking Using Sub Sampling", *Proceeding of the Fourth International Conference on Machine Learning and Cybern*, Guangzhou, 18-21 August, pp. 5308- 5313,2005 .

[26] S. Esakkirajan, T. Veerakumar, V. Senthil Murugan et P. Navaneethan : " Image Compression Using Multiwavelet and Multi-Stage Vector Quantization",

International Jour-nal of Signal Processing, Vol. 4, No. 4, 2008. http://www.waset.org/journals/ijice/v4/v4-4-32.pdf.

[27] M. Antonini, M. Barlaud, P. Mathieu et I. Daubechies:" Image Coding Using Wavelet Transform", *IEEE Transactions on Image Processing*, Vol. 1, No. 2, pp. 205-220, 1992.

[28] I. E. G. Richardson:" video Codec Design", John Wiley & Sons, New York, 2002.

[29] T. Acharya et P. S. Tsai:" JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Ar-chitectures", John Wiley & Sons, New York, 2005.

[30] S. Ktata, K. Ouni et N. Ellouze : "A Novel Compression Algorithm for Electrocardiogram Signals Based on Wavelet Transform and SPIHT", *International Journal of Sig-nal Processing*, Vol. 5, No. 4, pp. 32-37,2009 .

[31] R. Janaki et A. Tamilarasi : " Visually Improved Image Compression by Using Embedded Zero-Tree Wavelet Coding ", *IJCSI International Journal of Computer Science Issues*, Vol. 8, No. 2, pp. 593-599,2011 .

[32] S. M. Basha et B. C. Jinaga : " A Robust Image Compression Algorithm Using JPEG 2000 Standard with Golomb Rice Coding," *IJCSNS International Journal of Computer Science and Network Security*, Vol. 10, No. 12, pp. 26-33,2010 .

[33] C. Christopoulos, J. Askelof and M. Larsson, "Efficient Methods for Encoding Regions of Interest in the Upcom-ing JPEG 2000 Still Image Coding Standard," *IEEE Signal Processing Letters*, Vol. 7, No. 9, pp. 247-249,2000 .

[34] M. M. Siddeq:" Image Restoration Using Perception Neural Network " , Master Thesis, Computer Science Department, University of Technology, Iraq, 2001.