Chapitre II

Techniques de compression

Introduction

De nos jours, la représentation numérique des images est devenue un problème majeur, les différentes applications, telles que la télécopie, la vidéoconférence, l'imagerie médicale et satellitaire, la télévision haute définition et la télésurveillance, exigent sans cesse de très grands espaces de stockage de l'information et un débit de transmission très élevé, ces deux contraintes restent toujours en suspens malgré le développement de la technologie. La compression est donc un moyen incontournable pour faire face à ces contraintes. C'est une étape essentielle pour optimiser l'utilisation de ces grands volumes d'information dans les réseaux informatiques. Ce chapitre explore les notions de base de compression d'images (but, mesure de performance, compression sans et avec perte, techniques du codage sans perte, etc.). Dans la dernière section, nous exposons notre méthode de travail qui repose sur des transformations réversibles (en ondelettes et couleurs) qui traitent des entiers et retournent des entiers.

Compression d'image

La compression d'image est une application de la compression des données sur des images numériques. Cette compression a pour utilité de réduire la redondance des données d'une image afin de pouvoir l'emmagasiner sans occuper beaucoup d'espace ou la transmettre rapidement. La compression d'image peut être effectuée avec perte de données ou sans perte [3].

But de la compression d’image

La compression est une réduction du nombre de bits nécessaires pour représenter les images. Compresser les images permet d'optimiser la capacité de stockage et la vitesse de transfert des fichiers [16].

Mesures de performance de la compression d’images

a. Taux de compression

Le taux de compression donne une mesure de performance des méthodes de compression des images. Etant donné que l’objectif d’une compression est de minimiser la quantité d’informations nécessaire à la représentation d’une image, le taux de compression représente le rapport entre le nombre de bits utilisés par l'image originale et le nombre de bits utilisés par l'image compressée [17].

$$\tau = \frac{\text{nombre de bit utilisés pour représenter l'image originale}}{\text{nombre de bit utilisés par l'image compressée}} \quad \text{(II.1)}$$

b. Débit

Le débit binaire, qui est une autre mesure souvent utilisée, est déterminé à partir du nombre moyen de bits par pixel (Bpp) de l'image codée[18], il est défini comme suit:

$$\text{débit} = \frac{\text{nombre de bits par pixel dans l'image originale}}{\text{taux}} \quad \text{(II.2)}$$

c. Temps de calcul

La contrainte du temps est un facteur essentiel dans l'évaluation des performances de toute méthode de compression, elle revient à calculer le temps pris par la compression et la décompression des images. Cette contrainte est plus au moins imposée selon l'application visée par la compression (transmission ou archivage). En effet, il serait dommage, dans une application de transmission, que le temps gagné par une réduction de la taille des données à transmettre soit inférieur au temps passé à la compression / décompression. Cette qualité sera cependant moins cruciale dans des applications visant l'archivage de données [19].

Compression des données

La compression des données peut être définie comme étant un système dont l'entrée est une donnée sans compression et la sortie est un flux de données numériques relativement court représentant la donnée compressée. Le processus inverse est appelé décompression permettant la reconstruction de l'image à partir du flux de données numériques. Parfois, les systèmes de compression et de décompression ensembles sont appelés "Codec" (codage pour compression et décodage pour décompression) [20].

Méthodes de compression

On peut distinguer deux grandes familles d'algorithmes de compression, les méthodes dites sans perte ou réversibles garantissent la restitution parfaite des images, alors que les méthodes dites avec perte ou irréversibles modifient plus ou moins la valeur des pixels [5].

Compression sans perte(ou réversible)

La compression est dite sans perte lorsqu'il n'y a aucune perte des données sur l'information

D'origine. Il y a autant d'information après la compression qu'avant. Le but est de réduire la taille des images obtenues après la compression tout en ayant la possibilité de retrouver exactement l'image d'origine[14].Elle est utilisée dans des applications comme l'archivage des images médicales, l'imagerie satellitaire (le coût des images est élevé et les détails sont importants), les textes, les programmes et tout autre type de données nécessitant une conservation intégrale [5].

Il existe de nombreux types d'algorithmes de compression d'images sans perte. Voici les plus répandus.

Méthodes statistique

a. Codage de Shannon

Utilisé dans les années cinquante, le code de Shannon-Fano est le premier code à avoir exploité la redondance d'une source. Tous les symboles à compresser sont triés selon leur probabilité, et l'ensemble trié des symboles est coupé en deux parties de telle façon que les probabilités des deux parties soient le plus proche possible de l'égalité (la probabilité d'une partie étant égale à la somme des probabilités des différents symboles de cette partie)[21].

Tous les symboles de la première partie sont codés par un "0" suivi de leur code de Shannon-Fano en ne prenant en compte que les symboles de la première partie, et tous les symboles de la seconde partie sont codés par un "1" suivi de leur code de Shannon-Fano en ne prenant en compte que les symboles de la seconde partie, récursivement. Lorsqu'une partie ne contient qu'un seul symbole, celui-ci est représenté par un code vide (de longueur nulle).L'approche du codage de Shannon-Fano est descendante: l'algorithme part de l'ensemble des symboles et divise cet ensemble récursivement jusqu'à arriver à des parties ne contenant qu'un seul symbole [22].

L'inconvénient de cette approche est que, lorsqu'il n'est pas possible de séparer un ensemble de symboles et deux sous-ensembles de probabilités à peu près égales (c'est-à-dire lorsque l'un des sous-ensembles est beaucoup plus probable que l'autre), les codes produits ne sont pas optimaux [22].

b. Codage de Huffman

Le codage Huffman a été proposé par David Huffman en 1952.C'est une méthode statistique basée sur l'attribution d'un mot de code binaire pour chaque symbole de la chaîne à compresser.

La longueur des mots de code des symboles est variable. Les symboles ayant la probabilité d'apparition forte sont codés avec des séquences de bits plus courtes, tandis que les symboles dont la probabilité d'apparition est faible sont codés par des séquences plus longues. Le codeur de Huffman est un arbre binaire ordonné par tous les symboles et par leurs fréquences d'apparition. Les deux symboles les moins fréquents de la source sont reliés parleurs 'Parents' en faisant la somme de leurs fréquences. Les symboles prennent alors les valeurs "0" et "1"[3]. Le processus est répété sur l'ensemble des symboles jusqu'à ce qu'il ne reste qu'un seul symbole en formant la racine de l'arbre binaire. L'opération inverse est utilisée pour le décodage[23].

- **Algorithme du codage**

1. Les symboles sont triés et classés en fonction de leur fréquence.
2. A partir des deux symboles 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.
3. Le nœud crée remplace désormais les deux symboles dans la suite du processus. Aces derniers sont affectés respectivement les chiffres binaires "0" à la branche de gauche et "1" à la branche de droite.
4. La même démarche est reprise en considérant les deux symboles ou nœuds de poids le plus faible. Elle est renouvelée tant qu'il reste plus d'un nœud libre.

   Pour obtenir le code binaire de chaque symbole, on remonte l'arbre à partir de la racine jusqu'aux feuilles en rajoutant à chaque fois au code "0" ou un "1" selon la branche suivie. Il est intéressant de noter qu'on changeant la procédure de tri, on peut obtenir différents codes de Huffman pour une même source. En général, on cherche à établir le code de Huffman à variance minimale. Pour obtenir ce dernier, on doit toujours placer la nouvelle combinaison de lettres le plus haut possible dans la liste triée. Par ailleurs, on distingue des variantes de l'algorithme de Huffman telle que: le codage statique le codage semi-adaptatif et le codage adaptatif [24].

Pour une source X d'entropie H(X). La longueur moyenne L d'un mot de code obtenu par le codage de Huffman vérifie la relation suivante [24]**:**

$$\mathbf{H}(\mathbf{X}) \leq \mathbf{L} < H(\mathbf{X}) + \mathbf{1} \qquad (\text{II. } 3)$$

- **Exemple du codage**

Une source émet des lettres d'un alphabet A= {a_1, a2, a_3, a_4} avec les probabilités P (a_1)=0.1, P (a_2)=0.3, P (a_3)=0.25 et P (a_4) =0.3.

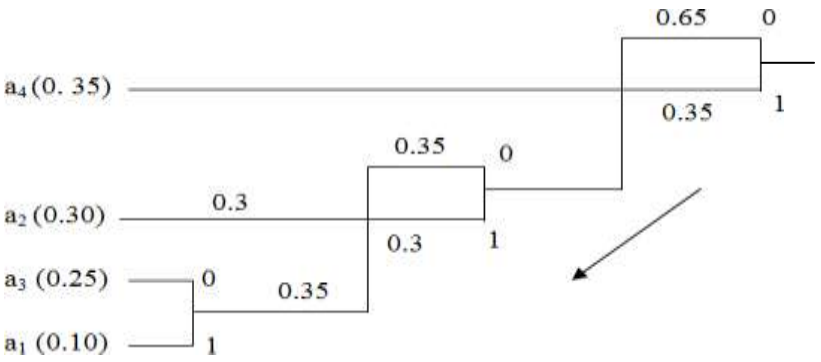


Figure II.1 : Algorithme de Huffman

Le code des symboles est: a_1= 001, a_3= 000, a2= 01, a_4= 1

- **Avantages et inconvénients du codage**

Le codage de Huffman offre une compression caractère par caractère optimale, Il a aussi l'avantage d'être facile à implanter par programmation, et le temps d'exécution est plutôt rapide. Toutefois, certaines méthodes de compression qui encode des mots plutôt que des caractères peuvent offrir un taux de compression plus intéressant. Ainsi, le codage de Huffman est souvent utilisé en conjonction avec d'autres techniques de compression comme par exemple: EZW [3].

c. Codage Arithmétique

Le codage arithmétique est un codage utilisant un modèle statistique, tout comme le codeur de Huffman [5]. Contrairement à ce dernier, il produit un code pour de symboles tout entière, et non pas un code par symbole. Chaque nouveau symbole lu modifie de façon incrémentale le code de sortie. Ce code de sortie est un nombre à virgule flottante compris entre 0 et 1, dont le nombre de chiffres après la virgule correspond au nombre de symboles. Contrairement à Huffman, il n'est pas obligatoire que chaque code ait un nombre entier de bits. Le codeur arithmétique est plus performant que le codeur de Huffman, mais il est plus complexe à implémenter [5].Décrivons brièvement ci-dessous l'algorithme de codage arithmétique dans le but d'en illustrer le principe, sachant que le décodage opère de manière inverse [3]:

- Calculer la probabilité associée à chaque symbole dans la chaîne à coder
- Associer à chaque symbole un sous intervalle proportionnel à sa probabilité, dans l'intervalle [0,1[(l'ordre de rangement des intervalles sera mémorisé car nécessaire au décodeur).
- Initialiser la limite inférieure de l'intervalle de travail à la valeur "0" et la limite supérieure à la valeur "1".

- Tant qu'il reste un symbole dans la chaîne à coder:
  - Largeur = limite supérieure - limite inférieure
  - Limite inférieure = limite inférieure + largeur x (limite basse du sous intervalle du symbole)
  - Limite supérieure = limite inférieure + largeur x (limite haute du sous intervalle du symbole).

A la fin, n'importe quelle valeur de l'intervalle final représente d'une manière unique la séquence d'entrée. Généralement, on choisit comme représentant de la séquence la limite inférieure de l'intervalle, ou bien on peut choisir la valeur moyenne. Il est à noter qu'à la fin du processus de codage, le codeur arithmétique génère un fichier composé d'un en-tête suivi de la représentation binaire du code choisi. De plus, l'en-tête peut contenir soit la table des probabilités, soit la table des fréquences de tous les symboles [1].

Le principal inconvénient de l'algorithme réside dans sa complexité d'implémentation.

Méthode par répétition

a. Codage RLE

Le principe de compression 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 pixels, tous juxtaposés [25]. Ainsi, au lieu de coder chaque pixel d'une image, le RLE regroupe les 36 valeurs voisines identiques et ne transmet une valeur qu'une seule fois, précédée par le nombre de répétitions.

Il est clair que cette approche fonctionne bien s'il y a beaucoup de répétitions dans l'image. Cet algorithme très simple, peut aboutir à des taux de compression plus élevés [1]. Ce type d'encodage est principalement utilisé par des formats d'images comme BMP ou PCX, ainsi que dans la numérisation d'images en noir et blanc [26]. Un exemple du codage RLE est illustré sur la figure II.2.

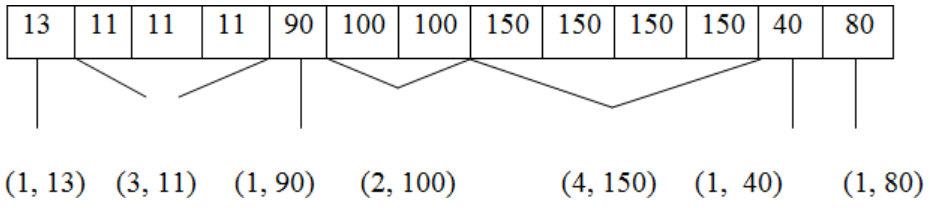


Figure II.2: Un exemple de codage par plage RLE

- **Avantages et inconvénients**

Le principal avantage de l'encodage RLE est son immense simplicité, son algorithme tient sur quelques lignes à peine autant au niveau de la compression que de la décompression. Il peut de plus être utilisé sur n'importe quel type de fichier ou après d'autres algorithmes décompression plus évolués pour gagner quelques octets de plus. En revanche, l'encodage RLE possède un désavantage de taille. Le fichier ou texte à compresser doit contenir plusieurs chaînes de caractères ou bits répétés pour être d'une quelconque utilité [26].

Méthode à base du dictionnaire

a. Codage (LZW)

Ce sont Abraham Lempel et Jakob Ziv qui ont inventé, en 1977, le premier algorithme de compression sous le nom de Lempel-Ziv-77 ou LZ77. LZ77 connut rapidement une nouvelle version :LZ78.

C'est une technique de codage à base d'un dictionnaire où nous mémorisons les chaînes qui se répètent dans ce dictionnaire. Ensuite, on remplace les chaînes mémorisées par leur adresse (ou indice) construite dans le dictionnaire [27]. L'élaboration du dictionnaire ainsi que la recherche de chaîne répétée sont différentes selon la version de l'algorithme.

Le premier algorithme de ce type a été mis au point par Lempel et Ziv en 1977 [28] et a donné lieu au programme LZ77. Celui-ci lit un flot de symboles et cherche des chaînes équivalentes dans une fenêtre de 4Ko précédant le flot d'entrée. Les équivalences sont remplacées par des codes. Les formats ZIP, ARJ et LHA basent leur compression sur cet algorithme [26].

Lempel et Ziv ont développé une nouvelle version de leur algorithme en 1978, LZ78, où le dictionnaire est construit à partir de tous les symboles précédemment rencontrés et non par une fenêtre, il est spécialisé dans la compression d'images et de fichiers binaires [26]. Puis en 1984, Terry Welch a modifié le format LZ78 pour pouvoir l'utiliser dans les contrôleurs de disques durs, le format LZW est né [26]. Le dictionnaire dans ce cas est initialement construit et contient l'ensemble des codes ASCII. Il est élaboré au fur et à mesure. Ce qui permet de changer la taille du dictionnaire au cours du codage [29]. Le codage LZW est une technique de compression réversible qui peut être appliquée à tout type de fichier de données, que ce soit: texte, image, fichier informatique, etc. Elle a été adoptée pour la mise en œuvre du format de compression d'images 'GIF' [30].L'Algorithme suivant montre son mécanisme :

Compression avec pertes (ou irréversible)

Ce type comporte une perte de données pendant le processus de codage/décodage. Le résultat qu'on peut en obtenir est une version altérée de l'image originale. Le but de ce type de compression est d'éliminer le plus d'information possible sans atténuer la qualité de l'image perçue par le système visuel humain [30].La compression avec pertes ne s'applique qu'aux données « perceptuelles », en général sonores ou visuelles il est impossible de retrouver les données d'origine après une telle compression. La compression avec perte est pour cela parfois appelée compression irréversible ou non conservatrice. La quantification est un des mécanismes utilisé dans les algorithmes de compression, qui produit des pertes d'information [31].

Méthodes spatiales et méthodes par transformation

Cette méthode s'intéresse au domaine dans lequel s'effectuent les opérations de base de la compression. Une image peut être représentée de deux façons strictement équivalentes:

- **Le domaine spatial**

Dans lequel l'image est représentée sous forme de pixels. C'est le domaine accessible visuellement à l'observateur [1].

- **Le domaine fréquentiel**

Dans lequel l'image est représentée sous forme de coefficients de fréquences. Le passage d'un domaine à l'autre se fait par des transformations mathématiques(en général linéaire) [1]

Dans le domaine spatial, l'information contenue dans l'image est distribuée sur toute la matrice image. Mais dans le domaine de fréquences, l'information est généralement plus concentrée.

a. Principe général d'un système de compression par transformée

La compression d'images par la méthode des transformées nécessite trois étapes de base, comme l'illustre la figure II.3. La première étape est la transformation des données de l'image pour obtenir des données moins corrélées. La deuxième étape est la quantification ou le seuillage, c'est dans cette étape que la perte d'information se produit.

Finalement le codage, les données quantifiées sont réduites par codage pour fin de transmission ou d'archivage. Dans la phase de décompression ou de reconstruction de l'image, trois étapes sont nécessaires. Les données transmises ou archivées sont décodées. La quantification inverse est appliquée au résultat du décodage. Puis, la transformation inverse est appliquée aux données du bloc [32].

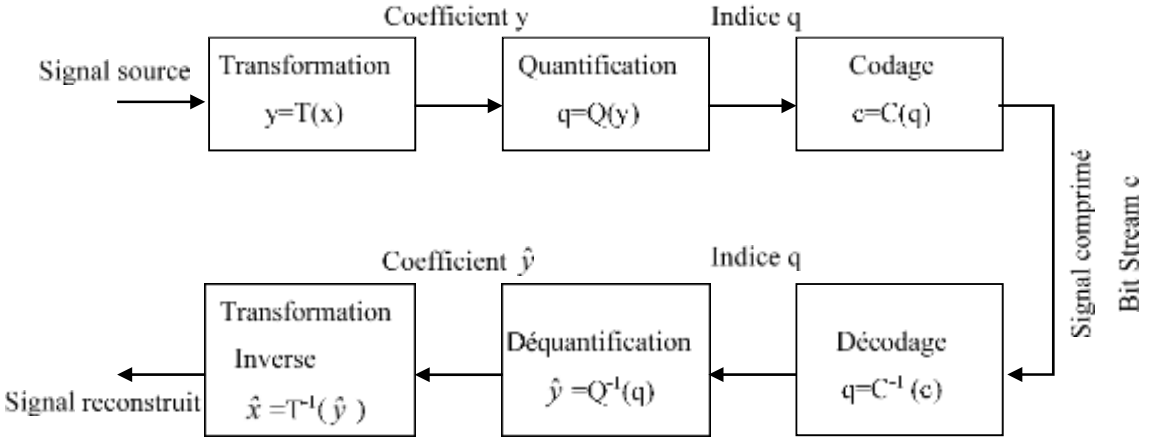


Figure II.3 Schéma typique de la compression/décompression par transformation (codec)

- **Transformation**

Dans ces méthodes, l'image de dimension NxN est subdivisée en sous images ou blocs détaille réduite. Chaque bloc subit une transformation mathématique orthogonale inversible linéaire du domaine spatial vers le domaine fréquentiel, indépendamment des autres blocs (transformée en un ensemble de coefficients plus ou moins indépendants).Les coefficients obtenus sont alors quantifiés et codés en vue de leur transmission ou de leur stockage. Pour retrouver l'intensité des pixels initiaux, on applique sur ces coefficients la transformation inverse [5]. L'objectif de ces transformations est double [5]: il s'agit de:

- ✓ Décorréler les données, c'est-à-dire d'obtenir des coefficients transformés moins corrélés que les pixels de l'image [5].
- ✓ Concentrer l'énergie sur un nombre réduit de coefficient, les coefficients ayant une valeur plus importante aux basses fréquences qu'aux hautes fréquences [5].

Dans ce but, plusieurs transformations linéaires ont été proposées dans la littérature scientifique [6]:

- Transformation de Fourier discrète (TFD).
- Transformation en cosinus discrète (TCD).

- **Quantification**

En général, la quantification apparaît en deuxième lieu dans un processus de compression, pour réduire la quantité d'information, et dégrade le signal de manière souvent irréversible [33]. Ainsi, une quantification est utilisée pour simplifier la représentation, tout en préservant l'information la plus pertinente. On remplace ainsi les valeurs initiales par un ensemble fini d'éléments qui donneront des résultats acceptables lors de la phase de décompression [5].

- **Codage**

Le codage est utilisé dans une chaîne de compression sans en général après l'étape de transformation. Cette étape cherche à éliminer toute redondance de telle façon que la quantité d'information à transmettre sois minimale, et permet de générer un message binaire (Bit Stream ou flot binaire) à partir duquel le décodeur sera en mesure de reconstruire soit une version approchée de l'image originale s'il s'agit d'une compression avec perte ou l'image originale s'il s'agit d'une compression sans perte.

b. Les transformées discrètes

Les méthodes par transformées figurent parmi les techniques de compression les plus employées. Elles n'agissent pas directement sur l'image numérique dans sa représentation canonique, mais sur le domaine de sa transformée. Elles permettent d'obtenir des taux de compression élevés tout en conservant une bonne qualité d'image.

Malgré qu'il existe un nombre considérable de transformées discrètes dans la littérature, on trouve que les transformées les plus utilisées dans le domaine de compression d'images sont la DCT et les DWT (Discret Wavelet Transform). Les ondelettes sont considérées dans le standard de compression JPEG 2000 [34], et la DCT dans les standards JPEG [35] et MPEG [36] (Moving Picture Experts Group) pour la compression des images fixes et animées, respectivement.

La popularité remarquable de la DCT est due essentiellement à sa capacité de compactage d'énergie (presque optimale) et à l'existence des algorithmes pour son calcul rapide et efficace.

a. Transformée de Fourier

Elle permet simplement de passer du domaine spatial au domaine fréquentiel [5]. Cette transformée rend donc visible les composantes en fréquence de l'image Il est intéressant de noter que nous pouvons revenir au domaine spatial via la transformée de fourrier inverse [5]. Une autre façon d'effectuer ce calcul permet de limiter considérablement la durée de cette transformation. C'est ce que l'on appelle la FFT (Fast Fourier Transform) [5].

b. Transformation en cosinus discrète DCT

La transformée en cosinus discrète est une transformation mathématique complexe qui a pour but de transformer le domaine de représentation de nos données, c'est-à-dire de passer d'un domaine spatial à un domaine fréquentiel. Notre œil est moins sensible à certaines données de l'image, le but de la transformée en cosinus discrète est de passer dans un domaine fréquentiel, ce qui nous permettra de trier efficacement l'ensemble des données de l'image et ainsi supprimer certaines données où l'œil humain ne verra que très peu de différences, ce qui revient à supprimer les hautes fréquences de l'image tout en gardant les données majeures qui sont représentées par les basses fréquences [7].

Durant la dernière décennie, la DCT (Discrète Cosine Transform) est émergée comme transformation d'image dans la plupart des systèmes visuels. La DCT a été largement déployé par les normes visuelles modernes de codage, par exemple, le MPEG [7].

Comme d'autre transformation, la DCT tente de dé corréler les données de l'image. Après la décorrélation chaque coefficient peut être codé indépendamment, sans perdre l'efficacité de la compression [7].

L'équation de la DCT est donnée comme suit :

$$DCT(i,j) = \frac{1}{2\sqrt{2N}} C(i)\, C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} pixel(x,y) \cos\left(\frac{(2x+1)\, i\pi}{2N}\right) \cos\left(\frac{(2y+1)\, j\pi}{2N}\right) \qquad (II.3)$$

- N : la largeur d'un bloc, ici N=8.
- i,j : les indices d'un coefficient de la DCT dans un bloc.
- x, y : les indices d'un pixel de l'image dans un bloc.
- DCT(i,j) : la valeur d'un coefficient dans un bloc.
- $C(x) = \frac{1}{\sqrt{2N}}$ si $x = 0$; $C(x) = 1$ si $x \neq 0$.

DCT (i, j) le coefficient repéré par la ligne i et la colonne j dans la matrice DCT [5].

La DCT est effectuée sur une matrice carrée NxN de valeurs de pixels en donnant une matrice carrée NxN de coefficients de fréquence. Le temps de calcul requis pour chaque élément dans la DCT dépend de la taille de la matrice [5].

❖ **Différence entre la FFT et la DCT**

- La DCT est beaucoup plus simple en termes de coup de programmation [5].
- La DCT est efficace dans la compression de multimédia (cas de JPEG) [5].
- Beaucoup plus utilisée [5].

c. La compression par ondelettes

Les ondelettes c'est d'abord une théorie mathématique d'analyse du signal, développée dans les années 90. On peut considérer qu'il s'agit d'une extension de l'analyse de Fourier [37]. La technologie de compression à base d'ondelettes offre une plus grande finesse au niveau de l'analyse du signal, et permet de mieux s'adapter aux propriétés locales de l'image. La transformation par ondelettes est une technique de compression d'image fixe très performante [38].

Passons maintenant à l'algorithme pyramidal utilisé. La décomposition en coefficients d'ondelettes n'utilise pas une fonction de moyenne, mais s'appuie sur deux filtres. Un filtre passe bas (H) et un filtre passe haut (L). La combinaison de ces filtres permet d'obtenir quatre sous images HH, HL, LH et LL, comme le montre le figure II.4. Ces filtres sont nommés miroirs en quadratures [5].

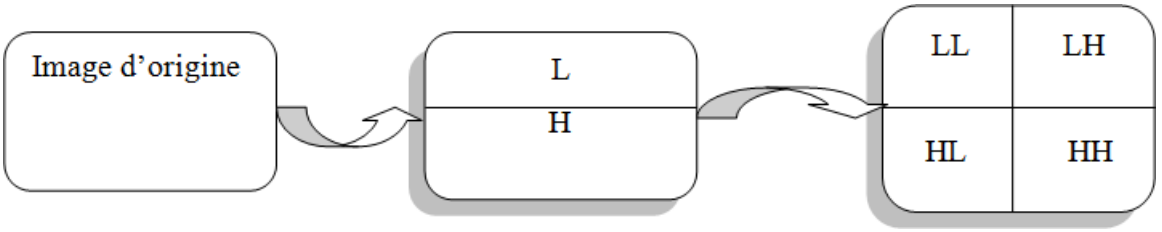


Figure II.4 : Transformation en colonnes et en lignes.

Chacune des quatre images obtenues par la transformation représente des informations bien distinctes.

Les étapes de compression par ondelettes sont usuellement :

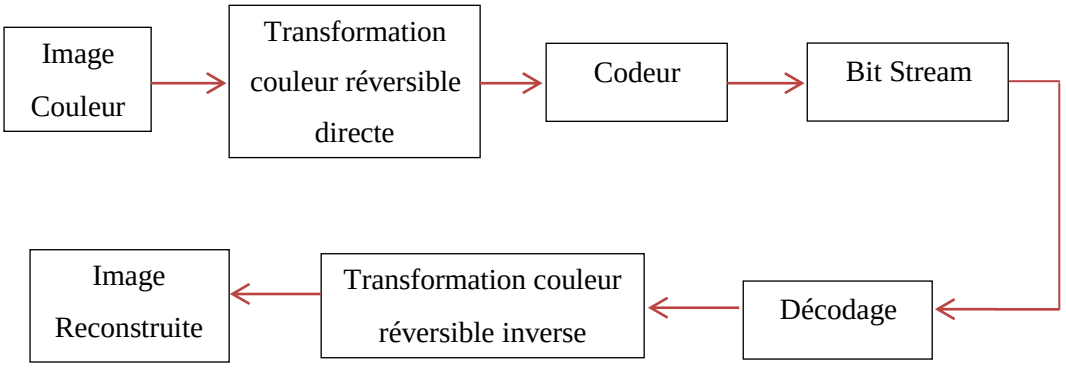
- Transformation par ondelettes.
- Quantification : les valeurs des coefficients de détails inférieurs à un certain niveau sont éliminées, en fonction de l'efficacité recherchée. C'est cette étape qui introduit des pertes.
- Codage des restantes : les données restantes sont transmises à un encodeur entropie, c'est à dire à un algorithme de compression de données (LZW, HUFFMAN, RLE, …) [1].

Présentation et application de notre méthode

Dans cette section, nous présentons notre approche de compression et décompression d'images. Les figures si dessous résument la structure de ces algorithmes ainsi que les étapes à suivre pour l'implémentation de nos codeurs (Huffman et Arithmétique) dans deux domaines différents: spatial et transformé [39], en appliquant un changement d'espace de couleurs RGB vers d'autres espaces, tels que Y'I'Q' et $O_1O_2O_3$. Ceci est dans l'optique de trouver l'algorithme de compression le plus performant et l'espace de couleurs le plus adapté à la compression des images couleur.

Codage appliqué dans le domaine spatial

Le schéma de compression/décompression sans perte des images couleur dans le domaine spatial est présenté dans la figure suivante :



FigureII.5 : Schéma de compression/décompression sans perte dans le domaine spatial.

a. Codage de Huffman

L'implémentation de notre algorithme de compression et décompression pour le codage de Huffman s'énonce comme suit :

Algorithme de compression:

Etape 1: lire l'image à compresser.

Etape 2: appliquer la transformation couleur réversible directe.

Etape 3: transformer les composantes de l'image couleur en vecteurs.

Etape 4: calculer l'Alphabet, les fréquences et les probabilités.

Etape 5: appliquer le codage de Huffman à chaque composante de l'image couleur.

Algorithme de décompression:

Etape 1: appliquer le décodage de Huffman à chaque composante de l'image couleur.

Etape 2: transformer les vecteurs en matrices.

Etape 3: appliquer la transformé couleur réversible inverse.

Etape 4: calculer l'entropie et le débit en Bpp (Bits par pixel).

Etape 5: afficher la taille de l'image originale et de l'image compressée en Bits.

Etape 6: afficher l'entropie et le débit.

b. Codage Arithmétique

L'algorithme faisant appel au codage Arithmétique se résume comme suit :

Algorithme de compression

Etape 1: lire l'image à compresser.

Etape 2: appliquer la transformation couleur réversible directe.

Etape 3: transformer les composantes de l'image couleur en vecteurs.

Etape 4: calculer l'Alphabet, les fréquences et les probabilités.

Etape 5: appliquer le codage Arithmétique à chaque composante de l'image couleur.

Algorithme de décompression

Etape 1: appliquer le décodage Arithmétique à chaque composante de l'image couleur.

Etape 2: transformer les vecteurs en matrices.

Etape 3: appliquer la transformé couleur réversible inverse.

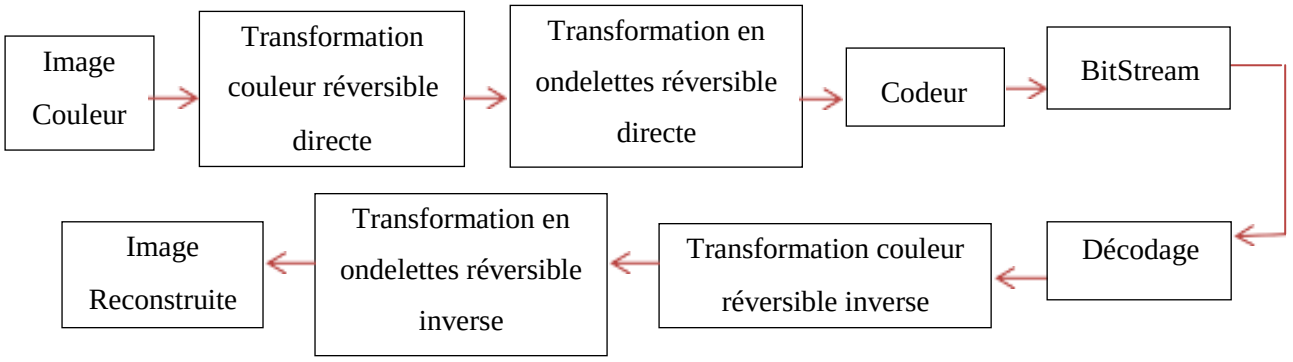
Etape 4: calculer l'entropie et le débit en Bpp (Bits par pixel).

Etape 5: afficher la taille de l'image originale et de l'image compressée en Bits.

Etape 6: afficher l'entropie et le débit.

Codage appliqué dans le domaine transformé

Le schéma de compression/décompression sans perte des images couleur dans le domaine transformé est illustré dans la figure suivante :



FigureII.6 : Schéma de compression/décompression sans perte dans le domaine transformé.

a. Codage de Huffman avec la transformée en ondelettes réversible (TOR)

L'algorithme du codage de Huffman basée sur une étape de transformation réversible s'établit comme suit :

Algorithme de compression

Etape 1: lire l'image à compresser.

Etape 2: appliquer la transformation couleur réversible directe.

Etape 3: appliquer la transformation en ondelettes réversible directe.

Etape 4: transformer les composantes de l'image couleur en vecteurs.

Etape 5: calculer l'Alphabet, les fréquences et les probabilités.

Etape 6: appliquer le codage de Huffman à chaque composante de l'image couleur.

Algorithme de décompression

Etape 1: appliquer le décodage de Huffman à chaque composante de l'image couleur.

Etape 2: transformer les vecteurs en matrices.

Etape 3: appliquer la transformé en ondelettes réversible inverse.

Etape 4: appliquer la transformé couleur réversible inverse.

Etape 5: calculer l'entropie et le débit en Bpp (Bits par pixel).

Etape 6: afficher la taille de l'image originale et de l'image compressée en Bits.

Etape 7: afficher l'entropie et le débit.

b. Codage Arithmétique avec la transformée en ondelettes réversible(TOR)

L'algorithme du codage Arithmétique basée sur une étape de transformation réversible s'établit comme suit :

Algorithme de compression

Etape 1: lire l'image à compresser.

Etape 2: appliquer la transformation couleur réversible directe.

Etape 3: appliquer la transformation en ondelettes réversible directe.

Etape 4: transformer les composantes de l'image couleur en vecteurs.

Etape 5: calculer l'Alphabet, les fréquences et les probabilités.

Etape 6: appliquer le codage Arithmétique à chaque composante de l'image couleur.

Algorithme de décompression

Etape 1: appliquer le décodage Arithmétique à chaque composante de l'image couleur.

Etape 2: transformer les vecteurs en matrices.

Etape 3: appliquer la transformé en ondelettes réversible inverse.

Etape 4: appliquer la transformé couleur réversible inverse.

Etape 5: calculer l'entropie et le débit en Bpp (Bits par pixel).

Etape 6: afficher la taille de l'image originale et de l'image compressée en Bits.

Etape 7: afficher l'entropie et le débit.

Conclusion

La compression des données est appelée à prendre un rôle encore plus important en raison du développement des réseaux et du multimédia. Nous avons abordé dans ce chapitre les concepts généraux de compression d'images fixes, les étapes principaux des algorithmes de compression avec et sans perte avec les différents méthodes existantes (Huffman, LZW, Arithmétique, DCT, etc.). Le chapitre s'est terminé par la présentation de notre méthode de travail qui est basée sur l'implémentation des deux algorithmes (Huffman et Arithmétique) dans le domaine spatial et transformé en se basant sur des transformations réversibles à coefficients entiers (en ondelettes et couleur).