République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Universitié Mohammed Seddik Benyahia – Jijel

Faculté des Sciences et de la technologie

Département d'Electronique

Filière : Télécommunications

Option : Systèmes des télécommunications

Mémoire de fin d'études

En vue de l'obtention du diplôme **: Master en Systèmes des télécommunications**

Thème

Compression d'images sans perte par des techniques du codage source

Présenté par:

BOUDJIT Yassamine

Encadré par:

Dr. BRAHIMI Tahar

Années Universitaire : 2018-2019

Remerciements

Je tiens avant tout à remercier Dieu le tout puissant de m'avoir donné le courage et la Volonté pour bien mener ce modeste travail.

En m'exprimant de la sorte, Je pense tout particulièrement À mon Encadreur,
Mr Brahimi Tahar, d'avoir dirigé avec patience et bon humeur ce modeste travail et qui, tout au long de celui-ci, m'a guidé par ses conseils pertinents et judicieux, qui nous a offert sa confiance et sa grande patience, veuillez trouver ici le témoignage de notre profondes reconnaissances.

Je tiens à exprimer ma profonde gratitude et à remercier également Dr K.ZERAOULIA Je lui en suis reconnaissant pour son aide précieuse.

Je remerciements vont aussi aux membres du Jury Pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leur proposition.

Enfin mes plus vifs remerciements vont à notre parent pour leurs soutiens et pour nous Avoir toujours encouragés dans ces études en nous offrant toutes les opportunités Possibles.

Dédicace

J'ai l'honneur de dédie ce modeste travail à mes chers parents Mohamed chérif et Samira et Ma chère tante Hakima qui m'avez dirigé et suivi pondant toute mes années d'étude, je ne pourrai jamais oublier d'exprimer ma profonde gratitude à:

Toute la famille Boudjit

Mes sœurs : Amina & Ahlem Pour leur soutien moral et leurs sacrifices le long de ma formation.

Mes chers frères : Abdeslam & Youcef

Tous mes cousins et mes cousines surtout (Ziyad, aya, Soumia, Abdallah …….)

Toutes mes tantes et mes oncles.

Je dédie ce travail à mes amies

A tous ceux qui me sont chères

A tous ceux qui m'aiment

A tous ceux que j'aime.

A Tous mes professeurs

Je dédie ce travail

Sommaire

Sommaire

Chapitre I

Notions générales sur la compression de données

Chapitre II

Outils et techniques du codage source avec application en compression d'images sans pertes

Chapitre III

Résultats et discussion

Liste des figures

Liste des Tableaux

Liste des abréviations

Françaises

2D- DCT transformée en cosinus discrète bidimensionnelle.

AC Codage par plage des coefficients

Bpp Bit par pixel

DC Codage différentiel du coefficient

TR Transformée réversible.

Anglos Saxon

BIP Band Inter leaved Pixel

BSQ Binary Stream Sequential

BIL Band Inter leaved Line

DCT Discret Cosine Transform

DWT Disctret Wavelet Transform

EQM l'erreur quadratique moyenne

GIF Graphic Interchange Format

JPEG 2000 Joint Photographic Experts Group 2000

JPEG Joint Photographic Experts Group

KLT transformée karhunen-loeve

LZW	Lempel-Ziv-Welch
LZ	Lempel-Ziv
MSE	Mean Square Error
PSNR	Peak Signal to Noise Ratio
RLE	Run Length Coding
RVB	Rouge Vert Bleu
SNR	signal to noise ratio
TIFF	Tagged Image File Format
PNG	Portable Network Graphic

Introduction Générale

Introduction Générale

La compression des données est l'une des technologies habilitantes de la révolution multimédia. Elle consiste à convertir des fichiers de données en fichiers plus petits pour un stockage et une transmission efficaces. Par définition, la compression est le processus de représentation des informations sous une forme compacte. Ce processus traite les informations sous forme numérique, c'est-à-dire, sous forme de nombres binaires représentés par des octets de données. Ces données contiennent de très grands ensembles de données.

Les techniques de compression sans perte n'entraînent aucune perte d'informations. Si les données ont été compressées sans perte, les données d'origine peuvent être récupérées exactement à partir des données compressées. Cette compression sans perte consiste à reconstruire les données d'origine à partir du fichier compressé sans aucune perte de données.

Ainsi, les informations ne changent pas pendant les processus de compression et de décompression. Ces types d'algorithmes de compression s'appellent des compressions réversibles car le message d'origine est reconstruit par le processus de décompression. Ces techniques de compression sans perte permettent de compresser des images médicales, du texte et des images **[1]**. Le codage est une étape essentielle dans la compression. Donc, la compression et le codage réduisent le nombre de bits par pixel à stocker ou à transmettre, en exploitant la redondance informationnelle dans l'image **[2]**.

Notre travail de projet de fin d'études s'inscrit dans le cadre de la compression de données sans perte. Il a pour but de mener une étude comparative entre deux techniques de codage source, couramment utilisées en compression de données. Il s'agit d'un travail d'évaluation et d'analyse comparative des performances de compression sans perte, des deux techniques sur un ensemble d'images de test en niveaux de gris.

Nous présentons quatre variantes envisageables des deux techniques : Huffman (variante H1) et Arithmétique (variante A1) pour leurs applications en compression d'images sans perte. Les variantes H1 et A1 s'appliquent directement sur l'image originale, alors

que les autres variantes H2 et A2 s'appliquent sur l'image transformée. Pour pouvoir effectuer une compression sans perte, il est nécessaire que le système de compression comporte une transformation réversible à coefficients entiers. En effet, l'étape de transformation est réalisée à l'aide d'une transformée en ondelettes réversible qui traite des entiers et retourne des entiers. Une comparaison des résultats obtenus avec la même base de données est ensuite effectuée en termes de débit (en Bpp) et de complexité de calcul (mesure de temps de codage/décodage).

Ce mémoire de fin d'études est organisé en trois chapitres :

Le chapitre I comporte une brève description des principaux outils essentiels à la compression de données. Il présentera le schéma élaboré pour effectuer de la compression par transformation, en décrivant chacun de ces modules. Des notions de base en compression d'images sont succinctement abordées.

Le Chapitre II est consacré à la présentation des principales techniques de codage source qui permettent une compression réversible sans perte. Ce chapitre se termine par la présentation de quatre variantes des deux techniques de codage: Huffman et Arithmétique aux fins de leurs applications à la compression d'images sans perte.

Le chapitre III présente les résultats obtenus, suivi par une analyse comparative et une discussion qui se base sur des critères objectifs. Nous terminons ce mémoire avec une conclusion générale.

Chapitre I

Chapitre I

Notions générales sur la compression de données

I.1. Introduction

Ce chapitre traite des concepts et des outils liés aux domaines de la compression d'images et de données. Nous discutons ensuite de la classification de compression en fonction de la perte ou non de l'information. Nous décrivons brièvement les transformations de compression couramment utilisées ainsi que les méthodes de compression souvent appliquées. De plus, nous présentons les outils utilisables pour mesurer la performance.

I.2. Définition et intérêt de la compression

La compression de données est un processus qui consiste à réduire le volume de données occupé par un fichier ou un ensemble de fichiers en le ou les transformant par un procédé

logique plus ou moins complexe dit algorithme. Ces algorithmes sont basés sur trois critères : le taux de compression, la qualité de compression, et *la vitesse de compression et de décompression.*

Un compresseur utilise un algorithme qui sert à optimiser les données en fonction du type de données à compresser ; un décompresseur est donc nécessaire pour reconstruire les données grâce à l'algorithme dual de celui utilisé pour la compression.

La méthode de compression dépend du type de données à compresser car une image ou un fichier audio ne représentent pas le même type de données.

- ***Exemples***

Pour une séquence vidéo dont les caractéristiques suivantes :

- 25 images par seconde ;
- 16 millions de couleurs (soit 3 octets par pixel) ;
- Résolution de 640 x 480.

Sans compression, il faudrait un débit de 23 Mo/s (25 x 3 x 640 x 480), et pour donner un ordre d'idée, cela représente un débit 130 fois plus important que celui d'un lecteur de CD-ROM simple vitesse (150 Ko/s). De plus, si on souhaite stocker 2 heures de vidéo, il nous faudrait une unité de stockage de 162 Go (équivalent à 34 DVD de 4,7 Go).

I.3. Notions sur l'image

Une image est une fonction qui associe à chaque composant une valeur précise.

Fonction :

$$\boldsymbol{R^2 \rightarrow R^n, (x, y) \rightarrow I(x, y)}$$

Où : n : est le nombre de composantes de l'image (n≥1).

I.3.1. Définition d'une image

Une image en informatique est appelée une image numérique et qui désigne toute image *acquise, créée, traitée* et *stockée* sous forme binaire.

I.3.2. Définition d'un pixel

Une image numérique est constituée d'un ensemble de points appelés pixels (abréviation de Picture Elément en anglais). Donc, un pixel représente le plus petit élément constitutif, adressable et contrôlable d'une image numérique.

Chaque pixel est représenté sous forme d'un petit carré, et il est composé de trois points de couleurs différentes (le rouge, le vert et le bleu). Ils ont l'apparence de petites lampes capables de composer jusqu'à 16 millions de couleurs **[3]**.

I.3.3. Taille des données d'une image

La taille des données d'une image est mesurée par le nombre total de pixel que comporte l'image multiplié par le nombre de bit par pixel et cela dépendra des dimensions de l'image (nombre de colonne, nombre de ligne), donc on peut écrire la relation suivante **[4]**:

$$\boldsymbol{Taille = C \times L \times M}$$

Où:

C: est le nombre de colonnes.

L: est le nombre de lignes.

M: est le nombre de bits par pixel.

I.3.4. Représentation de l'image

Il existe 6 types d'image: représentation vectorielle, représentation matricielle, N/B (Noir et Blanc), niveau de gris, couleur et multi-bandes.

I.3.4.1. Représentation vectorielle

C'est une image numérique composée d'objets géométriques individuels définis chacun par divers attributs de forme, de position, de couleur, etc.

I.3.4.2. Représentation matricielle

C'est une image numérique constituée de points juxtaposés telle que chaque point (pixel), porte des informations, positions et des couleurs. Une suite de pixels en ligne et en colonne forme une image matricielle.

I.3.4.3. Les images en Noir et Blanc

Ce sont des images dont chaque pixel est codé sur un seul bit et donc il peut contenir que des valeurs binaires soit 0 ou 1.

I.3.4.4. Les images en niveau de gris

Ce sont des images dont chaque pixel représente une valeur qui varie de 0, qui représente le noir, jusqu'à 255 qui représente le blanc.

I.3.4.5. Les images en couleurs

Les pixels dans cette représentation sont codés sur 3 couleurs (RVB) : le Rouge, le Vert et le Bleu et la somme des trois valeurs RVB donne une couleur précise pour le pixel **[5].**

I.3.4.6. Multi-bandes

Une image multi-bande est une image détenue par un capteur qui opère dans plusieurs bandes spectrales.

Le spectre électromagnétique est la répartition des fréquences (ou des longueurs d'onde) du rayonnement électromagnétique (visible et invisible) depuis les rayons Gamma jusqu'aux ondes hertziennes.

Une bande spectrale est un intervalle du spectre électromagnétique, compris entre deux limites précises en fréquences (ou en longueurs d'onde).

a) Structures de fichiers images

Pour faciliter le traitement et l'extraction de l'information multidimensionnelle des images numériques multi bandes, ces dernières sont stockées selon trois structures:

- Structure par pixels intercalés (BIP) (Band Inter leaved Pixel), pour cette structure les images multi bandes sont stockées par pixels intercalés.
- Structure séquentielle (BSQ) (Binary Stream Sequential), pour cette structure les images multi-bandes sont stockées l'une après l'autre dans le fichier.
- Structure par lignes intercalées (BIL) (Band Inter leaved Line), pour cette structure les images multi-bandes sont stockées par lignes intercalées.

I.4. Les différents formats d'images:

Nom du format	note	Type d'image	Compression des données	Nombre de couleurs supportées	Affichage progressif	Animation
TIFF Tagged Image File Format	- orienté vers les professionnels - reconnu sur tous types de système d'exploitation - permet d'obtenir une image de très bonne qualité - de taille volumineuse, même si elle est inférieure à celle des fichiers BMP	Matriciel	Compression ou pas avec ou sans pertes.	De monochrome à 16 millions.	Non	Non
JPEG **JPEG 2000** *Joint Photographic Expert Group*	-Conçu pour les photographies. -offre des taux de compression inégalés. - devenu le standard des formats d'image sur internet. - A utiliser avec précautions car ça peut brouiller l'image.	Matriciel	Oui, avec perte.	JPEG(16 millions) et JPEG 2000 (32 millions)	Oui	JPEG(Non) et JPEG 2000 (Oui)
GIF *Graphics Interchange Format*	-Très utilisé sur le Web malgré ses faiblesses. - Pose un problème de droit sur son format. de compression. -Déconseillé pour les photos.	Matriciel	Oui, sans perte.	256 maxi (palette)	Oui	Oui
PNG *Portable Network Graphic*	- permet le détourage des images - offre une image parfaite, avec un excellent rendu des nuances et des dégradés.	Matriciel	Oui, sans perte.	Palettisé (256 couleurs ou moins) ou 16 millions	Oui	Non

Tableau I.1 : les différents formats d'images.

I.5. Classification des algorithmes de compression

Le but principal d'un algorithme de compression est de réduire la quantité de données pour faciliter son stockage ou son transport.

Il existe plusieurs façons de comparer les types de compression. Pour cette raison, nous allons voir comment classifier les algorithmes de compression.

I.5.1. La compression sans / avec perte

La perte ou non des données représente le critère le plus important dans la classification. Les deux principales opérations dans la compression de données sont: la compression et la décompression. Ces étapes permettent de définir deux catégories de compression: la compression sans perte et avec pertes.

I.5.1.1. La compression sans perte

Une bonne partie des schémas de compression utilisés sont appelés sans pertes (connu aussi sous le nom de non destructible, réversible, ou conservative).

Elle est utilisée quand il est nécessaire de garder l'information intacte: il ne doit pas y avoir de différences entre le fichier original et ce même fichier après compression et décompression. Ce type de compression est vital non seulement pour le texte, mais également pour tout type de fichier devant conserver une qualité optimale.

I.5.1.2. Compression avec pertes

La compression avec pertes est aussi appelée communément compression destructrice. La compression avec pertes utilise des algorithmes qui compressent les données en les dégradant. Pour éviter que ces dégradations soient perçues par les utilisateurs, les concepteurs d'algorithmes pour la compression destructrice se basent sur les limites de perception au niveau de l'ouïe et de la vue chez l'être humain **[7]**.

Les algorithmes de compression destructrice sont utilisés pour compresser des données graphiques, audio et vidéo. Ils ne pourraient être appliqués sur des données textuelles ou sur celles d'un programme au risque de les rendre illisibles ou inexécutables. La perte d'information est irréversible, 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 conservative*.

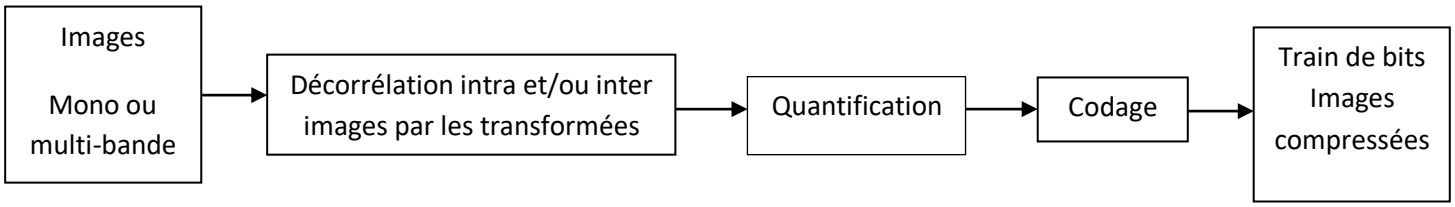


Figure I.1: Schéma de la compression.

I.6. Transformées discrètes

La transmission des images ou vidéos passe par des étapes de compression, quantification et codage. Les algorithmes de compression ont un grand besoin des transformées discrètes pour compresser les données. Les transformées discrètes sont donc une étape principale dans la compression.

I.6.1. Transformée en cosinus discrète « DCT »

C'est le fait d'appliquer une transformée en cosinus discrète à un bloc de pixels pour retirer la redondance des données de l'image.

La DCT est une variante de la transformée de Fourier. Elle a pour but de regrouper l'énergie en passant de la notion de pixels et couleurs à la notion de fréquences et amplitudes. En effet, elle prend un ensemble de points d'un domaine spatial et les transforme en une représentation équivalente dans le domaine fréquentiel. Pour une image couleur, il sera traité trois fonctions (de manière indépendantes) à 3 dimensions : X et Y, indiquant le pixel, et Z la valeur du pixel en ce point. Ces trois fonctions correspondent chacune à un des canaux RVB. Après l'application de cette transformée sur un bloc, on va avoir l'information essentielle stockée dans les basses fréquences et l'énergie sera regroupée en haut à gauche de la matrice. Pour ce qui est des hautes fréquences, on les retrouve en bas à droite de la matrice, plus on approche du bas droite de la matrice plus leurs coefficients tendent vers 0**[8].**

Cette transformée va être appliquée à chaque composante pour chaque bloc. En effet, pour une image codée sur 24bits, on va appliquer la DCT 3 fois sur chaque bloc (1 pour la composante Y, 1 pour Cb ou U et 1 pour V ou Cr) (figure I.2).

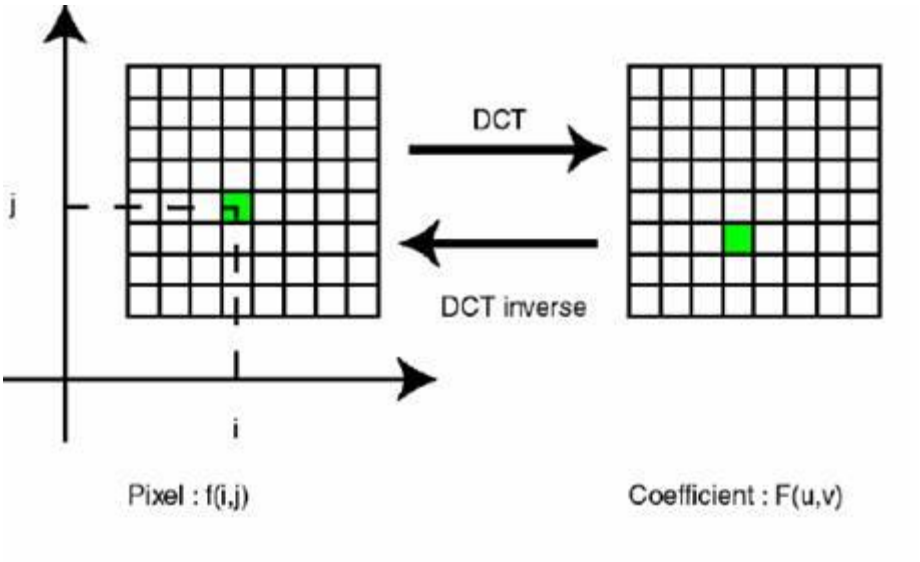


Figure I.2 ***: Application de la DCT sur chaque bloc.***

La DCT s'exprime mathématiquement comme suit **[8]** :

$$DCT(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left(\frac{(2x+1)i\pi}{2N}\right) \cos\left(\frac{(2y+1)j\pi}{2N}\right) \quad \textbf{(I.1)}$$

- 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.
- $C(x) = \frac{1}{\sqrt{2N}}$ $si\ x = 0, C(x) = 0\ sinon.$

-p (x,y) est la valeur du pixel aux coordonnées (x,y)

Ce qui donne dans notre cas (bloc de 8x8 pixels) **[8]** :

$$G_{u,v} = \alpha(u)\alpha(v) \sum_{x=0}^{7} \sum_{y=0}^{7} g_{x,y} \cos\left[\frac{\pi}{8}\left(x + \frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y + \frac{1}{2}\right)v\right] \quad \textbf{(I.2)}$$

$$\alpha_p(n) = \begin{cases} \sqrt{\frac{1}{8}}, & si\ \ n = 0 \\ \sqrt{\frac{2}{8}}, & si\ non \end{cases}$$

$g_{x,y}$: La valeur du pixel aux coordonnées (x,y).

$G_{u,v}$: La valeur de la DCT aux coordonnées (u,v).

La figure I.3 suivant montre les différentes 64 sous-images produites par la DCT :

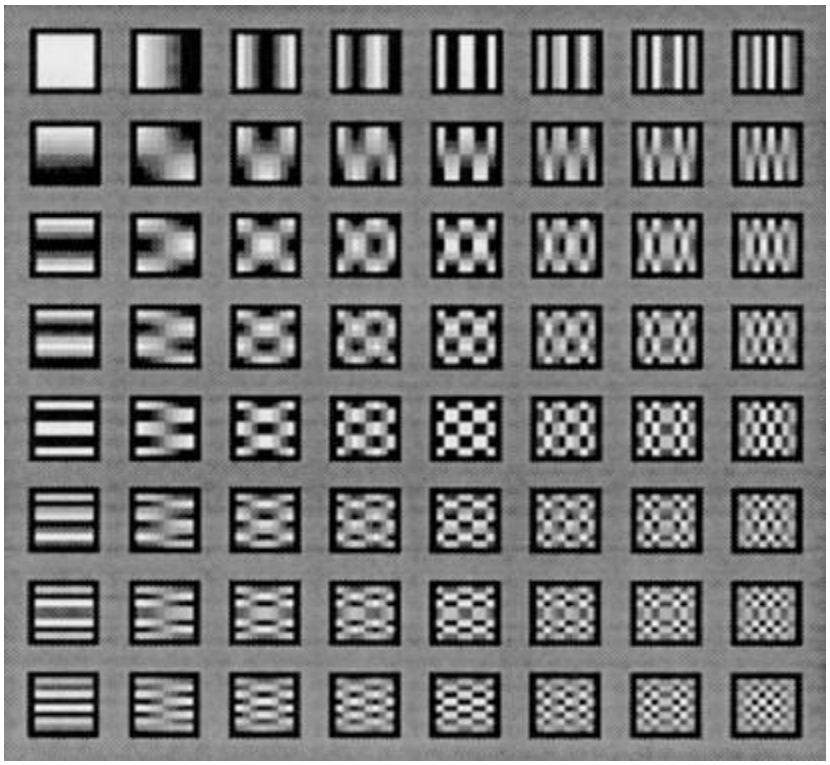


Figure I.3 : les différentes 64 sous-images produites par la DCT.

Avant d'appliquer la DCT, on doit faire une petite transformation au niveau des matrices. En effet, pour un pixel codé sur n=8 bits, on doit soustraire à chaque valeur $2^n/2$ =128 et ce pour passer de l'intervalle [0,255] à l'intervalle [-128,127] (mettre à zéro la gamme des valeurs). Après l'application de la DCT, on arrondit les nombres pour avoir des entiers **[7]**.

L'objectif principal de cette étape est d'obtenir une nouvelle représentation de chaque bloc contenant la même information et cette information concentrée sur peu d'éléments et c'est l'étape qui coûte le plus de temps mais elle est très importante puisqu'elle nous permet de séparer les basses fréquences et les hautes fréquences **[7]**.

a) Codage des coefficients

- ***Codage différentiel du coefficient DC***

La somme des pixels d'un bloc, représentée par le coefficient T(0; 0), est souvent très importante, il est alors plus avantageux d'utiliser un codage différentiel. Ce codage consiste à coder la différence entre le coefficient DC d'un bloc et le coefficient DC du bloc précédent.

- ***Codage par plage des coefficients AC***

Les coefficients AC sont parcourus en Zig-Zag et ensuite codés par plages afin d'exploiter le nombre de coefficients haute-fréquence (égales à 0), comme le montre la figure I.4.

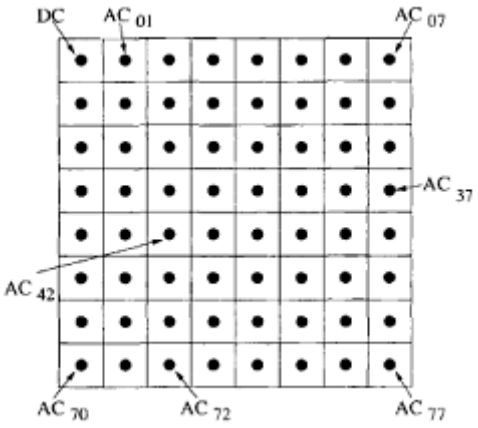


Figure I.4 : Les coefficients DC et AC.

I.6.2. Transformée en ondelette discrète « DWT »

La compression par ondelettes, aussi appelée DWT (Discrete Wavelet Transform) est une méthode basée sur la théorie mathématique de l'analyse du signal. Les ondelettes sont un ensemble de signaux élémentaires à partir desquels on peut reconstruire un signal complexe. La compression par ondelettes consistera donc à décomposer l'image perçue comme un signal en un ensemble d'images de plus petite résolution. Ce procédé se développe en trois phases :

- Tout d'abord, on procède à une transformation de l'image en ondelettes selon un schéma à plusieurs niveaux, processus relativement complexe.
- Ensuite, on réalise une quantification des informations. Lors de cette phase, les détails qui se situent au-dessous d'un certain seuil sont purement et simplement éliminés. C'est donc à ce niveau que se produit la perte d'informations.
- Enfin, on termine en codant les informations.

La décompression des images s'opère par le schéma inverse: les informations sont tout d'abord décodées pour fournir un ensemble à plusieurs niveaux d'ondelettes qui permettent la reconstitution progressive de l'image **[9]**.

La compression par ondelettes, particulièrement appropriée pour les images, est vraisemblablement la méthode de compression la plus performante à l'heure actuelle et est la base du JPEG 2000. Ses avantages sont multiples. En outre, elle offre une meilleure définition des détails et permet un téléchargement progressif de l'image **[9]**.

I.7. Méthodes de compression

I.7.1. Compression par JPEG

La compression JPEG (Joint Photographic Experts Group) est une méthode de compression non conservative de l'image. Elle s'appuie sur une analyse de la perception de l'œil humain, ainsi que sur l'usage de codeurs classiques (RLE, et Huffman) **[8]**.

Cette méthode est très performante puisqu'elle peut réduire la taille d'une image de près de 90%. Cependant, pour atteindre un tel taux, l'utilisateur doit faire un compromis Taille/Qualité. En effet, le caractère non conservatif de la compression provoque une dégradation de l'image proportionnelle au facteur de compression **[8]**.

La chaîne de compression du standard JPEG est illustrée sur la figure I.5 :

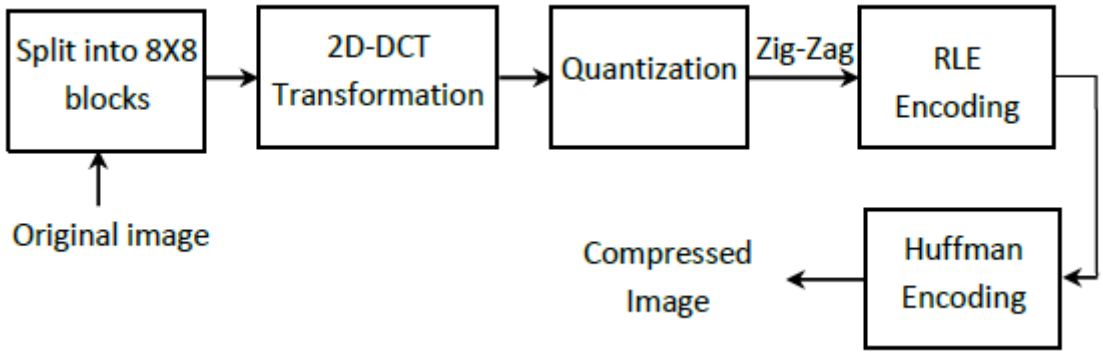


Figure I.5 : La chaîne de compression du standard JPEG.

L'algorithme a besoin de réaliser plusieurs étapes afin de pouvoir compresser une image. Premièrement, l'image de l'entrée est divisée en plusieurs blocs de taille fixe 8x8 pixel. Ensuite, la transformée 2D-DCT est appliquée sur chaque bloc afin de séparer les basses fréquences des

hautes fréquences. Par la suite, chaque bloc DCT est quantifié uniformément. Le résultat de la quantification est réorganisé à nouveau d'une manière zigzag, des fréquences basses aux fréquences hautes. Après cela, le codage RLE est appliqué pour réduire la longueur des séquences produites. Finalement, le processus de codage entropique réversible (tel que Huffman ou codage arithmétique) est effectué sur les données quantifiées pour produire des mots de code de longueur fixes ou variables.

I.7.2. Compression par JPEG2000

JPEG2000 est un système de codage d'image qui utilise les techniques états de l'art de la compression basées sur la transformée en ondelettes. Deux types d'ondelettes sont sélectionnés : l'ondelette Daubechies 9/7 pour la compression avec perte et la transformée réversible 5/3, implémenté via un schéma lifting entier, dans le cas d'une compression sans perte.

JPEG2000 qui se base sur un processus très complexe qui fait intervenir une chaîne d'opération qui commence par la transformée en ondelettes dyadique, une allocation de bits, une quantification et un codage entropique **[10]**.

I.8. Quantité d'informations

La théorie d'information permet d'évaluer la quantité d'information dans une image. Pour cela, chaque point d'une image est considéré comme une variable aléatoire**.**

Soit P un point d'une image en niveau de gris. C'est une variable aléatoire dont les valeurs sont des entiers de l'intervalle [0 255]**.**

Soit P (ni), la probabilité pour que le niveau de gris en P soit ni.

- ✓ La quantité d'information est donnée par :

$$\boldsymbol{I(n_i) = \log_a\left(\frac{1}{P(n_i)}\right) = -\log_2\big(P(n_i)\big)} \qquad (\boldsymbol{I.3})$$

I $(\boldsymbol{n_i})$: est l'incertitude (quantité d'information).

- ✓ On définit l'entropie comme suit:

$$\boldsymbol{H(p)\sum_{i=1}^{N} P(n_i)I(n_i) = -\sum_{i=1}^{N} p(n_i)\ \log_2\big(P(n_i)\big)} \qquad (\boldsymbol{I.4})$$

L'entropie H(p) est exprimé en bits/symbols et le logarithme est en base 2 car un bit peut avoir 2 valeurs différentes.

I.9. Définition des critères utilisés en compression

I.9.1. Taux de compression

Le taux de compression est un élément fondamental dans le domaine de la compression. C'est un outil de mesure qui évalue le degré de compression réalisé **[6].**

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.

Il est donc calculé comme suit:

$$\tau = \frac{\textbf{nombre de bit utilisés pour représenter l'image originale}}{\textbf{nombre de bit utilisés par l'image compressée}}$$

$$\tau = \frac{N \times M \times B}{Bitstream} \quad \textbf{(I.5)}$$

Où:

N, M : représentant respectivement le nombre de lignes et le nombre de colonnes de l'image.

B: désigne le nombre de bit utilisés pour la représentation d'un pixel.

I.9.2. le 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, il est défini comme suit:

$$débit = \frac{\textbf{nombre de bits par pixel dans l'image originale}}{taux}$$ Bit par pixel (Bpp)

$$débit = \frac{Bitstream}{N \times M} \quad \textbf{(I.6)}$$

I.10. Mesure de distorsion

La mesure de la distorsion est un sujet difficile. Il existe deux mesures couramment employées: l'erreur quadratique moyenne EQM (ou MSE pour Mean Square Error)**[11],** qui est définie de la manière suivante:

$$\mathbf{MSE} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(I(i,j)-\hat{I}(i,j))^2}{N \times M} \qquad \textbf{(I.7)}$$

I: L'image originale.

Î: L'image décodée.

M: le nombre de lignes de l'image.

N : le nombre de colonnes de l'image.

(i,j) : positionnement des pixels.

Et le rapport signal/bruit à son maximum : PSNR (Peak Signal Noise Ratio), qui est une métrique objective couramment utilisée pour mesurer la distorsion, son unité est le décibel (dB). Il est défini par la relation suivante :

$$\boldsymbol{PSNR = 10\log_{10}\left(\frac{(255)^2}{MSE}\right)} \qquad \textbf{(I.8)}$$

Ou 255 est la valeur maximal d'un pixel pour une image codée par 8 bit/pixel en niveaux de gris.

Le rapport signal/bruit (SNR) est une autre variante de l'évaluation objective de la qualité, également exprimée en dB. Il a comme expression:

$$\boldsymbol{SNR = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} I^2(i,j)}{\sum_{i=1}^{N}\sum_{j=1}^{M}(I(i,j)-\hat{I}(i,j)}} \qquad \textbf{(I.9)}$$

I.11. Conclusion

Dans ce chapitre nous avons présenté les concepts et les outils rattachés aux domaines des images et de compression de données. A la suite, nous avons discuté la classification classique de compression qui se base sur la perte ou non des données. Des transformées

couramment utilisées en compression (telles que la DCT, la transformée en ondelettes), et des méthodes de compression souvent employées (telles que JPEG et JPEG2000), ont été également décrites. Enfin, les outils nécessaires pour évaluer et comparer les performances de compression sont fournis à cet effet.

Chapitre II

Chapitre II

Outils et techniques du codage source avec application en compression d'images sans perte

II.1. Introduction

Les méthodes de compression ont pour effet de réduire le nombre moyen de bits par pixel (bpp) à stocker ou à transmettre, en tirant parti de la redondance informationnelle de l'image. Ce chapitre présente les informations de base importantes sur les méthodes principales de codage source, sans perte: Shannon-Fano, Huffman, Arithmétique, RLE, LZW,..etc. Dans la dernière section, nous présentons quatre variantes possibles des deux techniques: Huffman et arithmétique en vue de leur application en compression d'images sans perte.

II.2. Définition du codage

Le codage est une étape essentielle dans la compression, c'est une méthode de transformation qui convertit la représentions d'une information en une autre **[11].**

Il existe plusieurs techniques de codage. On citera Shannon Fano, Huffman, arithmétique, etc...

II.3. Techniques de compression sans perte

La croissance extrêmement rapide des données qui doivent être stockées et transférées a suscité la demande de meilleures techniques de transmission et de stockage. Les compressions de

données sans perte classées en deux types sont les suivantes: modèles et code et modèles de dictionnaire. Divers algorithmes de compression de données sans perte ont été proposés et utilisés. Le codage de Huffman, le codage arithmétique, l'algorithme de Shannon Fano, l'algorithme de codage par longueur de parcours sont quelques-unes des techniques utilisées **[12].**

II.3.1. Le codage de Shannon-Fano

Les principaux pionniers de la compression de données furent sans doute C. Shannon et R. M. Fano. Ils démontrèrent que l'on peut coder un message en utilisant la probabilité d'apparition d'un symbole ainsi que la base 2. Le principe est de recoder des informations sur moins de 8 bits**.**

II.3.1.1. L'algorithme de compression Shannon-Fano

Il utilise des codes de longueur variable, comportant d'autant plus de bits que la probabilité du symbole est faible. Les codes sont définis à l'aide d'un algorithme spécifique selon un arbre de Shannon-Fano **[14].**

1. Les symboles sont triés et classés en fonction de leur fréquence en commençant par le plus fréquent.
2. La liste des symboles est ensuite divisée en deux parties de manière à ce que le total des fréquences de chaque partie soit aussi proche que possible.
3. Le chiffre binaire 0 est affecté à la première partie de la liste, le chiffre 1 à la deuxième partie.
4. On répète les étapes 2 et 3 sur chacune des parties. Et ainsi de suite jusqu'à ce que chaque symbole corresponde à un code déterminé **[14].**

Remarque

Cette méthode ne permet pas d'approcher efficacement l'entropie. On lui préfère l'algorithme de Huffman.

II.3.2. Modèles et code

Code de modèle divisé en codage de Huffman et codage arithmétique.

II.3.2.1. Le codage de Huffmann

Un premier algorithme de codage de Huffman a été développé par David Huffman en 1951. Le codage de Huffman est un algorithme de codage entropique utilisé pour la compression de données sans perte. Dans cet algorithme, les codes de longueur fixe sont remplacés par des codes de longueur variable. Lorsque vous utilisez des mots de code de longueur variable, il est souhaitable de créer un code de préfixe, évitant ainsi la nécessité d'un séparateur pour déterminer les limites des mots de code. Huffman utilise un tel code de préfixe **[12].**

Cet algorithme permet d'obtenir de bons résultats, mais il faut conserver entre la compression et la décompression, le dictionnaire des codes utilisés**.**

La procédure de Huffman repose sur deux observations concernant les codes de préfixe optimaux **[13].**

1. Dans un code optimal, les symboles qui apparaissent plus fréquemment (ont une probabilité d'occurrence plus élevée) auront des mots de code plus courts que les symboles moins fréquents.
2. Dans un code optimal, les deux symboles les moins fréquents auront la même longueur.
3. Les deux mots de code les plus longs ne diffèrent que par leur dernier bit et correspondent au symbole le plus long**.**

a) Le But

Réduire le nombre de bits utilisés pour le codage des caractères fréquents et d'augmenter ce nombre pour des caractères plus rares **[15].**

b) Algorithme

L'algorithme de Huffman utilise l'approche gloutonne, c'est-à-dire qu'à chaque étape, l'algorithme choisit la meilleure option disponible. Un arbre binaire est construit de bas en haut.

- ***Algorithme de compression***

1. On cherche la fréquence des caractères (symboles).
2. On trie les caractères par ordre décroissant de fréquence.

3. On construit un arbre pour donner le code binaire de chaque caractère.

Construction de l'arbre : on relie deux à deux les caractères de fréquence les plus basses et on affecte à ce nœud la somme des fréquences des caractères. Puis, on répète ceci jusqu'à ce que l'arbre relie toutes les lettres. L'arbre étant construit, on met un 1 sur la branche à droite du nœud et un 0 sur celle de gauche **[16].**

❖ *Exemple d'encodage de Huffman*

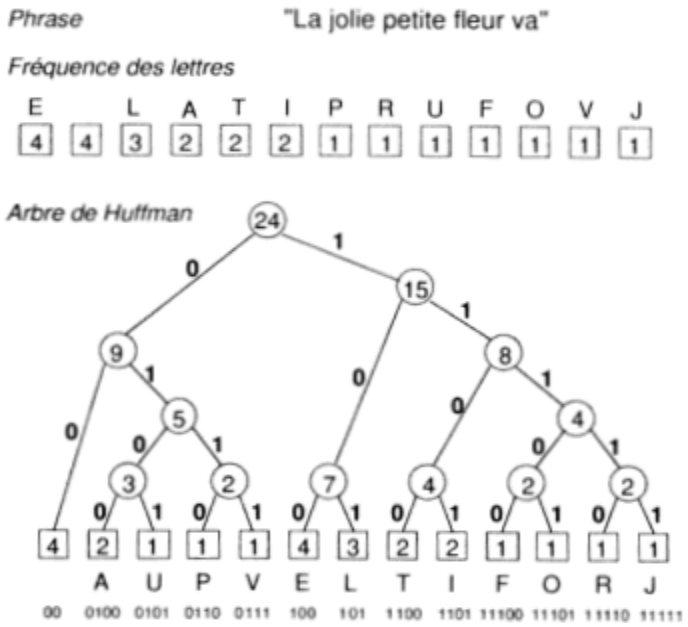


Figure II.1 : Arbre de Huffman.

Le fichier compressé se compose d'une suite de codes sans séparateur, bien que les codes comportent un nombre variable de bits, car chaque code a la propriété d'unicité de son préfixe.

➢ *Algorithme de décompression*

On transmet la bibliothèque de codage, puis on associe les caractères à leur code **[16]**.

c) Utilité et caractéristiques de compression

Cet algorithme permet de compresser aussi bien des images que des textes, on parle de compression statistique ou codage entropique (probabilités d'apparition d'un caractère). On obtient une compression satisfaisante et un temps de compression assez rapide. En revanche il faut transmettre la bibliothèque en même temps.

Pour une source X d'entropie H(X). La longueur moyenne L d'un mot de code obtenu par codage de Huffman vérifie la relation suivant **[13] :**

$$\boldsymbol{H(X) \leq L < H(X) + 1} \qquad \textbf{(II.1)}$$

- ✓ En résumé l'algorithme de codage de Huffman peut-être traduite de la manière suivante :
  1. Les symboles sont triés et classés en fonction de leur fréquence dans un ordre décroissant.
  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 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.
  4. La même démarche est reprise en considérant les deux symboles ou nœuds de poids le plus faible. Elle est renouvelée tant qu'il ne reste plus d'un nœud libre.

II.3.2.2. Codage arithmétique

La compression arithmétique, comme Huffman est une compression statistique, mais elle utilise des réels à la place des bits**.**

Dans cette section, on va présenter les idées principales du codage arithmétique **[17],** qui est une des techniques de compression sans pertes plus performante, et probablement la plus importante dans les applications et les normes actuelles.

Contrairement aux algorithmes de codage utilisant des mots de codes à longueur variable, le code ici, est associé à la séquence et non pas à chaque symbole pris individuellement.

Le codage Arithmétique se singularise par sa capacité à un ensemble de symboles sur un nombre non entier de bits. En réalité, il n'assigne pas un mot de code à chaque symbole mais il associe une valeur de l'intervalle [0,1] à un ensemble de symboles. Le principe repose sur le découpage de l'intervalle [0,1]. Chaque symbole se voit attribué une partition de l'intervalle dont la taille est égale à sa probabilité d'occurrence. L'ordre de rangement est mémorisé pour être utilisé lors du décodage.

L'objectif principal du codage Arithmétique est d'attribuer un intervalle à chaque symbole potentiel. Ensuite, un nombre décimal est attribué à cet intervalle. L'algorithme commence avec un intervalle de 0.0 et 1.0. Après la lecture de chaque symbole d'entrée de l'alphabet, l'intervalle est subdivisé dans un intervalle plus petit proportionnellement à la probabilité du symbole

d'entrée. Ce sous-intervalle devient alors le nouvel intervalle et est divisé en parties en fonction de la probabilité des symboles de l'alphabet saisi. Ceci est répété pour chaque symbole d'entrée. Et, à la fin, tout nombre à virgule flottante du dernier intervalle détermine de manière unique les données d'entrée **[18].**

- Propriétés du codage Arithmétique **[18]:**

1- Il utilise un nombre fractionnaire binaire.
2- Convient aux petits alphabets avec des probabilités très asymétriques.
3- Ce codage prend un flux de symbole d'entrée et le remplace par des nombres à virgule flottante (0, 1).
4- Cela produit des résultats dans un flux de bit.

a) Algorithme [25]

1. Calculer la probabilité associée à chaque symbole dans la chaîne à coder.
2. Associer à chaque symbole un sous-intervalle proportionnel à sa probabilité d'occurrence dans l'intervalle [0 1] (l'ordre de rangement sera mémorisé car il est nécessaire au décodeur).
3. Initialiser la limite inférieure de l'intervalle de travail à la valeur 0 et la limite supérieure à la valeur 1.
4. 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).

5. La limite inférieure code la chaîne de manière unique.

II.3.3. Modèle de dictionnaire

Modèle de dictionnaire divisé en Lempel Ziv Welch (LZW), Run Length Encoding (RLE).

II.3.3.1. Run Length Encoding (RLE)

Run Length Encoding (RLE) est le plus simple des algorithmes de compression de données. Il remplace les passages de deux ou plusieurs caractères identiques par un nombre

représentant la longueur du passage, suivi du caractère d'origine. Les caractères simples sont codés comme des exécutions de 1. La tâche principale de cet algorithme consiste à identifier les exécutions du fichier source et à enregistrer le symbole et la longueur de chaque exécution. L'algorithme Run Length Encoding utilise ces exécutions pour compresser le fichier source d'origine tout en conservant toutes les non-exécutions sans utiliser le processus de compression **[12].**

a) But

Cet algorithme élide les répétitions successives de caractères **[14].**

b) Algorithme de compression

- Recherche des caractères répétés plus de n fois (n fixé par l'utilisateur).
- Remplacement de l'itération de caractères par:

1. un caractère spécial identifiant une compression.
2. le nombre de fois où le caractère est répété.
3. le caractère répété **[16].**

c) Algorithme de décompression

Durant la lecture du fichier compressé, lorsque le caractère spécial est reconnu, on effectue l'opération inverse de la compression tout en supprimant ce caractère spécial.

Exemple : **AAAAARRRRRROLLLLBBBBBUUTTTTTT.**

On choisit comme caractère spécial : **@** et comme seuil de répétition : 3.

Après compression : **@5A@6RO@4L@5BUU@6T** gain : 11 caractères soit 38% **[16].**

d) Utilité et Caractéristiques de compression

➢ *Utilité*

Essentiellement pour la compression des images (car une image est composée de répétitions de pixels, de couleur identique, codés chacun par un caractère) **[14].**

➢ *Caractéristiques de compression*

- algorithme très simple.

-taux de compression relativement faible (40%).

❖ ***Exemple :*** un exemple du codage RLE est illustré sur la figure II.2.

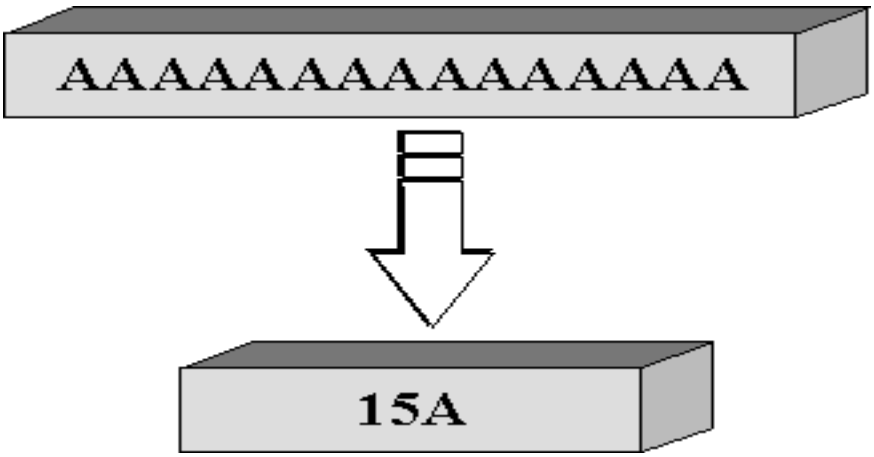


Figure II.2: Compression RLE.

II.3.3.2. Lempel_Ziv_Welch Codage: Lempel_Ziv Welch (LZW)

(LZW) est un algorithme universel de compression de données sans perte proposé par Abraham Lempel, Jacob Ziv et Terry Welch. Il a été publié par Welch en 1984 sous la forme d'une implémentation améliorée de l'algorithme LZ78 publié par Lempel et Ziv en 1978. LZW est un codage basé sur un dictionnaire. Le codage basé sur un dictionnaire peut être statique ou dynamique. Dans le codage statique du dictionnaire, le dictionnaire est fixé lors des processus de codage et de décodage. En codage de dictionnaire dynamique, le dictionnaire est mis à jour à la volée. L'algorithme est simple à mettre en œuvre et peut potentiellement générer un très haut débit dans les implémentations matérielles. Il s'agissait de l'algorithme de l'utilitaire de compression de fichiers UNIX, largement utilisé, et utilisé dans le format d'image GIF. La compression LZW est devenue la première méthode de compression d'image universelle largement utilisée sur les ordinateurs. Un gros fichier texte anglais peut généralement être compressé via LZW à environ la moitié de sa taille originale **[19].**

LZW est capable de travailler avec n'importe quel type de données. Il est rapide en compression et décompression et ne nécessite pas d'opération à virgule flottante. De par le fait qu'il encode au niveau bit et non au niveau de l'octet, il ne se soucie pas du processeur, et de la manière dont il code les informations**.**

LZW utilise la substitution des motifs en se basant sur un dictionnaire construit au fil de la compression.

- Le flot d'information à compresser est découpé en chaînes d'octets. Chaque chaîne est comparée au dictionnaire. Si elle n'est pas présente, elle est stockée. Elle est ensuite écrite dans le flot de sortie compressé.
- Quand une chaîne, déjà rencontrée, apparaît dans le flot, elle est codée et transmise si elle a une longueur inférieure au plus grand mot du dictionnaire.
- Pour le décodage, le logiciel reconstruit en fait le dictionnaire dans le sens inverse. Il n'est pas nécessaire ainsi de transmettre le dictionnaire. Bien souvent, pour la compression et la décompression, le dictionnaire est initialisé avec les 256 valeurs de la table ASCII. Ainsi, tous les codeurs et décodeurs LZW initialisent leurs dictionnaires de la même méthode.
- Le format TIFF effectue une concaténation de points. Les données sont ensuite soumises à la compression LZW. Ceci peut parfois produire quelques problèmes si l'image ne possède pas un nombre pair de point.
- Le format GIF quant à lui code chaque point sur un octet, puis soumet la suite d'octets à la compression LZW**.**

a) But

Cet algorithme réduit la taille des chaînes de caractères (c'est-à-dire les mots) récurrents.

b) *Algorithme de compression LZW*

Cet algorithme utilise une bibliothèque, c'est-à-dire une table de données contenant des chaînes de caractères.

Au cours du traitement de l'information, les chaînes de caractères sont placées une par une dans la bibliothèque. Lorsqu'une chaîne est déjà présente dans la bibliothèque, son code de fréquence d'utilisation est incrémenté. Les chaînes de caractères ayant des codes de fréquence élevés sont remplacées par un " mot " ayant un nombre de caractères le plus petit possible et le code de correspondance est inscrit dans la bibliothèque. On obtient ainsi une information encodée et sa bibliothèque **[16].**

- ✓ L'Algorithme de compression LZW est donné comme suit :

w= NIL;

while (lire un caractère k)

{

if wk existe dans le dictionnaire

w = wk;

else

ajouter wk au dictionnaire;

sortie du code pour w;

w = k;

}
Sortie du code pour w;

- ***Exemple***

La chaîne **:**"^WED^WE^WEE^WEB^WET".

Dictionnaire initial {^, W, E, D, B, T}.

W	K	Sortie	Index	Dictionnaire
NIL	^			
^	W	^	256	^W
W	E	W	257	WE
E		E	258	ED
D	^	D	259	D^
^	W		/	/
^W	E	256	260	^WE
E	^	E	261	E^
^	W	/	/	/
^W	E	/	/	/
^WE	E	260	262	^WEE
E	^	/	/	/
E^	W	261	263	E^W
W	E	/	/	/
WE	B	257	264	WEB
B	^	B	265	B^
^	W	/	/	/
^W	E	/	/	/
^WE	T	260	266	^WET
T	EOP	T		

Tableau II.1 : Codage LZW.

c) Algorithme de décompression LZW

Lors de la lecture de l'information encodée, les " mots " codés sont remplacés dans le fichier par leur correspondance lue dans la bibliothèque et le fichier d'origine est ainsi reconstitué **[16].**

✓ L'algorithme de décompression LZW est donné comme suit:

Lire un personnage k;

Transmettre k;

w = k;

while (lire un caractère k) /* k pourrait être un caractère ou un code. */

{

Entry = entrée du dictionnaire pour k; entrée de sortie;

Ajouter w + entry [1] au dictionnaire;

w = entry;

}

❖ ***Exemple***

La cha*îne* à décoder est : "^WED<256>E<260><261><257>B<260>T".

W	K	Sortie	Index	Dictionnaire
	^	^	/	/
^	W	W	256	^W
W	E	E	257	WE
E	D	D	258	ED
D	**<256>**	^W	259	D^
^W	W	E	260	^WE
E	E	^WE	261	^E
^WE	^	E^	262	^WEE
^E	W	WE	263	E^W
WE	E	B	264	WEB
B	E	^WE	265	B^
^WE	T	T	266	^WET

Tableau II.2: Décodage LZW.

d) Utilité et caractéristiques de compression

Cette méthode est peu efficace pour les images mais donne de bons résultats pour les textes et les données informatiques en général (plus de 50%).

II.4. Avantages et inconvénients de la compression sans perte

Des techniques	Avantages	Désavantages
encodage de longueur (RLE)	cet algorithme est facile à mettre en œuvre et ne nécessite pas beaucoup de puissance processeur **[20].**	La compression RLE n'est efficace qu'avec des fichiers contenant beaucoup de données répétitives **[20].**
Codage LZW	Compression simple, rapide et bonne **[21].** Table de mots de code dynamique construite pour chaque fichier **[21].** La décompression recrée la table de mots de code afin qu'il ne soit pas nécessaire de la passer **[23].**	La compression réelle est difficile à prédire Jin dal **[21].** Il occupe plus d'espace de stockage que le taux de compression optimal **[21].** L'algorithme LZW ne fonctionne que lorsque les données d'entrée sont suffisamment grandes et que la redondance des données est suffisante **[20].**
codage de Huffman	cet algorithme de compression très simple et efficace compressant des fichiers texte ou programme **[21].** Cette technique montre des séquences plus courtes pour des caractères plus fréquents **[20].** Sans préfixe: aucun codage de séquence de bits d'un caractère n'est le préfixe de tout autre codage de séquence de bits **[20].**	Une image compressée par cette technique est mieux compressée par d'autres algorithmes de compression **[21].** L'arbre de code doit également être transmis ainsi que le message (sauf si une table de code ou une table de prédiction est convenue entre l'expéditeur et le destinataire) **[20].** Données entières corrompues par un bit corrompu **[20].** La performance dépend d'une bonne estimation si l'estimation n'est pas meilleure que la performance est mauvaise **[20].**
codage arithmétique	sa capacité à garder le codage et le modeler séparés **[22].** aucune arborescence de code ne doit être transmise au récepteur **[22].** son utilisation les valeurs fractionnaires **[22].**	le codage arithmétique a des opérations complexes car il consiste en additions soustractions multiplications et divisions **[22].** codage arithmétique nettement plus lent que le codage de Huffman, il n'existe aucune précision infinie **[22].** deux structures de problèmes pour stocker les nombres et la division constante de l'intervalle peuvent entraîner un chevauchement de code **[22].**

Tableau II.3: Résumé des avantages et inconvénients de divers algorithmes de compression sans perte.

II.5. Transformer le codage

Pour comprimer efficacement les images, il est nécessaire de prendre en compte le phénomène de corrélation des pixels voisins (coefficients). Grâce à la transformation, les informations redondantes sont éliminées tout en préservant les informations essentielles pour obtenir une image finale de bonne qualité. En effet, l'étape de transformation a pour but de réorganiser les informations en représentant les composantes importantes de l'image (signal) avec le moins de coefficients possibles. Cette étape est également appelée étape de décorrélation, elle est réversible et n'entraîne aucune perte d'information. Entre les transformations utilisées en compression, on peut distinguer plusieurs transformations : la transformée cosinus discrète (DCT), la transformée karhunen-loeve (KLT), la transformée ondelettes discrète (DWT)... etc.

II.6. Transformée en ondelettes réversibles classiques

La transformation en ondelettes a été fréquemment utilisée pour comprimer les images, en particulier avec perte. Pour pouvoir effectuer une compression sans perte à l'aide de la transformée en ondelettes, il faut obtenir des coefficients d'ondelettes avec des valeurs entières afin d'éviter les erreurs de quantification. Ainsi, la reconstruction exacte du signal d'entrée peut être assurée. Pour cela des transformations dites réversibles sont mises en œuvre. De fait, ces transformations ont l'avantage de permettre le développement d'un schéma de compression sans perte et réversible.

II.7. Schéma lifting

Le schéma de lifting a été initié par Win Sweldens **[23]** et offre la possibilité de mettre en œuvre de manière performante les transformées en ondelettes. Cette technique peut être utilisée pour la réalisation des transformations réversibles, et a aussi l'avantage de permettre la construction de transformées en ondelettes sans faire appel à la transformation de Fourier, tout en mettant en œuvre un procédé efficace, rapide et moins complexe. Elle a également l'avantage de faciliter la mise en œuvre de la transformation inverse en inversant uniquement les opérations d'analyse. Il est donc inutile de calculer les filtres duaux pour effectuer la transformation inverse. En outre, les concepts de translation d'une ondelette mère ne sont plus employés.

II.8. Schéma lifting réversible

Pour pouvoir sans inconvénient réaliser une compression sans perte, il est nécessaire et indispensable que le système de compression dispose d'une transformation réversible. La version entière du schéma lifting s'avère un avantage appréciable pour les systèmes de compression progressive en particulier sans perte. En effet, l'élaboration de tel schéma (entier) s'appuis sur l'application d'un opérateur d'arrondi entraînant une transformation réversible qui a pour effet de transformer des entiers en entiers **[24].**

La version entière du schéma lifting offre un avantage appréciable pour les systèmes de compression sans perte. Ainsi, l'élaboration d'un schéma entier est basée sur l'application d'un opérateur d'arrondi dont le résultat une transformation réversible qui a pour effet de transformer des entiers en entiers **[24].**

II.9. Application à la compression d'image

Il existe plusieurs algorithmes, chacun utilise une technique différente. Voici quelques algorithmes :

II.9.1. Algorithmes appliqués directement sur l'image

Les schémas adoptés pour réaliser une compression/décompression sans perte sont illustrés sur les figures II.3 et II.4 :

Figure II.3: Schéma de compression.

FigureII.4 : Schéma de décompression.

II.9.1.1. Variante H1 (Codage de Huffman)

L'algorithme de compression/décompression utilisant le codage de Huffman s'énonce comme suit:

- Lire l'image à compresser
- Transformer l'image en un vecteur.

- Calculer l'Alphabet, les fréquences et les probabilités.
- Effectuer le codage de Huffman.
- Effectuer le décodage de Huffman.
- Transformer le vecteur en une matrice.
- Effectuer le test de compression sans perte
- Calculer l'entropie.
- Calculer le taux de compression
- Calculer le débit en BPP (Bits par pixel)
- Afficher la taille de l'image originale en Bits et en Kilo octets.
- Afficher la taille de l'image compressée en Bits et en Kilo octets.
- Afficher l'entropie, le débit et le taux de compression.
- Afficher l'image originale et l'image décompressée.

II.9.1.2. Variante A1 (Codage Arithmétique)

L'algorithme de compression/décompression faisant appel au codage Arithmétique se résume comme suit:

- Lire l'image à compresser
- Transformer l'image en un vecteur.
- Calculer l'Alphabet, les fréquences et les probabilités.
- Effectuer le codage d'Arithmétique.
- Effectuer le décodage d'Arithmétique.
- Transformer le vecteur en une matrice.
- Effectuer le test de compression sans perte
- Calculer l'entropie.
- Calculer le taux de compression
- Calculer le débit en BPP (Bits par pixel)
- Afficher la taille de l'image originale en Bits et en Kilo octets.
- Afficher la taille de l'image compressée en Bits et en Kilo octets.
- Afficher l'entropie, le débit et le taux de compression
- Afficher l'image originale et l'image décompressée

II.9.2. Algorithmes basées sur une étape de transformation

Les schémas typiques établis pour effectuer une compression/décompression, sans perte, à base d'une transformée réversible (à coefficients entiers) sont illustrés sur les figures II.5 et II.6:

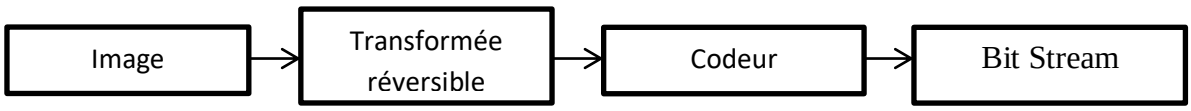


Figure II.5 : Schéma de compression.

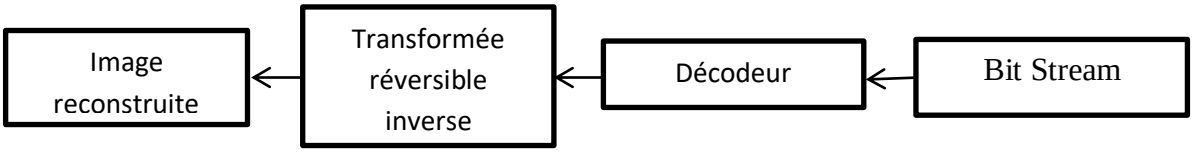


Figure II.6 : Schéma de décompression.

II.9.2.1. Variante H2 (Codage de Huffman + Transformée Réversible (TR))

L'algorithme de compression/décompression employant le codage de Huffman et basée sur une étape de transformation réversible s'établit comme suit:

- Lire l'image à compresser
- Effectuer la transformée réversible directe.
- Transformer l'image en un vecteur.
- Calculer l'Alphabet, les fréquences et les probabilités.
- Effectuer le codage de Huffman.
- Effectuer le décodage de Huffman.
- Transformer le vecteur en une matrice.
- Effectuer la transformée réversible inverse.
- Effectuer le test de compression sans perte
- Calculer l'entropie.
- Calculer le taux de compression
- Calculer le débit en BPP (Bits par pixel)
- Afficher la taille de l'image originale en Bits et en Kilo octets.
- Afficher la taille de l'image compressée en Bits et en Kilo octets.

- Afficher l'entropie, le débit et le taux de compression
- Afficher l'image originale et l'image décompressée.

II.9.2.2. Variante A2 (Codage Arithmétique+ Transformée Réversible(TR))

L'algorithme de compression/décompression par codage Arithmétique et reposant sur une étape de transformation réversible est défini comme suit:

- Lire l'image à compresser
- Effectuer la transformée réversible directe.
- Transformer l'image en un vecteur.
- Calculer l'Alphabet, les fréquences et les probabilités.
- Effectuer le codage d'Arithmétique.
- Effectuer le décodage d'Arithmétique.
- Transformer le vecteur en une matrice.
- Effectuer la transformée réversible inverse.
- Effectuer le test de compression sans perte
- Calculer l'entropie.
- Calculer le taux de compression
- Calculer le débit en BPP (Bits par pixel)
- Afficher la taille de l'image originale en Bits et en Kilo octets.
- Afficher la taille de l'image compressée en Bits et en Kilo octets.
- Afficher l'entropie, le débit et le taux de compression
- Afficher l'image originale et l'image décompressée.

II.10. Conclusion

Ce chapitre décrit plusieurs techniques de codage source, à l'exemple du codage de Shannon-Fano, Huffman, Arithmétique, RLE et LZW. En fait, toutes ces techniques n'engendrent aucune perte d'information et peuvent, par conséquent, être appliquées à la compression sans perte. Le présent chapitre se termine par la présentation de quatre variantes des deux techniques de codage Huffman et Arithmétique. Les variantes H1 et H2 s'appliquent directement sur l'image originale, tandis que les autres variantes A1 et A2 s'appliquent sur l'image transformée. L'évaluation des quatre variantes sera effectuée au chapitre 3.

Chapitre III

Chapitre III
Résultats et discussion

III.1. Introduction

Ce chapitre porte sur l'évaluation des performances de compression sans perte de deux techniques de codage source, présentées dans le chapitre précédent, en l'occurrence le codage de Huffman et le codage Arithmétique. En effet, les résultats de quatre algorithmes, décrits au chapitre précédent, sont comparés avec un ensemble d'images de test en termes de débits et de complexité de calcul. Les deux premiers algorithmes H1 et A1 s'appliquent directement sur l'image à compresser, tandis que les deux autres algorithmes H2 et A2 s'appliquent particulièrement sur l'image transformée, par une transformation en ondelettes réversible à coefficients entier pour assurer la compression sans perte.

III.2. Ensemble de données et ses caractéristiques

1. Cameraman.tif, échelle de gris, dimension 256 x 256, taille 64 Ko.
2. Truck.tiff, échelle de gris, dimension 512 x 512, taille 256 KO.
3. Airplane.tiff, échelle de gris, dimension 512 x 512, taille 256 KO.
4. Tank.tiff, échelle de gris, dimension 512 x 512, taille 256 KO.
5. Apc.tiff, échelle de gris, dimension 512 x 512, taille 256 KO.
6. Stream and bridge.tiff, échelle de gris, dimension 512 x 512, taille 256 KO.
7. Boat.tiff, échelle de gris, dimension 512 x 512, taille 256 KO.

Expérience de travail dans Matlab R2009b.

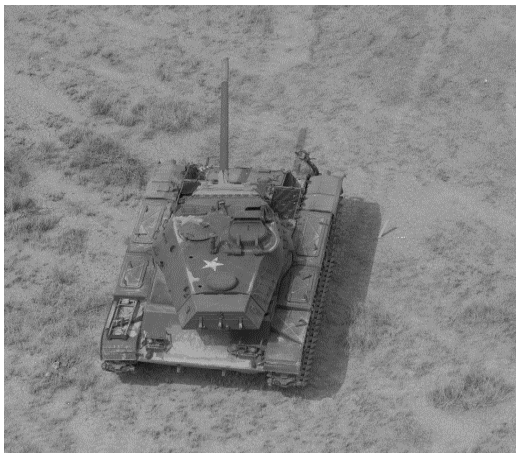
III.2.1. Formats d'image utilisés

(Cameraman.tif)

(**T**ruck.tiff)

(Apc.tiff)

(Tank.tiff)



(Boat.tiff)

(Stream and bridge.tiff)

(Airplane.tiff)

III.3. Expériences

III.3.1. Codeurs appliqués directement sur l'image

III.3.1.1. Comparaison entre le codage de Huffman et le codage Arithmétique en termes de débit (en BPP)

Dans le but de comparer les performances des deux codeurs Huffman et Arithmétique, appliqués directement à l'image à compresser, et utilisés pour l'évaluation des performances de compression sans perte, nous avons utilisé un ensemble d'images en niveau de gris de dimension variable: 256 x 256 et 512 x 512 (8 Bits par pixels). Les performances de la compression sans perte sont obtenues en se basant sur les mesures du débit exprimé en Bits par pixels et de la taille de l'image compressée exprimée en bits et en kilo-octets (K.O). Les résultats obtenus à partir des algorithmes mis en œuvre: H1et A1 sont résumés dans le tableau et la figure suivant.

			Codage de Huffman		**Codage Arithmétique**	
Image	**Taille de l'image originale**	**Entropie**	**Débit**	**Taille de l'image compressée**	**Débit**	**Taille de l'image compressée**
Cameraman	524288 Bits 64 KO	7.0097	7 .0448	461689 Bits 56.3585 KO	**7.0100**	**459406 Bits 56.0798 KO**
Truck	2097152 Bits 256 KO	6.0274	6.0488	1585647 Bits 193.5604 KO	**6.0275**	**1580069 Bits 192.8795 KO**
Airplane	2097152 Bits 256 KO	4.0045	4.0500	1061680 Bits 129.5996 KO	**4.0046**	**1049776 Bits 128.1465 KO**
Tank	2097152 Bits 256 KO	5.4957	5.5197	1446949 Bits 176.6295 KO	**5.4958**	**1440695 Bits 175.8661 KO**
Apc	2097152 Bits 256 KO	5.0534	5.0762	1330688 Bits 162.4375 KO	**5.0535**	**1324750 Bits 161.7126 KO**
Stream and Bridgo	2097152 Bits 256 KO	5.7056	5.7322	1502654 Bits 183.4294 KO	**5.7056**	**1495698 Bits 182.5803 KO**
Boat	2097152 Bits 256 KO	7.1914	7.2187	1892334 Bits 230.9978 KO	**7.1914**	**1885193 Bits 230.1261 KO**
Débit Moyen	/	**5.7840**	5.8192	/	**5.7841**	/
Taux compression moyen	/	***1.3831***	*1.3747*	/	***1.3831***	/

Tableau III.1 : Comparaison des performances des codeurs Huffman (variante H1) et Arithmétique (variante A1).

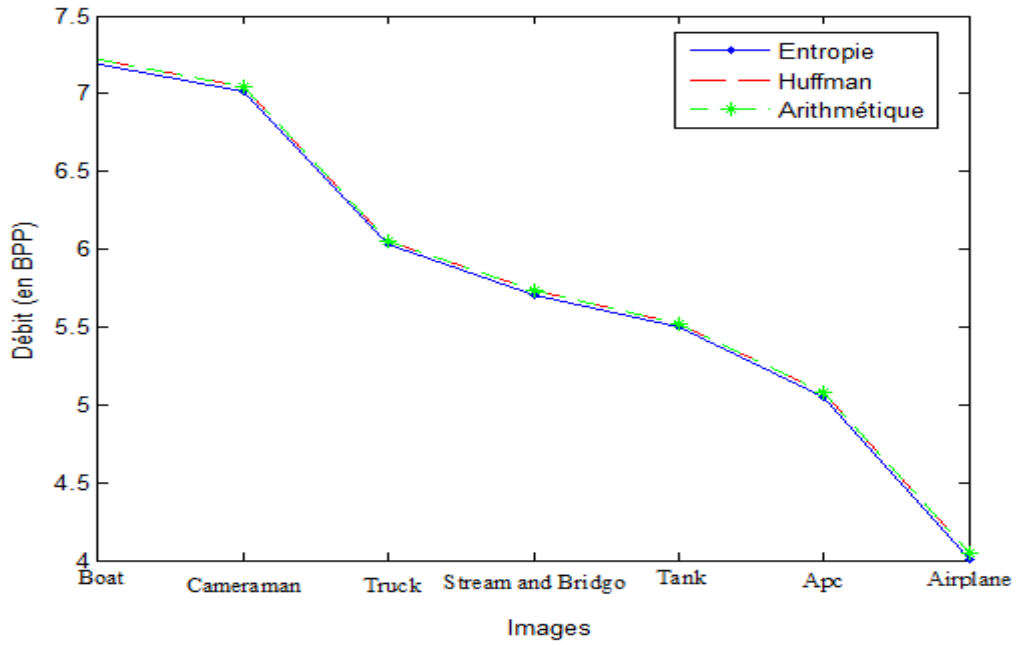


Figure III.1 : Comparaison des performances des codeurs Huffman (variante H1) et Arithmétique (variante A1).

Comme on peut le voir sur le tableau III.1 et la figure III.1, les meilleurs résultats sont obtenus en utilisant la méthode du codage Arithmétique. Par rapport au codage de Huffman, nous observons une diminution du débit (en BPP) et de la taille de l'image compressée (en Bits et en KO) pour toutes les images de test. En moyenne, les performances du codage Arithmétiques sont proches de l'entropie comme l'illustre le tableau III.1 (dernière ligne).

Théoriquement, la taille de l'image originale et l'entropie sont fixes et ne dépendent pas de la méthode de codage.

III.3.2. Comparaison entre le codage de Huffman (variante H1) et le codage Arithmétique (variante A1) en termes de la complexité de calcul

III.3.2.1. Machine PC

- Edition windows: Windows 7 Édition Intégrale.
- Type de système : Système d'exploitation 32 bits.
- Modèle du système: BIOS du portable HP 15: InsydeH2O Version 03.73.06F.31
- Processeur: Intel (R) Core (TM) i3-3110M CPU @ 2.40GHz 2.40 GHz.
- Mémoire installée (RAM) : 4.00 Go.

- Nom de la carte: Famille de cartes graphiques HD Intel (R).

Afin de comparer les performances, en terme de la complexité de calcul, des deux codeurs Huffman (variante H1) et Arithmétique (variante A1), appliqués directement à l'image à compresser, nous avons conservé les mêmes données de test, pour effectuer la compression d'images sans perte. Les performances des algorithmes implémentés H1 et A1 sont évaluées en mesurant les temps de codage/décodage (en secondes) pour chaque méthode. En outre, pour permettre une meilleure comparaison, nous avons utilisé le même P.C. pour les deux méthodes. Le tableau suivant présente les résultats obtenus :

	Codage de Huffman		**Codage Arithmétique**	
Image	**Temps de codage**	**Temps de décodage**	**Temps de codage**	**Temps de décodage**
Cameraman	20.478474	235.578254	**5.962985**	**7.501191**
Truck	76.904716	521.046023	**20.372942**	**26.050463**
Airplane	81.441617	350.899348	**14.618288**	**19.128465**
Tank	73.847167	445.545789	**19.327234**	**24.750983**
Apc	72.942447	408.356245	**17.864714**	**22.930980**
Stream and Bridgo	22.139157	246.544736	**19.775764**	**24.979882**
Boat	91.139690	944.286307	**24.472817**	**30.340937**
Moyenne	61.8955	442.8314	**17.4850**	**22.2404**

Tableau III.2 : Comparaison des performances des deux codeurs Huffman (variante H1) et Arithmétique (variante A1) en termes de la complexité de calcul en mesurant les temps codage et décodage (en secondes).

Comme on peut le constater dans le Tableau III.2, la méthode de Huffman (variante H1) est très coûteuse en temps de calcul, en particulier pour la phase de décodage en comparaison avec la méthode du codage Arithmétique (variante A1) qui réalise des gains significatifs pour le codage aussi bien pour le décodage. Notons aussi que le temps à consacrer au décodage est toujours supérieur au temps de codage pour les 2 méthodes. Toutefois, on remarque une différence notable de ces temps pour le cas de la méthode de Huffman (variante H1).

III.3.3. *Compression par transformée réversible: Codeurs appliqués à l'image transformée*

III.3.3.1. Effet du niveau

Le niveau de décomposition de la transformée réversible est un élément à prendre en considération lors de l'analyse des résultats de la compression. A cet effet, nous avons utilisé le même ensemble d'images en variant les niveaux de décomposition en ondelettes de 1 à 6. Dans le cadre de notre projet de fin d'études, nous avons utilisé la transformée en ondelettes réversible à coefficients entiers (dB1) pour pouvoir effectuer la compression sans perte. Les tableaux III.3 et III.4 et la figure III.2 présentent les résultats obtenus des algorithmes H2 et A2 dans l'étude de l'influence du niveau de décomposition sur les performances de compression.

		Méthode de codage Huffman + TR						
Image		**niveau – 1**	**niveau – 2**	**niveau - 3**	**niveau – 4**	**niveau - 5**	**niveau - 6**	**Moyenne**
Cameraman	**Débit**	6.1292	5.3503	5.0963	5.0225	5.0030	4.9990	5.2667
	Entropie	6.0969	5.3270	5.0699	4.9952	4.9754	4.9715	5.2393
Truck	**Débit**	6.1205	5.4009	5.1376	5.0517	5.0253	5.0174	5.2922
	Entropie	6.0951	6.0951	5.3720	5.1072	5.0205	4.9941	5.2634
Airplane	**Débit**	4.9923	4.2347	3.9412	3.8478	3.8196	3.8114	4.1078
	Entropie	4.9674	4.1971	3.9067	3.8122	3.7836	3.7753	4.0737
Tank	**Débit**	6.1983	5.4723	5.1985	5.1089	5.0821	5.0741	5.3557
	Entropie	6.1653	5.4476	5.1735	5.0841	5.0571	5.0492	5.3295
Apc	**Débit**	5.7966	5.1004	4.8288	4.7416	4.7142	4.7060	4.9813
	Entropie	5.7691	5.0704	4.7974	4.7077	4.6800	4.6717	4.9494
Stream and Bridgo	**Débit**	6.7128	6.0958	5.8547	5.7791	5.7570	5.7512	5.9918
	Entropie	6.6823	6.0669	5.8258	5.7505	5.7285	5.7226	5.9628
Boat	**Débit**	6.5061	5.7197	5.4550	5.3740	5.3259	5.3441	5.6249
	Entropie	6.4741	5.6951	5.4301	5.3493	5.3259	5.3195	5.5990

Tableau III.3 : Résultats de compression de la méthode de codage de Huffman dans le domaine transformée (variante H2) sur plusieurs niveaux de décomposition en ondelettes réversible.

		Méthode de codage Arithmétique + TR						
Image		**niveau – 1**	**niveau– 2**	**niveau-3**	**niveau-4**	**niveau-5**	**niveau-6**	**Moyenne**
Cameraman	**Débit**	6.0972	5.3273	5.0702	4.9954	4.9757	4.9718	5.2396
	Entropie	6.0969	5.3270	5.0699	4.9952	4.9754	4.9715	5.2393
Truck	**Débit**	6.0952	5.3720	5.1072	5.0206	4.9942	4.9863	5.2626
	Entropie	6.0951	5.3720	5.1072	5.0205	4.9941	4.9862	5.2625
Airplane	**Débit**	4.9674	4.1972	3.9068	3.8123	3.7837	3.7754	4.0738
	Entropie	4.9674	4.1971	3.9067	3.8122	3.7836	3.7753	4.0737
Tank	**Débit**	6.1654	5.4477	5.1735	5.0842	5.0572	5.0493	5.3295
	Entropie	6.1653	5.4476	5.1735	5.0841	5.0571	5.0492	5.3295
Apc	**Débit**	5.7692	5.0705	4.7975	4.7077	4.6801	4.6718	4.9495
	Entropie	5.7691	5.0704	4.7974	4.7077	4.6800	4.6717	4.9494
Stream and Bridgo	**Débit**	6.6824	6.0669	5.8259	5.7505	5.7285	5.7227	5.9628
	Entropie	6.6823	6.0669	5.8258	5.7505	5.7285	5.7226	5.9628
Boat	**Débit**	6.4742	5.6952	5.4301	5.3494	5.3260	5.3196	5.5991
	Entropie	6.4741	5.6951	5.4301	5.3493	5.3259	5.3195	5.5990

Tableau III.4 : Résultats de compression de la méthode de codage Arithmétique dans le domaine transformée (variante A2) sur plusieurs niveaux de décomposition en ondelettes réversible.

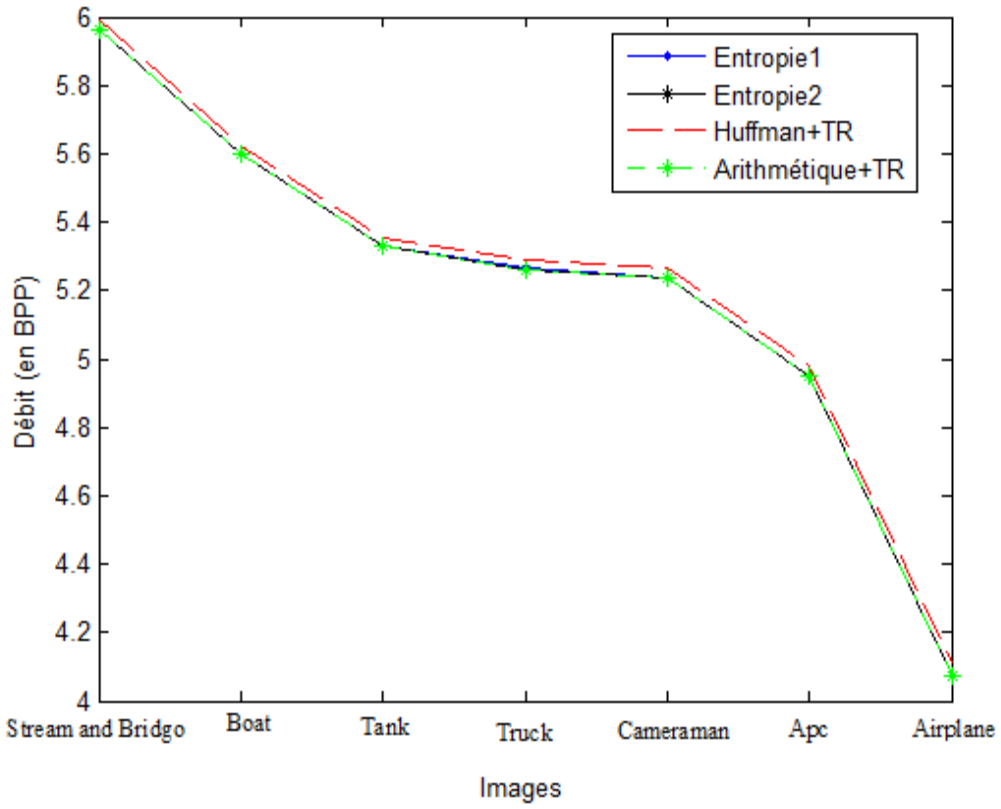


Figure III.2 : Courbe de comparaison des performances des variantes H2 et A2 sur plusieurs niveaux de décomposition en ondelettes réversible.

L'analyse des résultats montre que l'entropie et le débit sont inversement proportionnels au niveau de décomposition. En effet, on constate une importante diminution de l'entropie et du débit entre les niveaux 1 et 6 (ou 5). Ainsi, les meilleurs résultats enregistrés sont aux niveaux 5 et 6 pour les deux méthodes de codages Huffman (variante H2) et Arithmétique (variante A2) appliqués au domaine transformé, comme l'illustrent les tableaux III.3 et III.4 et la figure III.2.

III.3.4. Comparaison entre le codage de Huffman (variante H1) et le codage de Huffman dans le domaine transformé (variante H2)

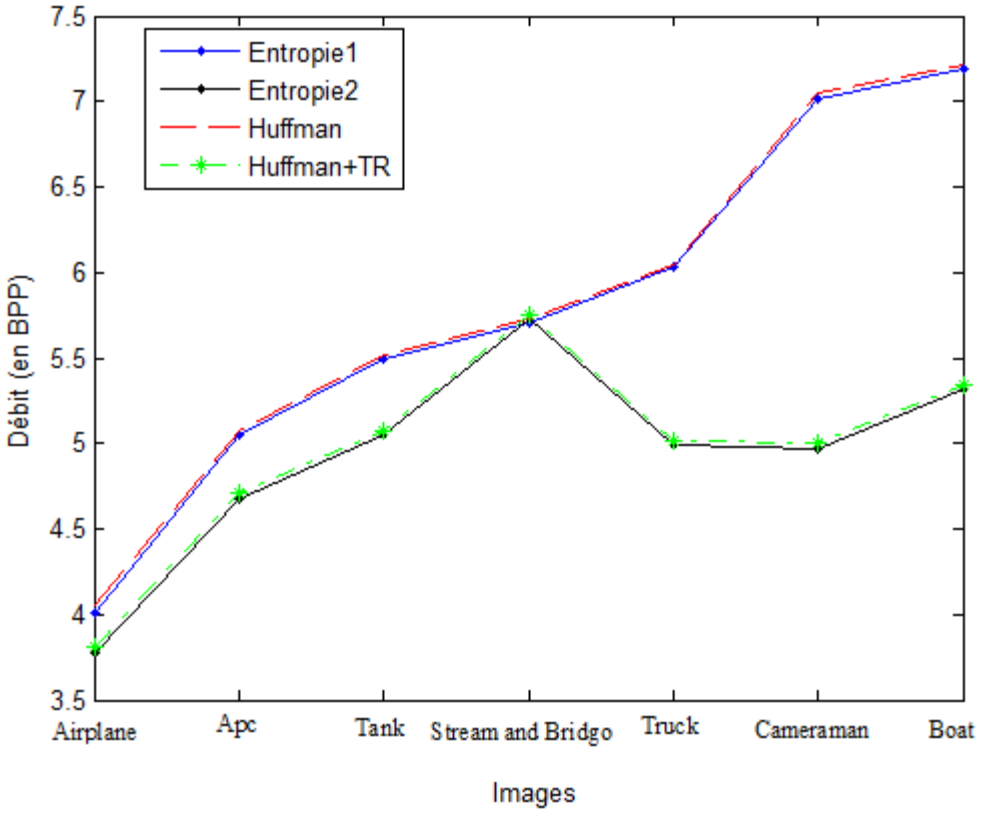
Nous comparerons dans cette section les quatre approches ainsi que les résultats obtenus avec les mêmes données de test. Pour les méthodes de codage mises en œuvre dans le domaine transformé, à savoir H2 et A2, nous avons choisi l'ondelette réversible (dB1) avec 6 niveaux de décomposition. L'analyse comparative des résultats de compression d'images sans perte est effectuée entre:

- ✓ Le codage de Huffman (H1) et le codage de Huffman dans le domaine transformé (H2)
- ✓ Le codage Arithmétique (A1) et le codage Arithmétique dans le domaine transformé (A2)
- ✓ Le codage de Huffman dans le domaine transformé (H2) et le codage Arithmétique dans le domaine transformé (A2)

III.3.4.1. Comparaison entre les algorithmes (H1 et H2), (A1 et A2)

		Méthode de codage Huffman			**Méthode de codage Huffman+TR**		
Image	**Taille de l'image Originale**	**Débit**	**Entropie**	**Taille de l'image compressée**	**Débit**	**Entropie**	**Taille de l'image compressée**
Cameraman	524288 Bits 64 KO	7.0448	7.0097	461689 Bits 56.3585 KO	**4.9990**	**4.9715**	**327614 Bits** **39.9919 KO**
Truck	2097152 Bits 256 KO	6.0488	6.0274	1585647 Bits 193.5604 KO	**5.0174**	**4.9862**	**1315278 Bits** **160.5564 KO**
Airplane	2097152 Bits 256 KO	4.0500	4.0045	1061680 Bits 129.5996 KO	**3.8114**	**3.7753**	**999136 Bits** **121.9648 KO**
Tank	2097152 Bits 256 KO	5.5197	5.4957	1446949 Bits 176.6295 KO	**5.0741**	**5.0492**	**1330134 Bits** **162.3699 KO**
Apc	2097152 Bits 256 KO	5.0762	5.0534	1330688 Bits 162.4375 KO	**4.7060**	**4.6717**	**1233658 Bits** **150.5930 KO**
Stream and Bridgo	2097152 Bits 256 KO	5.7322	5.7056	1502654 Bits 183.4294 KO	**5.7512**	**5.7226**	**1507631 Bits** **184.0370 KO**
Boat	2097152 Bits 256 KO	7.2187	7.1914	1892334 Bits 230.9978 KO	**5.3441**	**5.3195**	**1400912 Bits** **171.0098 KO**
Débit moyen	/	5.8192	5.7840	/	**4.9576**	**4.9280**	/
Taux de compression moyen	/	1.3747	1.3831	/	**1.6136**	**1.6233**	/

Tableau III.5 : Comparaison des performances des méthodes: codage Huffman (variante H1) et codage Huffman dans le domaine transformé (variante H2).



FigureIII.3 : Comparaison des performances des méthodes: codage Huffman (variante H1) et codage Huffman dans le domaine transformé (variante H2).

		Méthode de codage Arithmétique			Méthode de codage Arithmétique + TR		
Image	**Taille de l'image Originale**	**Débit**	**Entropie**	**Taille de l'image compressée**	**Débit**	**Entropie**	**Taille de l'image compressée**
Cameraman	524288 Bits 64 KO	7.0100	7.0097	459406 Bits 56.0797 KO	**4.9718**	**4.9715**	325832 Bits 39.7744 KO
Truck	2097152 Bits 256 KO	6.0275	6.0275	1580069 Bits 192.8795 KO	**4.9863**	**4.9862**	1307119 Bits 159.5604 KO
Airplane	2097152 Bits 256 KO	4.0046	4.0045	1049776 Bits 128.1465 KO	**3.7754**	**3.7753**	989694 Bits 120.8123KO
Tank	2097152 Bits 256 KO	5.4958	5.4957	1440695 Bits 175.8661 KO	**5.0493**	**5.0492**	1323638 Bits 161.5769KO
Apc	2097152 Bits 256 KO	5.0535	5.0534	1324750 Bits 161.7126 KO	**4.6718**	**4.6717**	1224689 Bits 149.4982 KO
Stream and Bridgo	2097152 Bits 256 KO	5.7056	5.7056	1495698 Bits 182.5803 KO	**5.7227**	**5.7226**	1500170 Bits 183.1262 KO
Boat	2097152 Bits 256 KO	7.1914	7.1914	1885193 Bits 230.1261 KO	**5.3196**	**5.3195**	1394502 Bits 170.2273 KO
Débit moyen	/	5.7841	5.7840	/	**4.9281**	**4.9280**	/
Taux de compression moyen	/	1.3831	1.3831	/	**1.6233**	**1.6233**	/

Tableau III.6 : Comparaison des performances des méthodes: codage Arithmétique (variante A1) et codage Arithmétique dans le domaine transformé (variante A2).

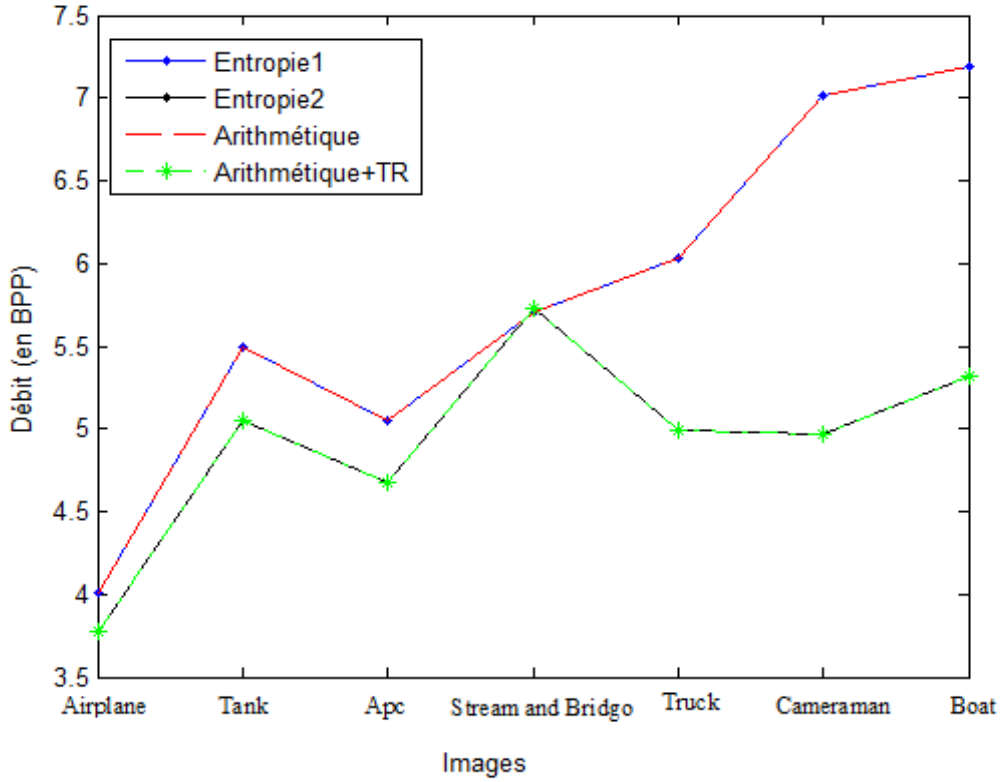


Figure III.4: Comparaison des performances des méthodes: codage Arithmétique (variante A1) et codage Arithmétique dans le domaine transformé (variante A2).

Une première interprétation des résultats obtenus, illustrés aux tableaux III.5 et III.6 et les figure III.3 et III.4 pourrait se fonder sur l'intérêt de la transformée réversible pour la compression sans perte d'images. L'ensemble de ces résultats indique que la transformée présente un potentiel intéressant pour améliorer les performances de compression d'images (sans perte pour le cas considéré dans le cadre de ce projet de fin d'études). Ainsi, à titre d'exemple pour l'image "cameraman", l'entropie et les débits observés pour le codage de Huffman et arithmétique, dans le domaine spatial (H1 et A1) sont respectivement 7.0097 Bpp, 7.0448 Bpp et 7.01 Bpp. A l'aide de l'utilisation de la transformée, dans le schéma de compression, l'entropie et les débits ont diminué respectivement à 4.9715 Bpp, 4.999 Bpp et 4.9718 Bpp. Le même constat peut être fait pour les autres images de test, entre autres, pour l'image Boat, l'entropie et les débits obtenus (de Huffman et arithmétique) avec H1 et A1 a voisinent respectivement {7.1914, 7.2187, 7.1914} Bpp. En employant H2 et A2, on observe une diminution de l'entropie et des débits {5,3195, 5,3441, 5.3196} Bpp respectivement. Par conséquent, l'examen des résultats permet de mettre en évidence des différences significatives observées entre les méthodes de codage appliquées au domaine spatial et celles appliquées au domaine transformé pour toutes les données de test.

III.3.4.2. Comparaison entre les variantes H2 et A2

Cette section est consacrée à établir une comparaison entres les 2 meilleures méthodes de codage, déjà décrites, qui sont appliquées dans le domaine transformé, en l'occurrence H2 (codage de Huffman + transformée réversible) et A2 (codage Arithmétique + transformée réversible). Pour effectuer une comparaison équitable, nous avons utilisé le même ensemble d'images de test, la même ondelette réversible (Haar) et le même niveau de décomposition (6).

			Méthode de codage Huffman+TR		**Méthode de codage Arithmétique + TR**	
Image	**Taille de l'image Originale**	**Entropie**	**Débit**	**Taille de l'image compressée**	**Débit**	**Taille de l'image compressée**
Cameraman	524288 Bit 64 KO	4.9715	4.9990	327614 Bit 39.9919 KO	4.9718	327614 Bit 39.9919 KO
Truck	2097152 Bit 256 KO	4.9862	5.0174	135278 Bit 160.5564 KO	4.9863	1307119 Bit 159.5604 KO
Airplane	2097152 Bit 256 KO	3.7753	3.8114	999136 Bit 121.9648 KO	3.7754	989694 Bit 161.5769 KO
Tank	2097152 Bit 256 KO	5.0492	5.0741	1330134 Bit 162.3699 KO	5.0493	1323638 Bit 161.5769 KO
Apc	2097152 Bit 256 KO	4.6717	4.7060	1233658 Bit 150.5930 KO	4.6718	1224689 Bit 149.4982 KO
Stream and Bridgo	2097152 Bit 256 KO	5.7226	5.7512	1507631 Bit 184.0370 KO	5.7227	1500170 Bit 183.1262 KO
Boat	2097152 Bit 256 KO	5.3195	5.3441	1400912 Bit 171.0098 KO	5.3196	1394502 Bit 170.2273 KO
Débit moyen	/	4.9280	4.9576	/	4.9281	/

Tableau III.7 : Comparaison des performances des méthodes de codage (Huffman et Arithmétique) dans le domaine transformé (variantes H2 et A2)

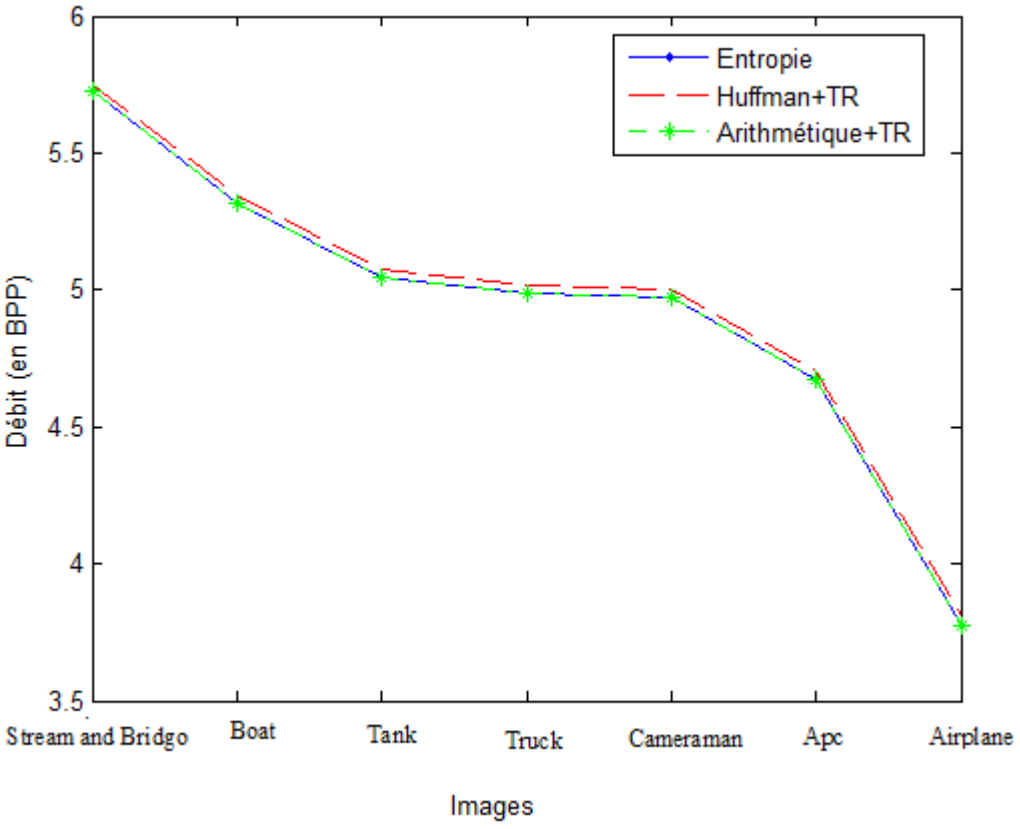


Figure III.5 : Comparaison des performances des méthodes de codage (Huffman et Arithmétique) dans le domaine transformé (variantes H2 et A2).

Comme on peut le constater sur le tableau III.7et figure III.5, A2 assure de faible débits pour toutes les images. En moyenne, le débit avoisine l'entropie. En effet, en augmentant la longueur de la séquence à coder, codage Arithmétique a pour caractéristique de garantir un débit plus proche de l'entropie. Notons qu'un codeur efficace tend à avoir une longueur moyenne (débit dans notre cas) plus similaire à l'entropie. Dans l'ensemble, la méthode du codage Arithmétique s'avèrent efficace. Evidemment, la comparaison avec le codage de Huffman confirme que le codage Arithmétique permet d'obtenir un résultat bien supérieur plus rapidement.

III.4. Conclusion :

Dans ce chapitre, nous avons évalué et comparé les performances de compression sans perte de deux techniques de codage : Huffman (variante H1) et Arithmétique (variante A1) sur des images en niveau de gris.

Les résultats obtenus de quatre variantes, avec les mêmes données de test, ont été analysés en termes de débit exprimé en Bpp, et de complexité de calcul exprimée en temps de codage/décodage (en secondes).

A la lumière de ces résultats, il semble que le codage Arithmétique donne les meilleurs résultats plus rapidement par rapport au codage de Huffman. Ainsi, les performances des variantes A1 et A2 sont toujours supérieures à celles des variantes H1 et H2. Par ailleurs, les résultats de notre étude montrent clairement que le codage dans le domaine transformé présente un grand intérêt pour améliorer les performances de compression d'images, en particulier sans perte (le cas considéré dans ce projet de fin d'études). Cela permet d'avoir de bien meilleurs résultats pour les méthodes de codage qui ont été appliquées au domaine transformé (H2 et A2) vis-à-vis celles qui sont appliquées de façon directe sur l'image originale (H1 et A1).

Conclusion Générale

Conclusion Générale

Les algorithmes de compression de données permettent non seulement de réduire la taille de données mais aussi la charge de traitement et de transmission de données.

Dans ce mémoire, nous avons étudié et comparé les performances de compression sans perte de deux techniques de codage source, couramment utilisées en compression de données, à savoir: le codage de Huffman et le codage Arithmétique sur des images en niveau de gris.

Nous avons présenté quatre variantes possibles des deux techniques : Huffman et Arithmétique pour leurs applications en compression d'images sans perte. Les variantes H1 et A1 concernent respectivement l'application directe des techniques Huffman et Arithmétique sur l'image originale, alors que les autres variantes H2 et A2 sont obtenues en appliquant les deux techniques, mentionnées précédemment, sur l'image transformée par une transformation en ondelettes réversible à coefficients entiers. Cette transformation, qui traite des entiers et retourne des entiers, a pour but d'assurer la mise en œuvre d'un schéma de compression sans perte.

Les résultats de notre étude menée sur les quatre variantes, avec les mêmes données de test, ont été analysés et comparés en termes de débit (en Bpp), et de complexité de calcul (mesure des temps de codage/décodage en secondes).

Sur la base des résultats présentés, le codage Arithmétique est meilleur, plus rapide et plus efficace que le codage de Huffman en termes de débit (en bits par pixel) et du temps de codage et décodage. Ce constat ne vaut pas seulement pour les variantes H1et A1, mais aussi pour les variantes H2 et A2. Ainsi, les variantes du codage Arithmétique: A1 et A2 réalisent toujours la meilleure performance par rapport aux variantes du codage de Huffman: H1 et H2. De plus, les résultats obtenus ont mis en évidence les avantages du codage dans le domaine transformé, qui montre un grand intérêt pour améliorer les performances de compression d'images sans perte. En effet, Les résultats présentés montrent une différence significative dans les débits entre le codage dans le domaine spatial (variantes H1 et A1) et transformé (variantes

H2 et A2). Effectivement, des variantes H2 et A2 améliorent significativement la performance du système de compression en comparaison avec les variantes H1 et A1.

Par ailleurs, comme il a été illustré aux tableaux des résultats, avec l'augmentation de la longueur de la séquence à coder, le codage arithmétique a la caractéristique de garantir un débit plus proche de l'entropie.

Comme perspectives, nous proposons de combiner les deux techniques Huffman et Arithmétique avec d'autres plus performantes, telles les techniques EZW, SPIHT,...etc. pour améliorer d'avantage les performances de compression.

Références Bibliographiques

Bibliographie :

[1] S. Melwin, A. S. Solomon, M. N. Nachappa, ‘’A survey of compression techniques,‘’ 2(1): 152-6, Int. J. Recent. Technol. Eng. , 2013.

[2] D. Mekimah, H. Sadmi, “Compression d’image par la technique EZW, ‘’ Département d’ Electronique, Mémoire de Master, Université de Jijel, 2018.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. l. Cayirci**,** ”A survey on sensor networks,” IEEE Communication Magazine, Vol.40, No. 8, pp 102-116, August 2003.

[4] F. Cabestaing. ‘’Traitement d’images,’’ Option SID. Master AG2I.

[5] S. Larabi. ‘’Systèmes multimédia,’’ Cours: Systèmes Multimédia, Master RSD, USTHB 2013. Disponible à : *perso.usthb.dz/~slarabi/CoursSM/Chapitre%200.pdf*

[6] D. Taubman, M. Marcellin. JPEG2000:’’ Image compression fundamentals standards and Practtice, ‘’ Kluwer Academic Publishers, 2002.

[7] C. Flament, L. Magain, ‘’Cours de conception multimédia, ‘’ Université Libre de Bruxelles, 2006.

[8] S. Ouchraa, H. Badri et Z. Drissi el meliani, ‘’Imagerie numérique,’’ Université Mohamed Rabat, 2010.

[9] S. Renard. ‘’La compression des données,’’ Club Photoshop de Nantes, Conférence du 14 octobre 1999**.** Disponible à : *www.info.univ-angers.fr/~gh/Farcompr/compression_sr.pdf*

[10] L. Clifford, ‘’A Study of the JPEG-2000 image compression standard, ‘’ Queen’s University Canada. Pour l’obtention du titre de Master à Queen’s University, May 2001.

[11] BEAUDOIN, Vincent.’’ Développement de nouvelles techniques de compression de données sans perte ‘’. Mémoire de Maitres en Sciences, Science et Génie, Québec, 2008.

[12] A. J. Maan. ” Analysis and comparison of algorithms for lossless data compression,” 3(3):139-46; Int. J. Inf. Computat. Technol., 2013.

[13] K. Sayood. “Introduction to data compression,” Third Edition, Morgan Kaufman Series in Multimedia Information and Systems, 2017.

[14] Ch. TAOUCHE, ‘’Implémentation d'un environnement parallèle pour la compression d'images à l'aide des fractales,’’ mémoire de Magister, Département d'Informatique, Université Mentouri Constantine, 2005.

[15] C. Benoit & A. Dusson, "La compression de données informatiques,"**.** Juin 1999. http://www.esil.univ-mrs.fr/~cbenoit/projets/comp/

[16] B. Christophe, D. Alexandre. ‘’TIPE sur la compression de données informatiques,’’ MSPI.

[17] G. L. J. Rissanen, ‘’Arithmetic Coding ‘’, IEEE Trans. on communication.

[18] O. Ghazi, A. Khukre, ” Lossy compression using stationary wavelet transform and vector quantization,” Master In Information Technology, Department of Information Technology, Institute of Graduate Studies and Research Alexandria University, 2016.

[19] G. Vijayvargiya, S. Silakari, R. A. Pandey survey: “Various techniques of image compression,” 11(10): 1-5; IJCSIS. 2013.

[20] P. Gupta, G. N. Purohit, V. Bansal .”A survey on image compression techniques,” 3(8): 7762-8; Int. J. Adv. Res. Comput. Commun. Eng. 2014.

[21] V. Jindal, A. K. Verma, S. Bawa. “Impact of compression algorithms on data transmission,” 2(2): 2319-526; Int. J. Adv. Comput. Theory. Eng., 2013.

[22] C. Iombo. Predictive. **“** Data compression using adaptive arithmetic coding,” PhD Thesis. Agricultural and Mechanical College, Louisiana State University; 2007.

[23] W. Sweldens, “The lifting scheme: A custom-design construction of biorthogonal wavelets,” Appl. Comput. Harmon. Anal., 3(2): 186-200, 1996.

[24] A. R. Calderbank, I. Daubechies, W. Sweldens, and B-L. Yeo, “Wavelet transforms that map integers to integers,” Applied and Computation Harmonic Analysis, 5(3): 332-369, July 1998.

[25] F. Davoine, ‘’Compression d'images par fractales basée sur la triangulation de delaunay,’’ Thèse l'INPG, Institut National Polytechnique de Grenoble, Décembre 1995.

[26] Ch. TAOUCHE, ‘’Implémentation d'un environnement parallèle pour la compression d'images à l'aide des fractales,’’ mémoire de Magister, Département d'Informatique, Université Mentouri Constantine, 2005.