Université Paris Sciences et Lettres

Cycle Pluridisciplinaire d'Études Supérieures

La compression JPEG

Mémoire de Recherche

Léonie CROS, Alexandre WERLEN, Julie ZHAN

Encadré par Irène WALDSPURGER

janvier 2025 - juin 2025

Table des matières

1 Introduction

Avec la démocratisation de l'informatique, les photographes, professionnels et amateurs, exploitent de plus en plus le potentiel des ordinateurs pour le traitement d'image, la retouche photo, le stockage, ou encore la visualisation. Néanmoins, stocker une image avec une résolution toujours plus élevée occupe une grande place sur les disques durs des ordinateurs. Prenons l'exemple d'un utilisateur qui conserve 2400 photos sur son téléphone portable. Une image non compressée occupe un espace de stockage d'environ 36 Mo, donc 2400 images non compressées représenteraient environ 86 Go de mémoire sur un appareil [1]. Cet espace consacré aux images peut affecter les performances de l'appareil et le nombre d'applications installables. Un simple algorithme de compression JPEG permet de réduire cet espace à 2,8 Go, soit 30 fois moins. Mais qu'est-ce que la compression JPEG, et pourquoi est-elle aussi efficace ?

En 1990, le JPEG (Joint Photographic Expert Group) réunit une trentaine d'experts pour définir un standard universel d'encodage d'images fixes qui est facilement interprétable. L'algorithme ainsi conçu repose à la fois sur des considérations empiriques, car des photographes décident visuellement ce qui peut être dégradé ou préservé, mais aussi mathématiques. En effet, de nombreux outils issus de l'analyse sont utilisés pour en extraire les éléments clés.

Les techniques utilisées par le standard JPEG s'appuient sur les travaux de Joseph Fourier qui, dans sa théorie éponyme, cherche à résoudre les équations de la chaleur. Ces travaux datant de 1811 permettent de mieux comprendre la décomposition de signaux périodiques et d'en extraire les composantes essentielles. L'idée derrière l'algorithme JPEG est d'appliquer à l'image qui est considérée comme un signal 2D, une transformation de Fourier pour améliorer la manière de représenter les éléments. En effet, cette représentation des données est plus adaptée car elle est plus compacte, ce qui permet de réduire la redondance des informations tout en préservant une qualité visuelle acceptable. Le standard JPEG permet ainsi d'optimiser le stockage et de réduire le coût de transmission des images.

Dans ce mémoire, nous allons étudier en détail l'algorithme JPEG selon trois approches complémentaires, en nous appuyant notamment sur l'article de référence *The JPEG Still Picture Compression Standard* de Gregory K. Wallace [2].

— Une première partie théorique, dans laquelle nous tâcherons de comprendre chaque étape du processus de compression en précisant les notions mathématiques utilisées. Nous nous arrêterons en particulier sur les dérivées de la transformée de Fourier : la transformée de Fourier discrète et la transformée en cosinus discrète.

— Une seconde partie expérimentale consacrée à l'implémentation de la compression JPEG en Python, où nous présenterons le code Python que nous avons réalisé afin d'illustrer les notions théoriques et de réaliser notre propre compression d'images.

— Une troisième et dernière partie, dans laquelle nous introduirons le format de compression JPEG 2000. Celui-ci repose non plus sur une transformée en cosinus discrète comme le format JPEG, mais sur une transformée en ondelettes dont nous présenterons les principales caractéristiques, avantages et inconvénients.

2 Etude de la compression JPEG

2.1 Généralités autour de la compression d'images

2.1.1 Compression avec ou sans perte

L'algorithme JPEG est un algorithme de compression. Il permet de réduire le volume de données nécessaires pour stocker une image en mémoire. Avant de présenter en détail cet algorithme, nous allons nous attarder sur les principales caractéristiques des algorithmes de compression. Il existe deux grandes catégories de compression d'images : la compression avec perte et la compression sans perte. Commençons par définir ces deux méthodes [3].

Définition 2.1 (Compression sans perte)**.** *Cette méthode repose seulement sur un changement de représentation des informations. Il s'agit donc d'une compression réversible, c'est-à-dire qu'il est possible de reconstruire à l'identique l'image originale. En effet, l'intégralité des données d'origine est conservée et la taille du fichier contenant l'image est tout de même réduite. Des formats comme PNG, GIF, BMP et RAW utilisent cette approche.*

Définition 2.2 (Compression avec perte)**.** *La compression avec perte est irréversible. En effet, en plus du changement de représentation, certaines informations non essentielles sont supprimées afin d'optimiser l'espace mémoire. Il existe plusieurs formats qui se basent sur cette méthode, dont JPEG, WebP et HEIF.*

Afin de pouvoir quantifier le degré de compression d'une image après application d'un algorithme de compression, on utilise le taux de compression [4].

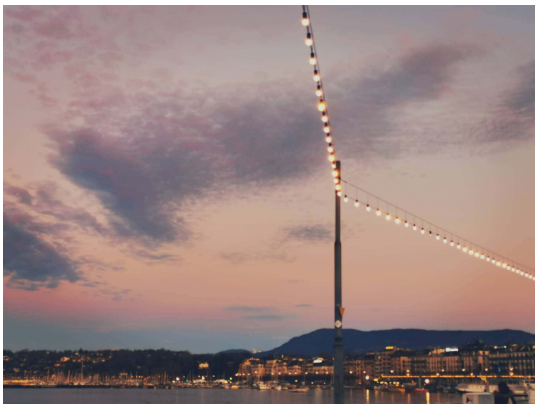
Définition 2.3 (Taux de compression)**.** *Le taux de compression entre une image initiale et son image compressée est défini comme le rapport entre la taille des données initiales nécessaires au stockage de l'image et la taille finale obtenue après compression. Plus ce rapport est élevé, plus la compression est efficace.*

Exemple 2.1. *Le taux de compression pour une compression sans perte est généralement compris entre 2 et 8. Pour une compression avec perte, il varie entre 3 et 100 [5].*

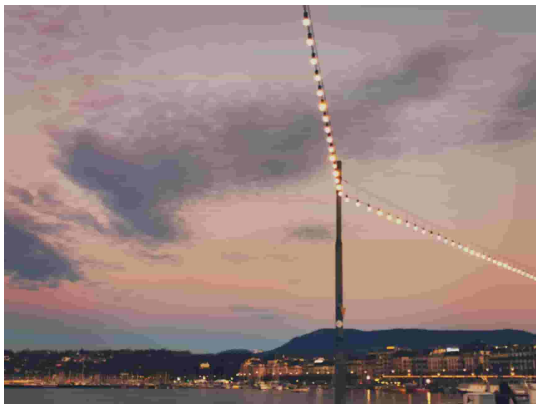
Intuitivement, il semble préférable d'utiliser une compression sans perte. Cependant, dans la plupart des usages quotidiens, nous n'avons pas besoin de voir une image dans sa résolution maximale. Des pertes non perceptibles par l'œil humain sont souvent tolérables, ce qui explique le choix d'une compression avec perte par les formats de compression les plus usuels comme JPEG. Les taux de compression sont alors bien plus élevés. Cependant, il est nécessaire de souligner que bien que la compression maximale soit recherchée, un taux de compression trop élevé nuit à la qualité visuelle de l'image.

2.1.2 Influence du taux de compression

Réalisons une comparaison de la qualité visuelle à différents taux de compression. Ci-dessous sont présentées une image de taille originale 61KB et sa compression à une taille de 19 KB.



(a) Image originale de 61KB



(b) Image compressée à 19KB

FIGURE 1 – Influence du taux de compression sur la qualité visuelle

Nous observons une grande différence de qualité visuelle entre l'image (a) et l'image (b). En effet, l'image (b) présente de larges zones d'aplat de couleur alors que l'image (a), l'image originale non-compressée, présente des dégradés. Ainsi l'image (b) comporte un nombre de nuances beaucoup moins important que l'image (a). Cet exemple illustre l'impact du taux de compression, qui mesure l'efficacité d'un algorithme de compression. Plus le taux de compression est élevé, plus la taille du fichier est réduite, mais au prix d'une possible dégradation visuelle.

2.2 Compression par transformée : notions élémentaires

Le format JPEG combine à la fois changement de représentation des données et suppression des informations non-essentielles à la perception de l'oeil humain. Nous allons dans cette section nous concentrer sur la présentation des outils d'algèbre linéaire utilisés lors du changement de réprésentation des données, étape aussi appelée compression par transformée.

Une image de taille $d \times d$ contient un nombre fini d'informations avec $N = d \times d$ valeurs de pixels. Pour plus de simplicité, cette image est divisée en blocs de taille 8×8 contenant ainsi 64 pixels qui seront traités indépendamment. Chaque bloc 8×8 de l'image est interprétée comme une fonction $f : \{0, ..., 7\} \times \{0, ..., 7\} \to \mathbb{R}$. f associe au pixel de coordonnées (x, y) son intensité. L'ensemble de ces fonctions constitue un espace vectoriel, que nous pouvons munir du produit scalaire suivant :

$$\langle f, g \rangle := \sum_{0 \leq x,y \leq 7} f(x, y)g(x, y).$$

Le principe de compression par transformée repose sur un changement de base : nous allons passer d'une base spatiale à une base fréquentielle. Commençons donc par rappeler quelques notions fondamentales autour des bases.

Définition 2.4 (Base orthogonale)**.** *Une base $(e_n)_{n \in N}$ est dite orthogonale si les vecteurs qui la composent sont orthogonaux deux à deux ie $\langle e_n, e_m \rangle = 0$ pour tous $n \neq m$ appartenant à N.*

Définition 2.5 (Base orthonormale)**.** *Une base $(e_n)_{n\in N}$ est dite orthonormale si elle est orthogonale et $\langle e_n, e_n\rangle = 1$ pour tout n appartenant à N.*

Proposition 2.6. *Formule de décomposition dans une base orthonormale :*

$$f = \sum_{n\in N} c_n e_n$$

avec $c_n = \langle f, e_n\rangle$ et $(e_n)_{n\in N}$ une base orthonormale.

2.3 Transformée en cosinus discrète

Maintenant que nous avons présenté les outils de compression par transformée, étudions le changement de base réalisé par l'algorithme JPEG. Nous noterons $\mathcal{B}_{DCT}$ la base de fonctions utilisée pour le codage JPEG. C'est la base orthonormale pour le produit scalaire introduit en section 2.1 formée par les fonctions $e_{u,v}$, $0 \leq u, v \leq 7$, définies par :

$$e_{u,v}(x,y) := C(u)C(v)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right),$$

où

$$C(u), C(v) = \frac{1}{\sqrt{2}} \quad \text{si } u, v = 0$$
$$C(u), C(v) = 1 \quad \text{sinon.}$$

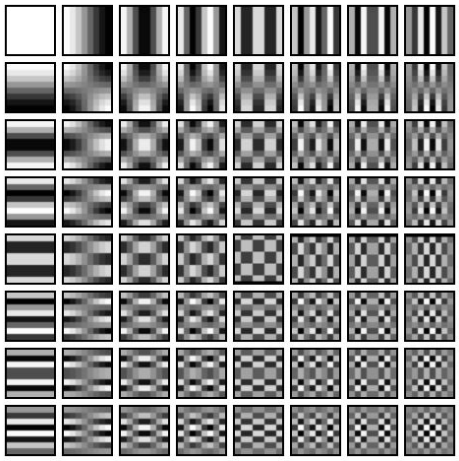


FIGURE 2 – Représentation de la base $\mathcal{B}_{DCT}$, aussi appelée base des cosinus locaux [6]

La fonction f correspondant au bloc à décrire peut alors s'écrire :

$$f = \sum_{0\leq u,v\leq 7} c_{u,v} e_{u,v},$$

où les coefficients $c_{u,v}$ sont les produits scalaires de f avec les fonctions $e_{u,v}$.

Remarque 2.1. *Par soucis de cohérence notationnelle avec les articles sur le sujet [2], nous noterons par la suite les $c_{u,v}$ plutôt que les $F(u,v)$. Rappelons qu'il s'agit des coefficients issus de la décomposition de la fonction f sur la base $\mathcal{B}_{DCT}$.*

La donnée des intensités $f(x,y)$ et celle des coefficients $F(u,v)$ contiennent alors exactement la même information. Mais la donnée des $F(u,v)$ est plus facile à manipuler car elle est exprimée dans une base plus adaptée. Le passage des intensités aux $F(u,v)$ est ce que l'on appelle la transformation en cosinus discrète bidimensionnelle ou DCT bidimensionnelle (*Discrete Cosine Transform* en anglais). Cette transformation ne modifie pas les données mais seulement la manière de les représenter. La formule de la DCT bidimensionnelle est donnée par [2] :

$$F(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)$$

où

$$C(u), C(v) = \frac{1}{\sqrt{2}} \quad \text{si } u, v = 0$$

$$C(u), C(v) = 1 \quad \text{sinon}$$

Il s'agit simplement du produit scalaire de f avec les fonctions $e_{u,v}$.

Remarque 2.2. *La base $\mathcal{B}_{DCT}$ ou base fréquentielle est plus adaptée que la base spatiale, mais nous nous demandons s'il n'existe pas une meilleure base. C'est ce que nous discuterons en section 4 en présentant la méthode des ondelettes.*

Remarque 2.3. *La transformée en cosinus discrète est théoriquement sans perte puisqu'il n'y a pas de modification de l'information. Cependant, le calcul informatique avec l'usage des virgules flottantes se fait à une précision finie, ce qui introduit de nombreuses approximations. C'est néanmoins par la suite que l'on effectue les approximations qui génèrent la plus grande perte d'information.*

2.4 Lien avec la transformée de Fourier discrète

Nous venons d'introduire la transformée en cosinus discrète, mais comment expliquer l'origine d'un tel choix ? Afin d'apporter des éléments de réponse à cette question, nous allons dans cette section aborder le lien entre la transformée de Fourier discrète (TFD) qui, pourrait sembler plus naturelle, et la DCT. Rappelons tout d'abord la définition de la TFD.

Définition 2.7 (Transformée de Fourier discrète)**.** *Soient N un entier non nul et $f : \{0,...,N-1\}^2 \to \mathbb{C}$. On définit la transformée de Fourier discrète $\hat{f}$ par la formule suivante :*

$$\forall (k,l) \in \{0,...,N-1\}^2, \hat{f}(k,l) = \sum_{s,t=0}^{N-1} f(s,t)e^{\frac{-2\pi i(k(s+1/2)+l(t+1/2))}{N}} \in \mathbb{C}$$

Proposition 2.8. *$\mathcal{B}_{TFD1} := \{e_0, ..., e_{N-1}\}$ avec :*

$$e_k : \left\{ \begin{array}{ccc} \{0, ..., N-1\} & \rightarrow & \mathbb{C} \\ s & \mapsto & e^{\frac{-2\pi iks}{N}} \end{array} \right. \tag{1}$$

pour k dans $\{0, ..., N-1\}$ est une base orthogonale de l'espace des fonctions de $\{0, ..., N-1\}$ dans $\mathbb{C}$.

Démonstration. *Pour tous (n,m) dans $\{0, ..., N-1\}^2$ tels que $n \neq m$, nous avons :*

$$\begin{aligned} \langle e_n, e_m \rangle &= \sum_{s=0}^{N-1} e^{\frac{-2\pi i(ns)}{N}} e^{\frac{2\pi i(ms)}{N}} \\ &= \sum_{s=0}^{N-1} e^{\frac{-2\pi i(n-m)s}{N}} \\ &= \frac{1 - e^{\frac{-2\pi i(n-m)N}{N}}}{1 - e^{\frac{-2\pi i(n-m)}{N}}} \\ &= \frac{1 - e^{-2\pi i(n-m)}}{1 - e^{\frac{-2\pi i(n-m)}{N}}} \\ &= 0 \end{aligned}$$

Ainsi, les $\{e_k\}$ forment une famille orthogonale. Un argument de dimension donne le caractère générateur.

La transformée de Fourier discrète permet donc d'obtenir à partir de f exprimée dans la base cartésienne, les coordonnées de f exprimée dans la base orthogonale $\mathcal{B}_{TFD}$. $\mathcal{B}_{TFD}$ étant la base construite à l'aide de $\mathcal{B}_{TFD1}$ avec des fonctions ayant pour espace de départ $\{0, ..., 7\}^2$. Nous avons affirmé dans la partie précédente sur la DCT que $\mathcal{B}_{DCT}$ était une base orthonormale. Ceci découle directement de l'orthogonalité de $\mathcal{B}_{TFD}$ et de l'usage de constantes de normalisation dans la définition de $\mathcal{B}_{DCT}$.

Mais pourquoi la DCT et sa base associée découlent-elles de la TFD et sa base ? Nous allons voir à présent que la DCT est un cas particulier de la TFD.

Définition 2.9. *La transformée discrète de f est une fonction à valeurs complexes que nous pouvons décomposer comme la somme de sa partie réelle et de sa partie imaginaire :*

- *Partie réelle : $\hat{f}_{cos}(k,l) = \sum_{s,t=0}^{N-1} f(s,t) \cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2)}{N}\right)$*
- *Partie imaginaire : $\hat{f}_{sin}(k,l) = -\sum_{s,t=0}^{N-1} f(s,t) \sin\left(\frac{2\pi(k(s+1/2)+l(t+1/2)}{N}\right)$*

La TFD fournit une représentation fréquentielle d'un signal discret, tout comme la DCT. Cependant, la TFD renvoie des valeurs complexes, tandis que la DCT constitue une variante à valeurs réelles qui en découle directement. Pour passer d'une tranformation à valeurs complexes à une version réelle, on a besoin d'appliquer la TFD sur une fonction rendue symétrique, car celle-ci possède des propriétés particulièrement intéressantes pour les fonctions symétriques.

Remarque 2.4. *Pour rendre une fonction plus symétrique, il nous suffit de prolonger la fonction* $f : \{0, ..., 7\} \times \{0, ..., 7\} \to \mathbb{R}$ *en une fonction* $\tilde{f} : \{0, ..., 15\} \times \{0, ..., 15\} \to \mathbb{R}$*, définie par symétrie selon deux axes.*

Concrètement, on impose alors les symétries suivantes :

- *Symétrie horizontale :* $\tilde{f}(x, y) = \tilde{f}(15 - x, y)$
- *Symétrie verticale :* $\tilde{f}(x, y) = \tilde{f}(x, 15 - y)$
- *Symétrie diagonale :* $\tilde{f}(x, y) = \tilde{f}(15 - x, 15 - y)$

Cette construction est illustrée par la figure ci-dessous.

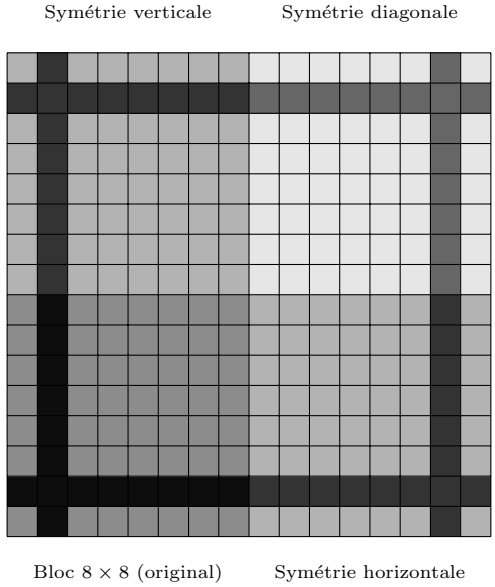


FIGURE 3 – Prolongement symétrique d'un bloc 8×8 en un bloc 16×16

Dans cette figure, le bloc en bas à gauche représente l'image originale 8×8*. Les deux blocs situés en haut à gauche et en bas à droite correspondent respectivement aux symétries verticale et horizontale, et le bloc en haut à droite représente la symétrie diagonale.*

Intéressons-nous maintenant aux propriétés de la transformée de Fourier discrète d'une fonction $g : \{0, ..., 15\} \times \{0, ..., 15\} \to \mathbb{R}$ possédant les symétries horizontale, verticale et diagonale présentées en remarque 2.4.

Proposition 2.10.

$$\forall (k, l) \in \{0, ..., 15\}^2, \overline{\hat{g}(k, l)} = \hat{g}(-k, -l) \tag{2}$$

Démonstration.

$$\begin{aligned}
\overline{\hat{g}(k,l)} &= \sum_{s,t=0}^{15} \overline{g(s,t)} e^{\overline{\frac{-2\pi i(k(s+1/2)+l(t+1/2))}{16}}} \\
&= \sum_{s,t=0}^{15} \overline{g(s,t)} e^{\frac{2\pi i(k(s+1/2)+l(t+1/2))}{16}} \\
&= \sum_{s,t=0}^{15} \overline{g(s,t)} e^{\frac{-2\pi i(-k(s+1/2)-l(t+1/2))}{16}}
\end{aligned}$$

Or, g est réelle, donc $\overline{g} = g$. *On a alors* $\overline{\hat{g}(k,l)} = \hat{g}(-k,-l)$.

Proposition 2.11.

$$\forall (k,l) \in \{0,...,15\}^2, \hat{g}(k,l) = \hat{g}(16-k, 16-l)$$

Démonstration. *Comme g est réelle, par la propriété 2.10 :*

$$\hat{g}(16-k,16-l) = \sum_{s,t=0}^{15} g(s,t) e^{\frac{-2\pi i((16-k)(s+1/2)+(16-l)(t+1/2))}{16}}$$

En utilisant la symétrie diagonale, ie $\hat{g}(x,y) = \hat{g}(16-x, 16-y)$, *nous avons :*

$$\begin{aligned}
\hat{g}(16-k,16-l) &= \sum_{s,t=0}^{15} g(16-s,16-t) e^{\frac{-2\pi i((16-k)(s+1/2)+(16-l)(t+1/2))}{16}} \\
&= \sum_{s,t=0}^{15} g(s,t) e^{\frac{-2\pi i(k(s+1/2)+l(t+1/2))}{16}} \\
&= \hat{g}(k,l)
\end{aligned}$$

Proposition 2.12. $\hat{g}_{sin} = 0$ *donc* $\hat{g} = \hat{g}_{cos}$.

Démonstration.

$$\hat{g}_{\sin}(k,l) = \mathcal{I}m(\hat{g}(k,l))$$

En appliquant la proposition 2.11, nous obtenons :

$$\begin{aligned}
\hat{g}_{\sin}(k,l) &= \mathcal{I}m(\hat{g}(16-k,16-l)) \\
&= \mathcal{I}m\Big(\sum_{s,t=0}^{15} g(s,t) e^{\frac{-2\pi i((16-k)(s+1/2)+(16-l)(t+1/2))}{16}}\Big) \\
&= \mathcal{I}m\Big(\sum_{s,t=0}^{15} g(s,t) e^{\frac{-2\pi i(-k(s+1/2)-l(t+1/2))}{16}}\Big) \\
&= -\mathcal{I}m(\hat{g}(k,l)) \\
&= -\hat{g}_{\sin}(k,l)
\end{aligned}$$

Donc $\hat{g}_{\sin} = 0$

On peut aussi démontrer cette propriété en utilisant la propriété 2.10. En effet, la propriété 2.11 nous donne que $\hat{g}(k,l) = \hat{g}(-k,-l)$ *or* $\hat{g}(-k,-l) = \overline{\hat{g}(k,l)}$. $\hat{g}(k,l)$ *est égal à son conjugué donc est réel.*

Notons $T\tilde{f}$ la transformation qui associe à notre fonction $f : \{0,...,7\}\times\{0,...,7\} \to \mathbb{R}$ la transformée de Fourier discrète de la fonction $\tilde{f} : \{0,...,15\} \times \{0,...,15\} \to \mathbb{R}$ la fonction symétrisée de f. Soit (k,l) dans $\{0,...,7\}^2$. Nous avons donc :

$$
\begin{aligned}
&T\tilde{f}(f)(k,l)\\
&\quad= \hat{\tilde{f}}(k,l)\\
&\quad= \hat{\tilde{f}}_{cos}(k,l)\\
&\quad= \sum_{s,t=0}^{15} \tilde{f}(s,t)\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2))}{16}\right)\\
&\quad= \sum_{s=0}^{7}\sum_{t=0}^{7} \tilde{f}(s,t)\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2))}{16}\right) + \sum_{s=8}^{15}\sum_{t=0}^{7} \tilde{f}(s,t)\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2))}{16}\right)\\
&\quad+ \sum_{s=0}^{7}\sum_{t=8}^{15} \tilde{f}(s,t)\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2))}{16}\right) + \sum_{s=8}^{15}\sum_{t=8}^{15} \tilde{f}(s,t)\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2))}{16}\right)
\end{aligned}
$$

En utilisant les symétries de f telles qu'elles sont détaillées à la remarque 2.4, nous avons :

$$
\begin{aligned}
&T\tilde{f}(f)(k,l)\\
&\quad= \sum_{s=0}^{7}\sum_{t=0}^{7}[\tilde{f}(s,t)\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2))}{16}\right)\\
&\quad+ \tilde{f}(15-(s+8),t)\cos\left(\frac{2\pi(k(s+1/2+8)+l(t+1/2))}{16}\right)\\
&\quad+ \tilde{f}(s,15-(t+8))\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2+8))}{16}\right)\\
&\quad+ \tilde{f}(15-(s+8),15-(t+8))\cos\left(\frac{2\pi(k(s+1/2+8)+l(t+1/2+8))}{16}\right)]
\end{aligned}
$$

Comme f et $\tilde{f}$ coïncident sur $\{0,...,7\}^2$, nous obtenons :

$$\begin{aligned}
&T\tilde{f}(f)(k,l)\\
&\quad= \sum_{s=0}^{7}\sum_{t=0}^{7}[f(s,t)\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2))}{16}\right)\\
&\quad+ f(7-s,t)\cos\left(\frac{2\pi(k(s+1/2+8)+l(t+1/2))}{16}\right)\\
&\quad+ f(s,7-t)\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2+8))}{16}\right)\\
&\quad+ f(7-s,7-t)\cos\left(\frac{2\pi(k(s+1/2+8)+l(t+1/2+8))}{16}\right)]\\
&\quad= \sum_{s=0}^{7}\sum_{t=0}^{7}f(s,t)[\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2))}{16}\right)+\cos\left(\frac{2\pi(k(16-(s+1/2))+l(t+1/2))}{16}\right)\\
&\quad+\cos\left(\frac{2\pi(k(s+1/2)+l(16-(t+1/2)))}{16}\right)+\cos\left(\frac{2\pi(k(16-(s+1/2))+l(16-(t+1/2)))}{16}\right)]
\end{aligned}$$

Ceci en réorganisant les termes de la somme. Par 2π-périodicité du cosinus, nous avons alors :

$$\begin{aligned}
&T\tilde{f}(f)(k,l)\\
&\quad= \sum_{s=0}^{7}\sum_{t=0}^{7}f(s,t)[\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2))}{16}\right)+\cos\left(\frac{2\pi(-k(s+1/2)+l(t+1/2))}{16}\right)\\
&\quad+\cos\left(\frac{2\pi(k(s+1/2)-l(t+1/2))}{16}\right)+\cos\left(\frac{2\pi(-k(s+1/2)-l(t+1/2))}{16}\right)]\\
&\quad= 2\sum_{s=0}^{7}\sum_{t=0}^{7}f(s,t)[\cos\left(\frac{2\pi(k(s+1/2)+l(t+1/2))}{16}\right)+\cos\left(\frac{2\pi(-k(s+1/2)+l(t+1/2))}{16}\right)]
\end{aligned}$$

Par la formule $cos(a+b) = cos(a)cos(b) - sin(a)sin(b)$:

$$T\tilde{f}(f)(k,l) = 4\sum_{s=0}^{7}\sum_{t=0}^{7}f(s,t)\cos\left(\frac{\pi k(2s+1)}{16}\right)\cos\left(\frac{\pi l(2t+1)}{16}\right)$$

A constantes près nous avons égalité entre $T\tilde{f}$ et F, avec F la transformation en cosinus discrète définie en section 2.3. Ainsi, la DCT est la version réelle de la TFD appliquée à une fonction prolongée par symétrie.

2.5 Processus de compression JPEG

Dans cette section nous allons détailler les étapes de la compression JPEG.

2.5.1 Application de la DCT

L'image à compresser est découpée en blocs de 8 pixels par 8 pixels. Chaque bloc est représenté par une matrice. Les blocs subissent une DCT bidimensionnelle qui, comme vu dans la section précédente, convertit le bloc du domaine spatial au domaine fréquentiel. Mais quel est l'intérêt d'un tel changement de base ? Pour répondre à cette question, observons les effets de la DCT sur la matrice des données :

— La première valeur de la matrice de coordonnées (0,0) est la plus élevée. On l'appelle composante DC (direct current) ou composante continue. Elle représente le signal continu du bloc de pixels, dans notre cas la moyenne de luminosité.

— Les 63 autres coefficients sont notés composantes AC (alternative current) et représentent les amplitudes des fréquences spatiales du bloc 8×8 de l'image.

— Les plus grandes valeurs sont regroupées dans le coin en haut à gauche de la matrice. Il s'agit des coefficients de faibles fréquences.

Ces observations nous permettent de mettre en lumière les intérêts d'un tel changement de base. JPEG est un format adapté aux images naturelles. Or, les images naturelles contiennent généralement beaucoup d'aplats locaux et peu de bordures, ce qui se traduit par une concentration de l'information dans les basses fréquences. De plus, l'œil humain est beaucoup plus sensible à ces basses fréquences plutôt qu'aux hautes fréquences correspondant aux détails. L'objectif de cette tranformation est donc de concentrer les informations visuelles essentielles dans un petit nombre de coefficients situés dans la même zone de notre matrice.

Remarque 2.5. *Les coefficients obtenus après DCT peuvent dépasser 255 ou être négatifs, il est donc nécessaire de passer d'un encodage sur 8 bits à une profondeur d'encodage plus élevée (16 bits en général).*

2.5.2 Quantification

Nous allons à présent diviser chacun des coefficients de la matrice obtenue après DCT par une constante. L'ensemble de ces coefficients, la matrice de quantification notée $Q(u, v)$, est choisie empiriquement de façon à s'adapter à la vision humaine. Les constantes dans la zone des basses fréquences (celles en haut à gauche de la matrice) sont plus faibles afin de conserver les informations importantes pour notre système visuel. En réduisant l'impact des hautes fréquences, cette étape introduit une perte d'information.

On note la formule de quantification comme suit : La division est une division terme a terme ou une division d'Hadamard

$$F^Q(u, v) = \text{round}\left(\frac{F(u, v)}{Q(u, v)}\right)$$

Voici une matrice typique pour une table de quantification standard pour la luminance. C'est celle que nous utiliserons principalement pour notre code [**ipol.2022.399**] :

$$Q = \begin{bmatrix} 1 & 2 & 2 & 3 & 5 & 8 & 10 & 12 \\ 2 & 2 & 3 & 4 & 5 & 12 & 12 & 11 \\ 3 & 3 & 3 & 5 & 8 & 11 & 14 & 11 \\ 3 & 3 & 4 & 6 & 10 & 17 & 16 & 12 \\ 4 & 4 & 7 & 11 & 14 & 22 & 21 & 15 \\ 5 & 7 & 11 & 13 & 16 & 21 & 23 & 18 \\ 10 & 13 & 16 & 17 & 21 & 24 & 24 & 20 \\ 14 & 18 & 19 & 20 & 22 & 20 & 21 & 20 \end{bmatrix}$$

2.5.3 Arrondi

Nous arrondissons ensuite tous les coefficients de la matrice afin de ne travailler qu'avec des nombres entiers. De nombreux coefficients deviennent alors nuls. Si on regarde la moitié inférieure droite des matrices, on constate que les coefficients DCT étaient déjà assez petits et qu'ils ont été divisés par des coefficients de quantification relativement élevés.

Remarque 2.6. *La valeur la plus élevée dans la matrice restant relativement faible, on peut donc à nouveau stocker les coefficients quantifiés avec une profondeur de 8 bits.*

2.5.4 Réorganisation en zigzag

Après la quantification, les coefficients correspondant aux hautes fréquences deviennent nuls. L'encodage en zigzag permet d'ordonner les coefficients du plus significatif au moins significatif pour regrouper les valeurs nulles ensemble. Le parcours à suivre est illustré par la figure 4. Cette réorganisation facilite le codage entropique car nous pouvons représenter la suite de valeurs nulles par une seule notation de type [N,0].

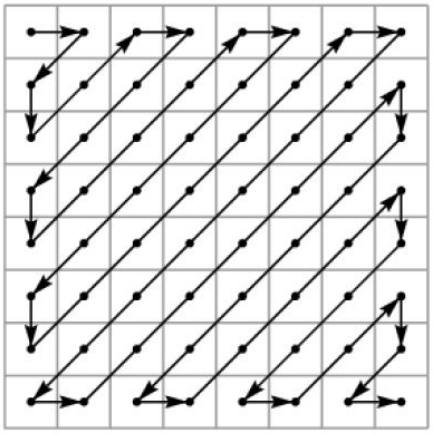


FIGURE 4 – Parcours en zigzag d'un bloc 8x8 [7]

2.5.5 Run-Length Encoding (RLE)

C'est avec le RLE et le codage Huffman que la compression de l'image a véritablement lieu. Dans un premier temps, on procède à un simple codage par substitution. En effet, le processus de quantification réduit à 0 les plus faibles coefficients au lieu de tous les conserver. Nous remplaçons alors les suites de zéros adjacents par un caractère spécial puis le nombre de 0 de la suite. On procède ainsi sur chacune des matrices 8×8 de l'image. On peut ensuite concaténer toutes les subdivisions pour obtenir une longue chaîne de caractères.
Par exemple, considérons la suite de données suivante :

$$[8, 0, 0, 1, 0, 0, 0, 0]$$

Après encodage RLE, on obtient :

$$[8, (0, 2), 1, (0, 4)]$$

2.5.6 Codage Huffman

La dernière étape de la compression est le codage Huffman. Étant donné un alphabet quelconque, il est clair que certains éléments sont plus fréquents que d'autres. Ainsi, lors de la compression on voudra associer une chaîne de bits plus courte pour des caractères fréquents et une chaîne plus longue pour les caractères plus rares, minimisant ainsi la taille de la transmission. Le codage Huffman est la méthode optimale pour résoudre ce type de problème. Comme sa preuve et son fonctionnement sont relativement complexes et que ce n'est pas le sujet du mémoire, nous admettons simplement son optimalité et ne détaillons pas davantage son fonctionnement. Ainsi nous obtenons une chaîne de bits la plus courte possible et un dictionnaire qui permet d'associer à chaque chaine de bits vers son caractère d'origine.

Remarque 2.7. *L'encodage RLE et Huffman sont des algorithmes lossless, c'est-à-dire qu'ils n'introduisent pas de perte de données.*

Remarque 2.8. *L'algorithme JPEG standard utilise un dérivé de l'algorithme Huffman un peu plus efficace pour des raisons de simplicité nous avons utilisé l'approche générale.*

2.6 Décompression

Pour reconstruire l'image originale, on suit l'inverse des étapes de compression :

— Le **décodage entropique** : On applique d'abord le décodage de Huffman pour retrouver la séquence initiale de coefficients DCT, puis le décodage RLE et un zigzag inverse pour reconstruire la matrice 8×8 avec ses zéros replacés.

— La **déquantification** : Pour récupérer une approximation des coefficients DCT, on utilise la relation inverse de la quantification :

$$F^{Q'}(u, v) = F^{Q}(u, v) * Q(u, v)$$

La déquantification est réalisée en multipliant les coefficients terme à terme, * représente donc un produit d'Hadamard. Cette étape permet de restaurer la dimension des coefficients comprimés à l'étape de la quantification.

— L'**inverse DCT (IDCT)** : Un changement de base étant une opération bijective, il existe une fonction inverse pour la DCT. Celle-ci est donnée par la formule :

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)$$

où

$$C(u), C(v) = \frac{1}{\sqrt{2}} \quad \text{si } u, v = 0$$

$$C(u), C(v) = 1 \quad \text{sinon}$$

L'application de l'IDCT permet le retour dans le domaine spatial et la reconstruction de l'image en réassemblant tous les blocs transformés. Algébriquement, cette étape s'apparente à une multiplication par la matrice inverse. Cependant, les pertes introduites par la quantification sont irréversibles, ce qui explique les artefacts visibles dans une image fortement compressée.

3 Analyse expérimentale de la compression JPEG

Dans cette section, nous allons présenter notre implémentation du processus de compression JPEG et les résultats expérimentaux. Nous avons choisi d'implémenter la compression JPEG en utilisant le paradigme de la programmation orientée objet en Python. Cette approche permet une structuration claire du code et facilite la manipulation des images en blocs distincts.

Le code est accessible au Repository GitHub suivant : https ://github.com/22werlal/JPEG-test.git

Dans notre code, deux classes principales sont utilisées : Image et Bloc 8x8.

3.1 Classe Image

La classe Image est responsable de la gestion de l'image entière et de sa division en blocs de 8×8. Chaque bloc sera traité séparément pour appliquer la compression JPEG.

3.1.1 Attributs

Nous avons défini plusieurs attributs pour cette classe. A l'image à traiter est affecté le type Image, qui dispose de plusieurs attributs :
— contenu : Contient le tableau numpy des valeurs de l'image (matrice de pixels)
— lenx : Longueur des lignes de contenu
— leny : Longueur des colonnes de contenu
— blocs : Liste des blocs de 8 par 8 de type Bloc8x8 de contenu

3.1.2 Découpage de l'image

Cette classe possède plusieurs méthodes. La plus importante est **division** qui permet de diviser l'image à traiter en blocs de 8 par 8. Cette méthode peut aussi s'appliquer quand les dimensions de l'image à traiter ne sont pas divisibles par 8. Dans ce cas-là, les blocs incomplets sont complétés en répétant les dernières lignes et colonnes pour garantir une structure correcte.

3.2 Classe Bloc8x8

La classe Bloc8x8 est utilisée pour représenter chaque bloc de 8×8 pixels et appliquer les différents étapes de la compresssion JPEG.

3.2.1 Attributs

Chaque bloc contient plusieurs attributs, permettant de suivre les transformations successives de la compression. Commençons tout d'abord par présenter quelques-uns de ses attributs :

- contenu : Contient le tableau numpy des valeurs du bloc
- comp : Image compilée, compressée après l'étape de DCT et de quantification
- DCT : Coefficients obtenus après la DCT
- line : Coefficients réorganisés selon l'ordre zigzag
- RLE : Image compressée avec le codage Run-Length Encoding
- erreur : Erreur de compression entre l'image originale et l'image compressée

3.2.2 Méthodes

Passons maintenant aux méthodes.

- Tout d'abord, la méthode `check_matrice` permet de vérifier si le bloc est bien une matrice de 8x8 et si les valeurs des pixels sont comprises entre 0 et 255.

- La méthode `show` est utilisée pour afficher l'image originale et l'image compressée afin de comparer la qualité de compression.

- Dans la méthode `calcul`, nous appliquons la Transformée en Cosinus Discrète bidimensionnelle puis la quantification à chaque bloc de 8x8. Pour cela, nous utilisons la fonction `dct` du module `scipy.fftpack` qui permet d'appliquer la DCT-II efficacement en mode orthonormé (`norm='ortho'`). La formule s'explique par le fait qu'une DCT-2D peut être décomposée en deux DCT-1D successives (appliquant sur les lignes et colonnes), et la normalisation permet de s'assurer que la transformation est orthonormée afin d'éviter les distorsions d'amplitude et d'avoir une transformation réversible correcte. Ensuite, la formule de la quantification est appliquée pour réduire les valeurs des hautes fréquences qui sont moins perceptibles. On arrondit les valeurs avec `round` et les convertit en entier en utilisant `astype`.

- Les coefficients DCT sont réorganisés dans la méthode `zigzag`.

- Dans la méthode `RLE`, on regroupe les séries des coefficients nuls pour compresser efficacement la représentation.

- `inverse_calcul` correspond au processus de décompression où on applique la formule de déquantification puis l'inverse DCT grâce à la fonction **idct** issue du même module que `dct`. La déquantification multiplie chaque coefficient DCT par la matrice de quantification Q.

- Dans `score`, on évalue l'erreur moyenne absolue(MAE) entre le bloc compressé(`comp`) et original(`contenu`). L'erreur moyenne absolue représente la moyenne des différences absolues entre les valeurs de pixels d'origine et les valeurs des pixels compressés. Elle est définie par :

$$MAE = \frac{1}{64} \sum_{i=0}^{7} \sum_{j=0}^{7} |\text{comp}[i,j] - \text{contenu}[i,j]|$$

où dans un bloc 8×8, nous avons $contenu[i, j]$ la valeur d'un pixel dans le bloc original et $comp[i, j]$ la valeur d'un pixel dans le bloc compressé après l'étape de décompression.

3.3 Résultats expérimentaux

3.3.1 Principaux résultats et méthode de génération

Afin d'évaluer l'efficacité de notre code, nous avons sélectionné un ensemble de douze images tests, libre de droit. Le choix de ces images a été guidé par une volonté de couvrir un large éventail de cas d'usage. Ce panel est constitué de photographies classiques, de paysages, de portraits ainsi que de gros plans afin d'être représentatif des diverses situations que l'on peut rencontrer. Nous avons également testé les cas limites, dans lesquels le processus JPEG est moins approprié. C'est ainsi que nous avons sélectionné une page de texte, des gravures, et un dessin vectoriel.

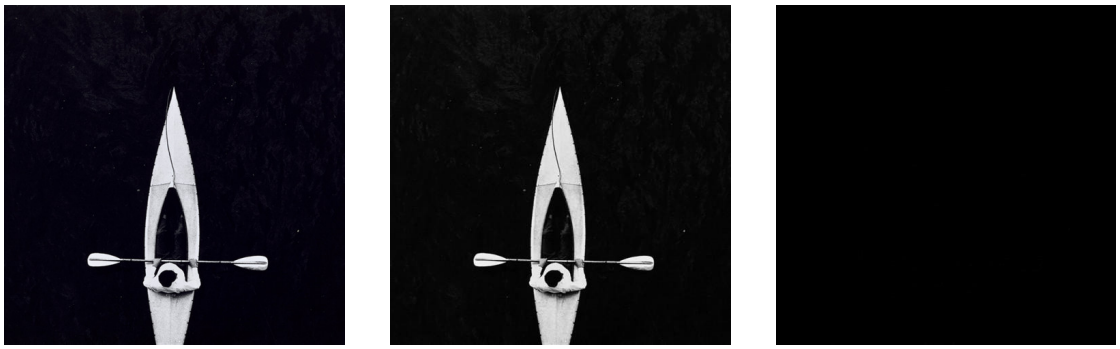
Tout d'abord, afin d'avoir une idée générale du bon fonctionnement de notre algorithme, nous avons regroupé dans le tableau 1 trois indicateurs statistiques : le taux de compression moyen, l'erreur moyenne et la variance. Ces trois indicateurs sont obtenus en effectuant la moyenne des informations issues de notre algorithme de compression appliqué à chacune des douze images de notre panel. Nous avons effectué l'expérience avec trois matrices de quantification différentes correspondant à des taux de compression plus ou moins faibles [8].

TABLE 1 – Taux de compression en fonction de la matrice de quantification utilisée

Matrice de quantification	Taux de compression moyen	Erreur moyenne	Variance
faible	2.43	0.31	0.2
intermédiaire	3.2	0.94	3
forte	5.2	6.5	171

3.3.2 Résultats qualitatifs

Nous allons dans cette section présenter quelques exemples de résultats obtenus grâce à notre programme.



(a) Image originale (2000Kb) (b) Image compressée 342Kb (c) Erreur résiduelle (0.258/px en moyenne)

FIGURE 5 – Compression par DCT cas favorable

Commençons par un cas particulièrement favorable. Pour cela, nous avons choisi une image représentant un kayakiste. Cette image est constituée de grandes zones de couleur homogènes et de transitions douces entre les différents niveaux de luminosité. Le résultat de la compression est présenté à la figure 5. A l'oeil nu, aucune différence notable ne peut être observée entre l'image originale et l'image compressée. De plus, la matrice d'erreur résiduelle ne révèle aucun artefact de compression.

(a) Image originale (2000Kb) (b) Image compressée 342Kb (c) Erreur résiduelle (0.395/px en moyenne)

FIGURE 6 – Compression par DCT cas d'enchevêtrement

Étudions maintenant deux cas où la DCT donne de moins bons résultats. Pour accentuer les effets de la quantification, nous utiliserons notre matrice de quantification la plus forte, multipliée par 2. La première situation où l'on remarque une perte visuelle de la qualité est le cas de zones pour lesquelles les couleurs sont particulièrement enchevêtrées. Nous avons choisi l'exemple d'un arbre de couleur sombre se détachant sur un ciel clair. Comme nous pouvons voir à la figure 6, les branches paraissent relativement nettes sur l'image originale. Au contaire, elles semblent beaucoup plus floues sur l'image comprimée. Cette impression est corroborée par la matrice d'erreur qui témoigne d'erreurs de compression notables au voisinage de l'arbre.

(a) Image originale (b) Image compressée 342Kb (c) Erreur résiduelle (4.2/px en moyenne)

FIGURE 7 – Compression par DCT cas défavorable contraste abrupt

Un autre cas un peu litigieux est le texte. En effet, la séparation brutale très fine entre les pixels blancs et les pixels noirs est une situation particulièrement inadaptée à la compression par DCT. Après application de l'algorithme, le texte est plus flou et moins lisible à distance. De plus, la division par bloc de 8 pixels de la DCT crée à proximité du texte une impression de rupture avec les zones blanches. Ces zones du texte étant beaucoup plus homogènes, elles conservent leur qualité visuelle.

3.3.3 Résultats quantitatifs

Nous allons maintenant présenter quelques résultats mettant en évidence plusieurs éléments clés liés à la DCT et à l'agorithme JPEG, notamment l'influence de la matrice de quantification sur le taux de compression.

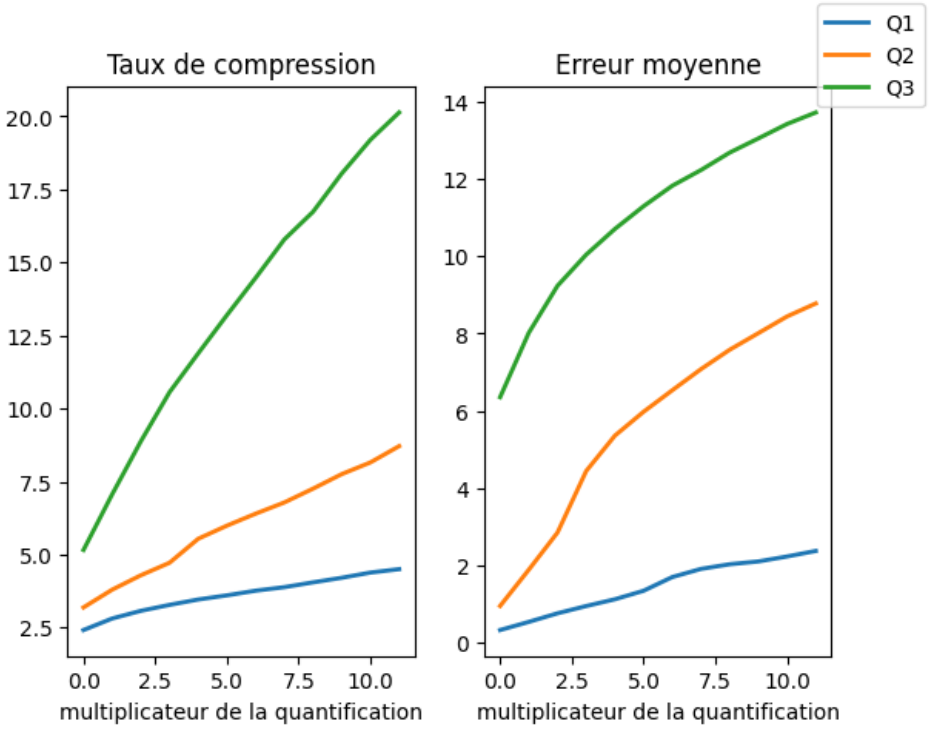


FIGURE 8 – Evolution du taux de compression et de l'erreur moyenne des trois matrices de quantification en fonction du multiplicateur de quantification

Sur la figure 8, les différentes matrices de quantification vues à la table 1 sont notées Q1, Q2 et Q3, correspondant respectivement à une matrice de quantification faible, intermédiaire, et forte. Ce sont des matrices standard JPEG. La matrice Q3 a des coefficients plus élevés que Q2 donc on obtient un meilleur taux de compression quand on l'applique. Cependant, l'erreur résiduelle est alors plus grande. Q2 a quant à elle des coefficients plus élevés que Q1. Pour obtenir un plus large panel de matrices de quantification, nous avons multiplié Q1, Q2 et Q3 par un coefficient entre 1 et 10 appelé multiplicateur de la quantification. Dans la figure 8, on affiche l'évolution du taux de compression et de l'erreur moyenne en fonction du multiplicateur de la quantification utilisé et ce pour Q1, Q2 et Q3. Cela permet d'observer l'amélioration du taux de compression et la dégradation de la qualité.

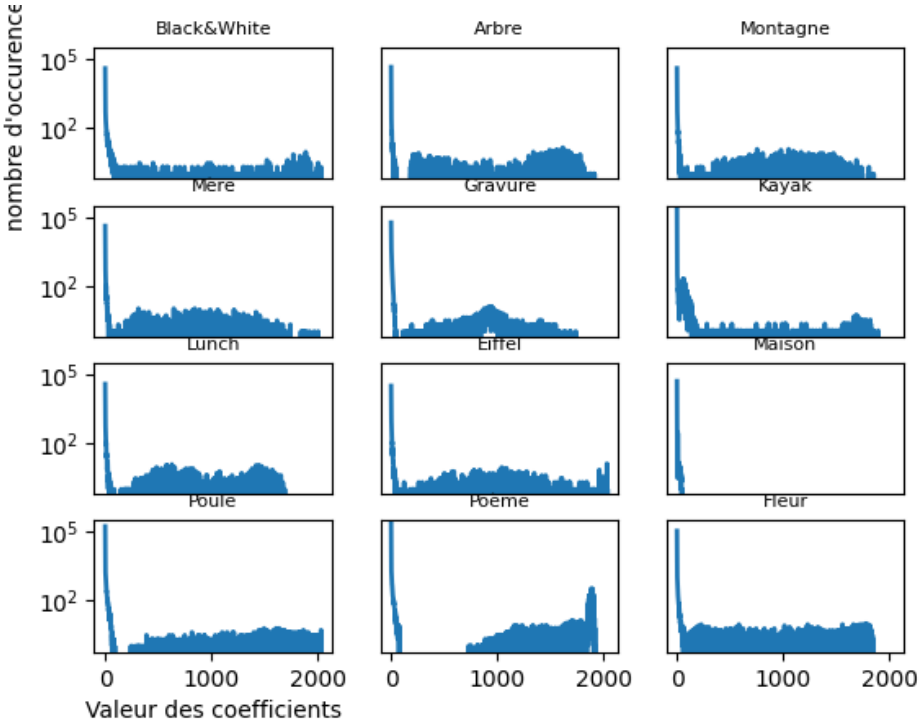


FIGURE 9 – Distribution des coefficients DCT pour 12 images test

Les deux figures suivantes représentent la distribution des coefficients de la DCT pour chaque image après l'étape de déquantification. Ces coefficients sont des entiers. Pour chaque image on trie les coefficients et pour chaque valeur on représente le nombre d'occurence de cette valeur en ordonnée sur une échelle logarithmique.
Dans la cas des images bien compressées, on observe un pic marqué au niveau du coefficient nul, suivi d'une distribution concentrée autour des coefficients de faible intensité. A l'inverse, pour les images qui se compressent mal, on constate un pic au niveau des hautes fréquences, qui représente généralement les artefacts visuels. C'est notamment le cas des images «poème» et «arbre» illustrées ci-dessous.

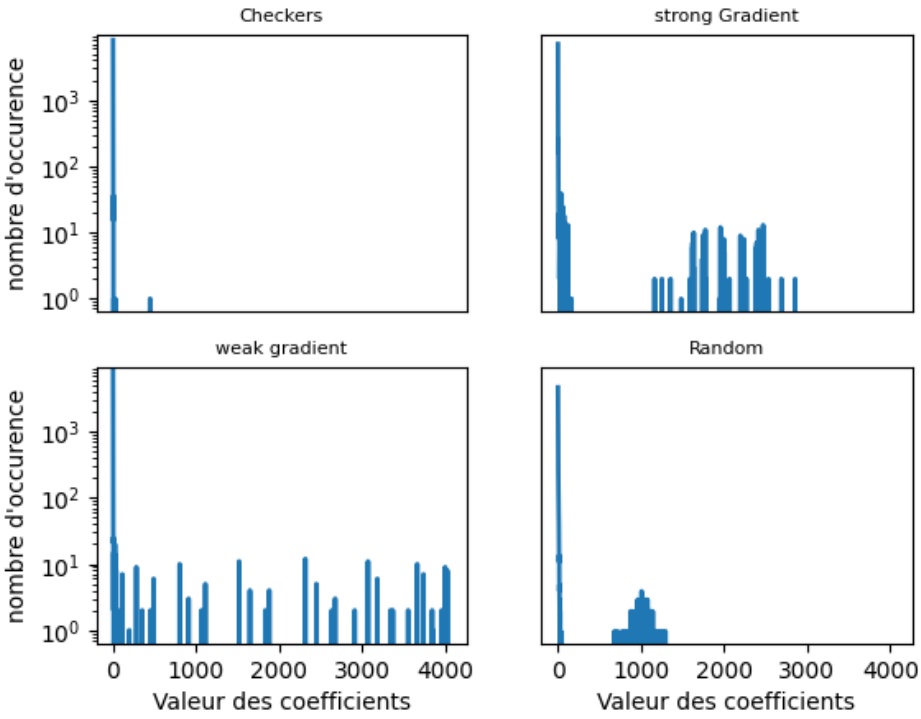


FIGURE 10 – Distribution des coefficients DCT pour 4 images synthétiques

Pour compléter ces observations, nous avons mené la même analyse sur quatre images synthétiques : un damier, un gradient faible, un gradient fort et une image aléatoire. Nous pouvons constater que le damier est efficacement représenté par quelques coefficients DCT. Le gradient fort nécessite davantage de coefficients intermédiaires, tandis que le grandient faible requiert des coefficients répartis sur l'ensemble des fréquences. Enfin, l'image aléatoire est bien résumée par quelques coefficients. Nous sommes assez surpris par ce résultat. Cependant nous pouvons peut-être faire l'hypothèse que comme la compression a été effectuée avec la matrice de quantification la plus importante, les différences les plus subtiles sont gommées. On constate d'ailleurs une erreur moyenne plus importante que pour les gradients.

4 Compression par ondelettes et JPEG 2000

Nous avons vu dans la section 2 que dans la compression JPEG, le changement de base choisi était celui de la transformée en cosinus discrète. Cependant, nous nous demandons s'il n'existe pas des changements de base plus optimaux. Ainsi, dans cette partie, nous allons rapidement présenter un autre type de compression : la compression par ondelettes.

Le principe de la compression par ondelettes est similaire à celle de la DCT : il s'agit de transformer l'image dans un domaine où l'information est concentrée sur peu de coefficients, pour ensuite quantifier ceux qui sont négligeables et parvenir à réduire le volume occupé par l'information contenue dans l'image.

4.1 Transformée en ondelettes discrète

Contrairement à la DCT qui repose sur la représentation d'un signal à l'aide de fonctions sinusoïdales, la DWT (*Discrete Wavelet Transform*) se base sur des fonctions oscillantes appelées ondelettes, qui sont localisées à la fois dans l'espace et en fréquence. Cette propriété des ondelettes permet une meilleure adaptation aux variations locales du signal, autrement dit aux zones détaillées des images. De plus, la DWT est par défaut appliquée non pas sur un bloc de taille 8×8, mais sur l'ensemble de l'image, ce qui évite les artefacts et les discontinuités que l'on observe souvent dans une compression JPEG. Il est aussi possible de découper l'image en pavés pour diminuer le temps de calcul, mais on retrouve alors les conséquences visuelles présentes dans la compression JPEG sur les bornes des pavés.

Dans une DWT 1D, le signal est traité par une paire de filtres : un filtre passe-bas qui conserve les basses fréquences en fournissant une version lissée et floue du signal, et un filtre passe-haut qui, quant à lui, conserve les hautes fréquences en retenant les détails.

Dans le cas d'une compression d'image, nous appliquons la DWT 1D sur les lignes et les colonnes pour obtenir une DWT 2D, et l'image est cette fois-ci séparée en 4 sous-bandes, où L représente *low frequences* (basses fréquences) et H représente *high frequences* (hautes fréquences) [9] :

— LL : basses fréquences dans les deux directions (zone lissée)
— LH : basses fréquences horizontalement et haute fréquence verticalement (contours verticaux)
— HL : hautes fréquences horizontalement et basse fréquence verticalement (contours horizontaux)
— HH : hautes fréquences dans les deux directions (détails fins)

Cette décomposition est itérative : on peut réappliquer la DWT sur le sous-bloc LL, ce qui découpe cette zone en 4 nouvelles sous-bandes, et ainsi de suite. Chaque niveau apporte alors une précision supplémentaire sur les détails de plus en plus fins, ce qui nous permet d'avoir une représentation multi-résolution de l'image. Ainsi, nous pouvons reconstruire l'image à des niveaux de résolution différents, facilitant l'affichage progressif de l'image, qui est généralement utilisé sur les pages web où l'image se charge petit à petit.

4.2 Avantages et limites

Entre 1997 et 2000, le *Joint Expert Photographic Group* a élaboré un nouvel algorithme de compression basé sur la DWT : JPEG 2000. Ce format prend en charge à la fois la compression avec perte et sans perte, et possède plusieurs avantages [10] :

— l'efficacité de la compression est améliorée
— la résilience à l'erreur est plus importante
— le format est plus flexible
— la compression sans perte est meilleure
— il est possible de passer d'une compression avec pertes à une compression sans perte
— la représentation à plusieurs résolutions permet un affichage progressif

Cependant, malgré ces avantages, JPEG 2000 est moins utilisé en pratique que le JPEG classique. Plusieurs raisons expliquent cela[11] :

— la complexité à mettre en oeuvre : JPEG 2000 n'est pas rétrocompatible et nécessite plus de RAM que le JPEG
— l'incompatibilité logicielle : contrairement au format JPEG classique qui est omniprésent, JPEG 2000 n'a pas été pris en compte par tous les navigateurs, ce qui ne le rend pas accessible à tous les utilisateurs
— le manque de besoin grand public pour les fonctionnalités avancées comme la compression sans perte ou la représentation à multi-résolution

En effet, les avantages de JPEG 2000 répondent principalement à des besoins professionnels ou spécifiques, notamment dans les domaines de la médecine, du cinéma numérique ou de l'imagerie satellitaire. Pour un utilisateur amateur, les fonctionnalités offertes par le format JPEG classique suffisent, tant en qualité qu'en compatibilité.

Remarque 4.1. *La DCT peut être vue comme un cas particulier de transformée en ondelettes, où les ondelettes sont des fonctions cosinus tronquées définies par :*

$$\psi_k(x) = \cos\left(\frac{k\pi x}{N}\right) \mathbb{1}_{\{0 \leq x < N\}}$$

Ces fonctions sont bien localisées dans le domaine fréquentiel, mais pas dans le domaine spatial car les fonctions cosinus sont définies globalement (non nulles) sur tout le bloc et ne peuvent pas s'adapter aux variations locales du signal. Or, comme vu précédemment, les ondelettes sont localisées à la fois dans le domaine fréquentiel et spatial, c'est pourquoi la DWT permet souvent une meilleure compression que la DCT dans les images complexes.

5 Conclusion

Au cours de notre travail, nous avons pu découvrir comment était construit l'algorithme JPEG pour la compression des images. Nous avons dans la section 2 étudié l'aspect théorique en tâchant d'explorer les liens entre compression JPEG et analyse de Fourier. Puis afin d'illustrer cela, nous avons en section 3 brièvement présenté notre propre code de la compression JPEG. Cette phase expérimentale nous montre les effets des différents paramètres lors de la compression sur la qualité des images. Le code complet est disponible via le lien fourni dans l'introduction de la section 3 de ce mémoire. Enfin, nous avons élargi notre étude à un autre type de compression proposé par le standard JPEG : la compression par ondelettes. Celle-ci repose sur des outils mathématiques différents et offre des performances supérieures dans certains cas, notamment pour les images à forts contrastes ou les vidéos.

Ainsi, ce travail nous a permis de mieux comprendre comment les mathématiques, et en particulier l'analyse fonctionnelle et numérique, interviennent concrètement dans des applications du quotidien. Dans un monde où le volume d'images échangées et stockées ne cesse de croître, la compression reste un enjeu majeur tant du point de vue théorique que pratique.

Références

[1] *Combien de photos pouvez-vous vraiment sauvegarder sur 128 Go ?* URL : `https://www.harrypotterforever.fr/128-go-combien-de-photos/`.

[2] Gregory K. Wallace. "The JPEG Still Picture Compression Standard". In : 38.1 (February 1992), p. xviii-xxxiv.

[3] *Compression de données*. URL : https://fr.wikipedia.org/wiki/Compression_de_donn%C3%A9es.

[4] *Taux de compression de données*. URL : https://fr.wikipedia.org/wiki/Taux_de_compression_de_donn%C3%A9es.

[5] *JPEG*. URL : https://fr.wikipedia.org/wiki/JPEG.

[6] *La compression JPEG et la DCT*. URL : https://sorciersdesalem.math.cnrs.fr/JPEG/jpeg-DCT.html.

[7] *Cours/TD 4 Compression par transformée. Codage JPEG*. URL : https://pageperso.lis-lab.fr/andreea.dragut/enseignementCLAA/CoursTD4_Fin.pdf.

[8] Tina NIKOUKHAH et al. "A Reliable JPEG Quantization Table Estimator". In : *Image Processing On Line* 12 (2022), p. 173-197. DOI : https://doi.org/10.5201/ipol.2022.399.

[9] Rajan JOSHI. "An overview of the JPEG 2000 still image compression standard". In : (2002). DOI : https://doi.org/10.1016/S0923-5965(01)00024-8.

[10] *JPEG 2000*. URL : https://fr.wikipedia.org/wiki/JPEG_2000.

[11] *Format JPEG 2000*. URL : https://www.adobe.com/fr/creativecloud/file-types/image/raster/jpeg-2000-file.html.