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

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.

Ce travail est une adaptation de notre mémoire de fin de licence. Nous allons avant tout aborder ici l'aspect théorique de la compression JPEG en instistant sur les notions mathématiques utilisées. Notre objectif est de rendre accessible à tout étudiant de licence 3 les dessous de cet algorithme de compression. En particulier, nous souhaitons faire découvrir au lecteur le lien entre le changement de représentation des données utilisé par le standard JPEG et la transformée de Fourier telle qu'elle est étudiée en cours de licence. Mais ce travail ne s'arrête pas à la théorie car tout l'intérêt de ce sujet est de proposer notre propre version de l'algorithme JPEG afin de donner une illustration concrète. Certes, le code que nous allons vous proposer n'est pas optimal, mais nous espérons qu'il remplira sa fonction première : permettre une compréhension avancée du lecteur en laissant le moins possible de zones d'ombre.

Ce code se veut accessible grâce à de multiples commentaires et propose aussi pour aller plus loin une analyse expérimentale. Vous pourrez le retrouver en cliquant sur le lien au bas de ce pragraphe, accompagné d'une fiche explicative.

https://github.com/leoniecrosgardes/La-compression-JPEG-analyse-experimentale

2 Rappels autour des changements de base

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. Chaque bloc sera traité indépendamment et interprété comme une fonction $f : \{0, ..., 7\} \times \{0, ..., 7\} \to \mathbb{R}$. La fonction 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 2.1 (Base orthogonale)**.** *Une base $(e_n)_{n\in\{0,...,N-1\}}$ 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 {0,...,N-1}.*

Définition 2.2 (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 {0,...,N-1}.*

Proposition 2.3. *Formule de décomposition dans une base orthonormale :*

$$\forall f \in \mathbb{R}, f = \sum_{n\in N} c_n e_n$$

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

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

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

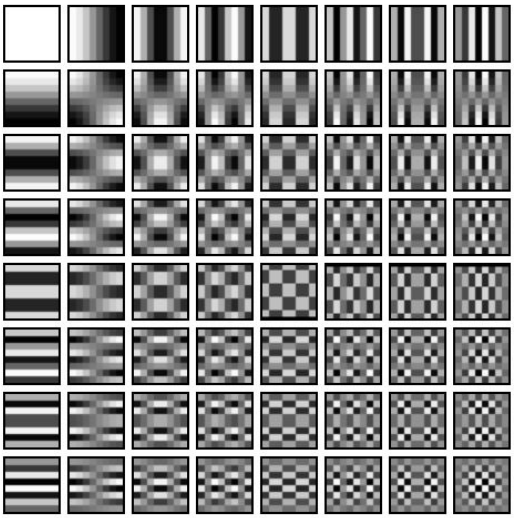

FIGURE 1 – Représentation de la base $\mathcal{B}_{DCT}$, aussi appelée base des cosinus locaux [2]

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 3.1. *Par soucis de cohérence notationnelle avec les articles sur le sujet [3], 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 [3] :

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

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 3.2. *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.*

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 4.1 (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 :*

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

Remarque 4.1. *La transformée de Fourier discrète, tout comme la tranformée de Fourier, est définie à constantes près. Ici, nous avons fait le choix d'une constante de 1/2 dans l'exponentielle. Ce choix simplifiera bien des calculs par la suite.*

Proposition 4.2. *$\mathcal{B}_{TFD1} := \{e_0, ..., e_{N-1}\}$ avec :*

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

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

Démonstration. *Pour toute paire (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 4.3. *La transformée de Fourier 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 4.2. *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.*
Si $\tilde{f}$ *est symétrique horizontalement et verticalement, alors on a* $\forall x, y \in \{0,...,15\}$ *:*

$$\tilde{f}(x,y) \overset{sym.hor}{=} \tilde{f}(15-x,y) \overset{sym.ver}{=} \tilde{f}(15-x,15-y)$$

Cette construction est illustrée par la figure ci-dessous.

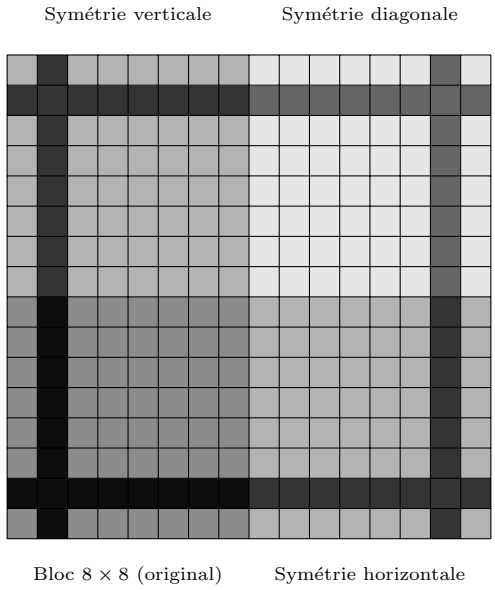

FIGURE 2 – Prolongement symétrique d'un bloc 8×8 en un bloc 16×16

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 4.2.

Proposition 4.4.

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

Démonstration.

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

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

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

Proposition 4.5.

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

Démonstration. *Comme g est réelle, par la propriété 4.4 :*

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

En utilisant la symétrie diagonale, i.e. $\hat{g}(x,y) = \hat{g}(16-x,16-y)$, *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 4.6. $\hat{g}_{sin} = 0$ *donc* $\hat{g} = \hat{g}_{cos}$.

Démonstration.

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

En appliquant la proposition 4.5, nous obtenons :

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

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

On peut aussi démontrer cette propriété en utilisant la propriété 4.4. En effet, la propriété 4.5 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 4.2, 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 :

$$T\tilde{f}(f)(k,l)$$

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

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 3. Ainsi, la DCT est la version réelle de la TFD appliquée à une fonction prolongée par symétrie.

5 Processus de compression JPEG

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

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 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 5.1. *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).*

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 :

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

où la division est une division terme à terme aussi appelée division d'Hadamard.

Voici une matrice typique pour une table de quantification standard pour la luminance. C'est celle que nous utiliserons principalement pour notre code [4] :

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

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 5.2. *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.*

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 3. Cette réorganisation facilite le codage entropique car nous pouvons représenter la suite de valeurs nulles par une seule notation de type [N,0].

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.

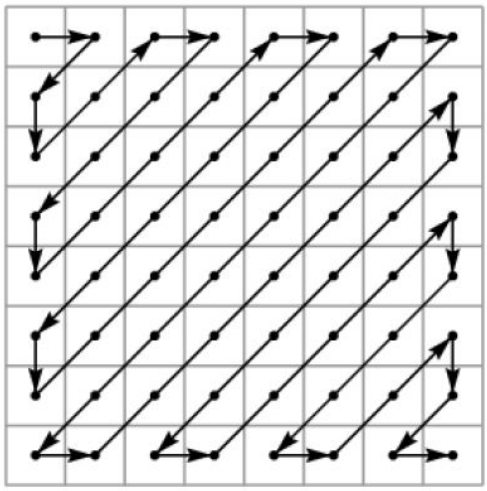

FIGURE 3 – Parcours en zigzag d'un bloc 8x8 [5]

Par exemple, considérons la suite de données suivante :

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

Après encodage RLE, on obtient :

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

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 une méthode quasi-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 son caractère d'origine.

Remarque 5.3. *L'encodage RLE et Huffman sont des algorithmes lossless, c'est-à-dire qu'ils n'introduisent pas de perte de données.*

Remarque 5.4. *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.*

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.

7 Conclusion

Au cours de ce travail, nous avons pu réaliser à l'aide d'outils de licence, une étude théorique de l'algorithme de compression JPEG. Nous avons fortement insisté sur le changement de représentation des données au coeur de cet algorithme en décortiquant son lien avec la transformée de Fourier discrète. Nous avons pu nous rendre compte que le changement de base choisi, ici la transformée en cosinus discrète, était crucial. Mais pourquoi utiliser ce changement de base ci ? N'existe pas des changements de base plus optimaux ? C'est ce questionnement qui a conduit les experts du JPEG à élaborer l'algorithme de compression JPEG2000. Cet algorithme est quant à lui un algorithme de compression par ondelettes, les ondelettes étant des fonctions oscillantes localisées à la fois dans l'espace et en fréquence. JPEG2000 offre des fonctionnalités différentes de celles de l'algorihme JPEG comme l'affichage multi-résolution. Il représente un pas de plus dans le domaine du traitement du signal dont ce travail n'a été qu'une introduction. L'approche théorique proposée dans ce document est complétée par une approche expérimentale sous forme de code Python. Vous pouvez accéder à ce code via le lien présent en fin d'introduction.

Références

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

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

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

[4] 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`.

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