

Rapport de Projet : Compression d'images JPEG

Étude théorique, Implémentation C++ et Analyse des performances

Khanh-Phuong NGUYEN

Département Informatique – Traitement du Signal

Année universitaire 2025–2026

Table des matières

1	Introduction	2
1.1	Contexte et Problématique	2
1.2	Objectifs du projet	2
2	Modélisation Mathématique et Algorithmique	3
2.1	Prétraitement et Espace Colorimétrique	3
2.2	Transformée en Cosinus Discrète (DCT)	3
2.3	Quantification : L'étape de compression	3
2.4	Codage Entropique (Sans perte)	4
2.4.1	Parcours Zig-Zag	4
2.4.2	Codage RLE et Différentiel	4
2.4.3	Codage de Huffman	4
3	Implémentation et Retours d'expérience	5
3.1	Architecture du programme	5
3.2	Difficultés rencontrées et solutions	5
4	Résultats et Analyse	6
4.1	Métriques de qualité	6
4.2	Tests sur l'image "Lenna"	6
5	Conclusion	7

Chapitre 1

Introduction

1.1 Contexte et Problématique

Les images numériques occupent une place centrale dans les échanges actuels (photographie, vidéo, web). Une image brute (RAW ou BMP) en couleurs de taille $N \times M$ codée sur 24 bits (3 octets par pixel) nécessite un espace de stockage considérable. Par exemple, une simple photo 12 mégapixels pèserait environ 36 Mo sans compression. Ce volume est incompatible avec les contraintes de bande passante des réseaux mobiles et de stockage sur serveurs.

L'objectif de ce projet est d'implémenter une chaîne de compression conforme au standard **JPEG** (Joint Photographic Experts Group) [1]. Ce standard repose sur une approche de compression *avec perte* (lossy), tirant parti des limites de la perception visuelle humaine pour réduire drastiquement la taille des fichiers.

1.2 Objectifs du projet

Le travail présenté dans ce rapport consiste à :

1. Comprendre et modéliser la chaîne de codage JPEG complète.
2. Implémenter les algorithmes clés : changement d'espace colorimétrique, DCT, Quantification, et codage entropique (RLE/Huffman).
3. Analyser le compromis entre la qualité visuelle (PSNR) et le taux de compression en jouant sur le facteur de qualité (F_q).

Chapitre 2

Modélisation Mathématique et Algorithmique

La chaîne JPEG découpe l'image en blocs de 8×8 pixels et traite chaque bloc indépendamment. Voici les étapes détaillées, basées sur les notes de cours [2].

2.1 Prétraitement et Espace Colorimétrique

Deux cas sont traités :

- **Image niveaux de gris (PGM)** : il n'y a qu'un canal Y . Aucune conversion colorimétrique ni sous-échantillonnage n'est nécessaire ; les blocs 8×8 sont passés directement à la DCT.
- **Image couleur (PPM)** : l'image RGB est convertie en **YCbCr** pour séparer luminance et chrominances : Y (luminance), Cb (chrominance bleue), Cr (chrominance rouge). On peut alors sous-échantillonner Cb/Cr ($4 : 4 : 4$ ou $4 : 2 : 0$) pour réduire le débit sans forte perte visuelle.

Cette séparation, spécifique au cas couleur, exploite la sensibilité plus élevée de l'œil à la luminance qu'aux variations chromatiques.

2.2 Transformée en Cosinus Discrète (DCT)

Conformément à T.81 (Annexe K), on applique la DCT-II 2D sur chaque bloc 8×8 après recentrage à $I'(x, y) = I(x, y) - 128$:

$$F(u, v) = \frac{1}{4} C(u) C(v) \sum_{x=0}^7 \sum_{y=0}^7 I'(x, y) \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right), \quad C(k) = \begin{cases} \frac{1}{\sqrt{2}} & k = 0, \\ 1 & k \neq 0. \end{cases}$$

La reconstruction (IDCT) suit la formule réciproque avec les mêmes $C(u)C(v)$, puis un ajout de 128 et un écrêtage dans $[0, 255]$.

Interprétation : $F(0, 0)$ (DC) porte la moyenne lumineuse du bloc ; les coefficients **AC** (haut fréquences vers les coins) codent les détails de plus en plus fins avant quantification.

2.3 Quantification : L'étape de compression

La DCT seule ne réduit pas la taille : la compression provient de la **quantification** définie dans la norme JPEG. Deux tables de base (luminance et chrominance) sont fournies par T.81.

Dans le code, on applique le facteur de qualité utilisateur F_q en recalculant une table adaptée :

$$S = \begin{cases} \frac{5000}{F_q} & \text{si } F_q < 50, \\ 200 - 2F_q & \text{sinon.} \end{cases} \quad \text{puis} \quad Q_{F_q}(u, v) = \max\left(1, \min\left(255, \frac{S \cdot Q_{base}(u, v) + 50}{100}\right)\right).$$

On divise chaque coefficient fréquentiel par $Q_{F_q}(u, v)$ et on arrondit :

$$F_q(u, v) = \text{round}\left(\frac{F(u, v)}{Q_{F_q}(u, v)}\right).$$

Les hautes fréquences (coins de la matrice) reçoivent des diviseurs plus grands et deviennent souvent nuls, ce qui favorise le RLE et Huffman ultérieurs.

2.4 Codage Entropique (Sans perte)

Une fois quantifié, le bloc contient beaucoup de zéros. J'ai implémenté les étapes suivantes :

2.4.1 Parcours Zig-Zag

Au lieu de lire la matrice ligne par ligne, on la lit en *zig-zag*. Cela permet de regrouper tous les coefficients non-nuls au début et de rejeter les suites de zéros à la fin.

2.4.2 Codage RLE et Différentiel

- **DC** : Codé par différence ($Diff = DC_i - DC_{i-1}$).
- **AC** : Codé par **RLE**. On stocke des paires (`run, level`).
- Le marqueur **EOB** (End Of Block) indique la fin du bloc.

2.4.3 Codage de Huffman

Les symboles sont traduits en séquences binaires grâce aux tables de Huffman définies dans la norme [1].

Chapitre 3

Implémentation et Retours d'expérience

3.1 Architecture du programme

- **Organisation** : modules C++ séparés pour le cœur (lecture/écriture, bitstream), la DCT, la quantification et le Huffman. Le binaire `jpeg_cli` orchestre compression et décompression.
- **Pipeline niveaux de gris** : découpe en blocs $8 \times 8 \rightarrow$ DCT \rightarrow quantification \rightarrow zig-zag \rightarrow RLE/Huffman \rightarrow écriture `.huff` + métadonnées `.meta`. Décompression fait l'inverse avec IDCT.
- **Pipeline couleur** : conversion RGB \rightarrow YCbCr, sous-échantillonnage optionnel Cb/Cr ($4:4:4$ ou $4:2:0$), puis la même chaîne par canal ; à la sortie, sur-échantillonnage et conversion YCbCr \rightarrow RGB.
- **Fichiers générés** : `.meta` contient dimensions, qualité, mode couleur et tables ; `.huff` contient le flux entropique (RLE + Huffman).
- **Tests** : jeux de tests unitaires (dct, quantification, rle, huffman, compression) et images exemples PGM/PPM pour valider la chaîne complète.

3.2 Difficultés rencontrées et solutions

1. **Bit-packing** : nécessité d'un tampon de bits pour émettre des codes Huffman de longueur variable sans perdre d'alignement.
2. **Décalage des niveaux** : oubli initial du -128 avant DCT qui saturait les DC ; corrigé en centrant systématiquement les blocs.
3. **Padding** : ajout d'un bourrage contrôlé lorsque l'image n'est pas multiple de 8 en hauteur/largeur, avec mémorisation dans `.meta` pour la reconstruction.
4. **Chroma** : gestion du ré-échantillonnage $4:2:0$ en sortie (duplication simple des échantillons Cb/Cr) pour rester cohérent avec la taille Y .

Chapitre 4

Résultats et Analyse

4.1 Métriques de qualité

Pour évaluer mon compresseur, j'utilise le taux de compression (T) et le PSNR :

$$PSNR = 10 \log_{10} \left(\frac{255^2}{EQM} \right) \text{ dB}$$

4.2 Tests sur l'image "Lenna"

Qualité (Q)	Taux (T)	PSNR (dB)	Observation visuelle
90	0,55	38,9	Visuellement identique à l'original.
50	0,79	31,1	Légers artefacts au zoom.
10	0,93	25,7	<i>Blocking</i> visible et perte de détails fins.

TABLE 4.1 – Performances mesurées sur Lenna NB (128×128) avec les valeurs issues du programme (T = taille compressée / taille originale).

Mesures directes (EQM) obtenues avec `jpeg_cli` sur le fichier `lenna.img` :

- $Q = 90$: EQM $\approx 8,35$.
- $Q = 50$: EQM $\approx 50,37$.
- $Q = 10$: EQM $\approx 176,55$.

Analyse. La dégradation suit bien le standard :

- **Tendance PSNR/qualité** : $Q = 90$ reste au-dessus de 38 dB (quasi indiscernable), $Q = 50$ descend à 31 dB (artefacts discrets), $Q = 10$ tombe à 26 dB avec *blocking* marqué.
- **Taux de compression** : T augmente quand Q baisse ($0,55 \rightarrow 0,93$) ; l'usage d'une table plus sévère annule davantage d'AC.
- **Répartition fréquentielle** : les blocs aux textures fines (plumes, chapeau) perdent les hautes fréquences dès $Q = 50$; les zones lisses (joues, fond) restent stables.
- **Erreurs typiques** : *blocking* et légers anneaux (ringing) apparaissent en bas Q autour des transitions fortes (bord du chapeau, yeux).

Couleur (rappel synthétique). Sur une PPM 8×8 artificielle, la chaîne 4 :4 :4 restitue l'image sans artefact visible ; en 4 :2 :0, les contours colorés restent acceptables mais les transitions très fines sont adoucies. Sur Lenna couleur pleine taille, le même comportement est attendu : perte surtout dans Cb/Cr après sous-échantillonnage et quantification.

Chapitre 5

Conclusion

Bilan. La chaîne implémentée suit les étapes clés du JPEG : DCT centrée, tables de quantification issues de T.81 et scalées par F_q , parcours zig-zag, RLE et Huffman. Les mesures sur Lenna confirment le compromis attendu entre PSNR et taux de compression.

Limites actuelles.

- Pas de mode progressif ni d'arithmétique, uniquement Huffman de base.
- Sous-échantillonnage chroma avec ré-échantillonnage bilinéaire minimal (simple duplication), améliorable par filtres mieux adaptés.
- Pas d'optimisation SIMD ; la DCT est calculée en flottant standard.

Pistes d'amélioration.

- Vectoriser la DCT/IDCT (SIMD) et utiliser des entiers sur 16 bits après mise à l'échelle pour accélérer.
- Ajouter un filtrage d'up-sampling chroma plus doux (lanczos/bicubique) et un léger post-filtre deblocking pour bas Q .
- Supporter le mode progressif et l'optimisation des tables Huffman à partir des statistiques d'image.

Bibliographie

- [1] ITU-T Recommendation T.81, *Digital Compression and Coding of Continuous-Tone Still Images (JPEG)*, 1992.
- [2] Notes de cours “JPEG 25–26”, support pédagogique sur la chaîne JPEG, Université de Poitiers, 2025.