



Méthodes de compression numérique d'images

Réalisé par

Philippe Morin (536 776 382)

Emilie Carré-Smith (111 235 839)

Maxime Tousignant-Tremblay (536 772 369)

Dans le cadre du cours
Physique Numérique (PHY-3500)

Résumé

Dans un contexte où la gestion efficace de l'espace de stockage devient de plus en plus cruciale, ce projet explore le développement de méthodes de compression d'image algorithmiques. L'objectif principal est d'élaborer des techniques de base pour compresser des images, en mettant un accent particulier sur leur accessibilité et leur efficacité. Ce rapport décrit d'abord les méthodes utilisées, les plus importantes étant la compression colorimétrique, la compression en « Discrete Cosine Transform (DCT) », le procédé de quantification et l'encodage Huffman. Ces algorithmes de compression sont testés sur une image de référence, une version bruitée de cette même image, ainsi que des acquisitions prises avec une caméra en expérimentation. L'efficacité des méthodes appliquées individuellement s'est avérée très efficace, en ce qui concerne la compression virtuelle de données et le critère de similarité d'images, le *MSSIM*. Cependant, lors de la réduction réelle de stockage, les résultats prometteurs ont été démentis, ayant obtenu un facteur qualité-performance maximal pour une réduction d'espace de 30.787%, un *MSSIM* de 42.710% et un aspect verdâtre a été ajouté à l'image. De plus, l'algorithme n'a pas démontré significativement de capacité à filtrer les bruits ajoutés. À la toute fin du développement, les algorithmes ont été améliorés pour supporter les images prises avec caméra, ce qui a permis d'obtenir une nette amélioration du filtre de bruit et de la compression, pouvant maintenant atteindre une réduction de 48.159 % et un facteur *MSSIM* de 97.019%. La compression *.jpeg* reste supérieure à la combinaison de méthodes conçue lors du projet. Cependant, l'objectif du projet est atteint, ayant pu découvrir les limitations dues à l'utilisation de types de données volumineuses et que les résultats, surtout ceux de la caméra, sont extrêmement satisfaisants.

Table des matières

1	Introduction	1
2	Méthodes	1
2.1	Choix des données à analyser	1
2.2	Compression colorimétrique	2
2.3	Algorithme de compression DCT	2
2.4	Algorithme de compression SVD	3
2.5	Algorithmes d'encodage	4
2.6	Évaluation des performances de compression et de la robustesse	4
3	Résultats	5
3.1	Algorithmes de compression	5
3.2	Effet de l'ajout de bruits	10
3.3	Traitements des acquisitions avec détecteur	11
4	Discussion	11
4.1	Algorithme de compression par colorimétrie	11
4.2	Algorithme de compression DCT	11
4.3	Algorithme de compression SVD	12
4.4	Algorithmes d'encodage	13
4.5	Algorithme de compression à méthodes mixtes	13
4.6	Effet de l'ajout de bruits	14
4.7	Grande image sur l'ensemble du projet	15
4.8	Traitements des acquisitions prises avec caméra	16
5	Conclusion	16

1 Introduction

La compression de données est un outil extrêmement puissant qui influence directement le quotidien de chacun, ainsi que l'économie. Que ce soit pour des films, des vidéos, des fichiers audio, des images ou autres documents, des techniques ont été développées afin d'encoder l'information et de filtrer celle inutile pour réduire le stockage occupé par les données [8]. Sans grande surprise, le nombre d'acquisitions pouvant être prises lors d'une séance de mesure est limité, ce qui est une contrainte inévitable. Un exemple concret d'information inutile est la présence de détails ou de couleurs trop précises afin d'être perçus distinctement par l'œil humain. De plus, du bruit de mesure est intrinsèquement présent lors d'acquisitions de séquences, pouvant provenir de la physique des particules et du détecteur, du circuit d'acquisition et de la numérisation. [7]. Ainsi, être en mesure de comprendre et concevoir des algorithmes de compression d'images est une compétence utile pour tous scientifiques, afin d'avoir accès à une compression de données personnalisée à ses besoins.

L'objectif principal de ce rapport est de développer des méthodes algorithmiques de compression d'images de base. Les objectifs secondaires sont de rendre ces méthodes accessibles en les vulgarisant adéquatement et de démontrer leur efficacité. La suite de ce rapport présente la réalisation de ces méthodes de base, les résultats associés à ces méthodes, une discussion sur les points forts et les faiblesses de ces méthodes, et une brève conclusion sur l'essence du rapport et une ouverture sur la suite d'un tel projet.

2 Méthodes

2.1 Choix des données à analyser

Pour débuter l'expérimentation, il faut déterminer les données sur lesquelles les algorithmes de compression vont opérer. Trois cas de figure sont pertinents : une image détaillée afin de développer et caractériser les performances de compression, des versions bruitées de l'image d'analyse, et des acquisitions obtenues avec une caméra disponible sur le marché. Beaucoup d'options sont disponibles. À des fins d'accélération du développement des algorithmes, prévenir d'éventuels problèmes d'acquisition et de mieux observer les changements résultant des algorithmes, l'image *.tiff* d'un félin avec une panoplie de détails est utilisée (apparence illustrée distinctement dans la section 3). De plus, pour s'assurer de l'impact des méthodes selon la capacité à filtrer les bruits nuisibles, un algorithme simple d'ajout de bruit peut être appliqué aux images. L'algorithme conçu permet d'ajouter plusieurs types de bruits à l'image, et aussi de varier l'ampleur de ce bruit. Le bruit peut suivre plusieurs distributions, telles qu'une distribution gaussienne, une distribution de Poisson, une distribution aléatoire ou bien des bruits d'interférences « speckle » [9]. Ensuite, la caméra choisie afin d'effectuer les acquisitions expérimentales est la caméra *1.6 MP CMOS compact scientific* de Thorlabs [10]. Cette caméra est souvent utilisée dans des systèmes des montages de microscopes. Alors, l'acquisition de marques et de défauts présents sur un simple rapporteur d'angle en plastique transparent semble bien indiquée. Cela permet de pallier à l'inaccessibilité d'échantillons biologiques conservés dans des lames de verre.

2.2 Compression colorimétrique

La première étape de compression de l'image consiste à réorganiser l'information contenue dans le fichier afin d'obtenir une image YCbCr, qui contient dans la matrice Y la luminance de l'image, et les matrices Cb et Cr contiennent la couleur[5]. L'œil humain étant plus sensible à la variation de luminosité qu'à la variation de couleur, il serait possible de réduire la résolution en couleur de l'image, en sous-échantillonnant les informations contenues dans les canaux Cb et Cr, pour réduire la taille du fichier image. À noter que les algorithmes de compression pensés vont opérer sur un canal à la fois, c'est-à-dire que les matrices seront traitées individuellement par les algorithmes, puis recombinées à l'étape finale de recombinaison. Pour effectuer la conversion en YCbCr, il faut ouvrir les *.tiff* en format RGB, puis déterminer les paramètres en utilisant les coefficients appropriés ci-dessous, selon le standard BT.2020 [1].

$$\begin{aligned}C &= rR + gG + bB \\Y &= 0,299R + 0,587G + 0,114B \\Cb &= 0,564(B - Y) \\Cr &= 0,713(R - Y)\end{aligned}$$

Où C est la couleur résultante, R, G, B les vecteurs de couleurs pour le rouge, le vert et le bleu et r, g, b sont les intensités des couleurs, ayant un nombre de valeurs possibles conformément à la résolution en bits de l'image.

2.3 Algorithme de compression DCT

Une fois l'image compressée en YCbCr en couleur sous-échantillonnée, la compression algorithmique peut être entamée. Parmi les techniques de compression avec pertes, la Transformée Cosinus Discrète (DCT) est l'une des plus utilisées, notamment dans les standards de compression tels que JPEG. Cette méthode se base sur la transformation des données de l'image du domaine spatial au domaine fréquentiel, permettant ainsi une quantification plus efficace des informations [5]. Ainsi, l'image est segmentée en matrice de blocs de dimension $N \times N$, où N est un entier positif typiquement égal à 8 [5]. Pour un bloc d'image de taille $N \times N$, la DCT est définie par l'équation suivante pour chaque coefficient $C(u, v)$:

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right],$$

Où $f(x, y)$ est l'intensité du pixel à la position (x, y) , u et v sont les indices des fréquences horizontales et verticales et $\alpha(u)$ et $\alpha(v)$ sont des facteurs d'échelle définis par :

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{si } u = 0 \\ \sqrt{\frac{2}{N}} & \text{autrement} \end{cases}$$

Après l'application de la DCT, il faut effectuer le retrait d'information en employant la quantification, afin de réduire le nombre de bits nécessaires pour stocker l'image. La quantification consiste à simplifier les valeurs des coefficients de transformée obtenues, réduisant ainsi la précision, mais

conservant l'essentiel de l'information visuelle. Alors, chaque coefficient DCT est divisé par un coefficient de quantification correspondant, qui est déterminé par une table de quantification JPEG standard ou personnalisée. Ce coefficient contrôle le niveau de compression et de perte de qualité :

$$B'_{i,j} = \text{round} \left(\frac{B_{i,j}}{Q_{i,j}} \right)$$

Où $B_{i,j}$ est le coefficient DCT du bloc, $Q_{i,j}$ est le coefficient de la table de quantification pour la position (i,j) , et $B'_{i,j}$ est le coefficient quantifié. Les tables de quantification varient la quantité de compression en utilisant des valeurs plus grandes pour une compression plus forte (moins de précision) et des valeurs plus petites pour moins de compression. Ces tables sont souvent optimisées pour la perception humaine, réduisant davantage les fréquences où l'œil est moins sensible. Celle utilisée dans la méthode de quantification du projet est extraite d'un code existant [2].

Alternativement à la méthode DCT, il est possible d'utiliser la compression en considérant uniquement les amplitudes fréquentielles obtenues dans l'espace de Fourier suite à une des transformées de Fourier rapides, et en appliquant la même quantification que dans l'algorithme DCT.

2.4 Algorithme de compression SVD

La décomposition en valeurs singulières (SVD) est une technique d'algèbre linéaire qui décompose une matrice en trois autres matrices, révélant des propriétés fondamentales de la matrice originale [6]. Pour une image, qui peut être représentée comme une matrice où chaque élément représente l'intensité d'un pixel, la SVD découpe cette matrice en composantes qui révèlent des structures importantes de l'image. Les valeurs singulières dans Σ donnent une indication de la quantité d'information portée par les vecteurs singuliers correspondants [6]. Dans le contexte d'une image, les premières valeurs singulières (les plus grandes) capturent les caractéristiques les plus significatives de l'image, tandis que les valeurs singulières plus petites peuvent être associées à des détails fins ou du bruit. A est ici une matrice de dimensions $m \times n$ représentant une image en niveaux de gris. La SVD de A s'écrit comme suit :

$$A = U\Sigma V^T$$

où : - U est une matrice $m \times m$ orthogonale contenant les vecteurs propres de AA^T . Ces vecteurs sont appelés vecteurs singuliers à gauche. - Σ est une matrice diagonale $m \times n$ avec des valeurs non négatives sur la diagonale, appelées valeurs singulières. Elles sont triées par ordre décroissant et mesurent l'importance ou le "poids" de chaque vecteur singulier. - V^T est la transposée d'une matrice $n \times n$ orthogonale contenant les vecteurs propres de A^TA . Ces vecteurs sont appelés vecteurs singuliers à droite.

Ainsi, en conservant uniquement les k premières valeurs singulières et les colonnes correspondantes de U et V , on peut obtenir une approximation de rang k de A :

$$A_k = U_k \Sigma_k V_k^T$$

Où U_k , Σ_k , et V_k^T contiennent respectivement les k premières colonnes, les k premières valeurs singulières, et les k premières lignes de V^T . Cette approximation de bas rang minimise l'erreur quadratique entre la matrice originale et sa version compressée, par rapport à toute autre approximation de même rang.

La décompression, ou reconstruction de l'image consiste à inverser le processus de compression pour retrouver une forme très similaire à l'image originale. Étant donné que le stockage des informations est compris dans U_k , Σ_k , et V_k^T , la reconstruction se fait simplement en recalculant le produit A_k de ces matrices. De plus, la compression SVD est d'un intérêt particulier à être développé, afin de la comparer à la méthode typiquement utilisée en compression JPEG, soit la méthode DCT. De plus, ces méthodes de compression peuvent être combinées, afin de pouvoir évaluer les conséquences de ce choix de design atypique.

2.5 Algorithmes d'encodage

Une fois l'information compressée avec le plus de pertes possible tolérées, il faut tenter l'encodage d'informations afin de compresser réellement l'image en nombre de bits sauvegardés. En effet, la simple sauvegarde des fichiers décompressés ou des *ndarray* n'est pas en mesure d'obtenir un gain net d'espace. Il faut plutôt effectuer un encodage, visant à réarranger l'information, dont l'efficacité dépend directement des modifications effectuées auparavant par les autres algorithmes de compression. L'encodage Huffman est une technique de compression de données sans perte qui utilise des codes de longueur variable pour représenter les symboles, en fonction de leur fréquence d'occurrence. Les symboles les plus fréquents sont codés avec les chaînes de bits les plus courts, et les moins fréquents avec les plus longues. Cette méthode permet ainsi de minimiser la longueur totale des données codées, en s'adaptant dynamiquement à la distribution réelle des symboles. L'encodage de Huffman suit les étapes suivantes [3].

1. **Analyse des fréquences** : Compter la fréquence de chaque symbole dans les données (dans le cas de JPEG, les symboles peuvent être les coefficients quantifiés).
2. **Construction de l'Arbre Huffman** : Créer un nœud pour chaque symbole et le placer dans une file de priorité basée sur la fréquence. Répéter jusqu'à ce que la file soit vide. Retirer les deux nœuds avec la plus faible fréquence. Fusionner ces nœuds pour former un nouveau nœud dont la fréquence est la somme des deux fréquences. Insérer le nouveau nœud dans la file.
3. **Génération des codes Huffman** : Assigner un code binaire à chaque symbole en parcourant l'arbre de Huffman, en attribuant un "0" pour aller à gauche et un "1" pour aller à droite.

Avant d'entamer l'encodage de Huffman, il peut être avantageux d'effectuer un encodage en longueur de course (RLE) [5]. Plus les matrices comporteront des éléments identiques, plus la méthode est efficace.

2.6 Évaluation des performances de compression et de la robustesse

Une fois la méthode de compression développée, des métriques et des critères doivent être imposés afin d'évaluer sa pertinence. Ainsi, trois critères sont considérés, soient, le taux de réduction virtuel, le taux de réduction effectif et le calcul du *MSSIM* (indice de similarité de structure). Le taux de réduction virtuel correspond à la quantité de signaux dans l'image en format *ndarray* filtrés numériquement lors de l'exécution des diverses méthodes de compression. Il s'agit davantage d'une métrique correspondant à l'efficacité de filtrage de signaux divergents que de la réduction actuelle en bits de la taille du fichier, étant le second critère évalué. Le *MSSIM* est basé sur les fortes interdépendances entre les pixels, proportionnelles à leur proximité. Ces interdépendances portent des informations importantes sur la structure de l'objet dans l'espace visuel. L'indice *MSSIM* prend en compte les changements de luminance, de contraste et de structure des images, et ce, en sommant

l'impact pour toutes les fenêtres de pixels pouvant être formés. Ici, une fenêtre correspond, pour chaque pixel, à la plus grande région carrée autour du pixel donné. Le *MSSIM* par pixel entre deux fenêtres d'images x et y de taille commune $N \times N$, entre l'image compressée et initiale, est défini comme suit :

$$\text{MSSIM}(x, y, \text{pixel}) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Où : μ_x et μ_y sont les moyennes de x et y , σ_x^2 et σ_y^2 sont les variances de x et y , σ_{xy} est la covariance entre x et y , $c_1 = (k_1 L)^2$ et $c_2 = (k_2 L)^2$ sont deux variables pour stabiliser la division avec un petit dénominateur, L est la plage dynamique des valeurs de pixel (généralement $L = 2^n - 1$ où n est le nombre de bits par pixel) et $k_1 = 0.01$ et $k_2 = 0.03$ sont des constantes par défaut.

Ainsi, considérant tous les pixels, le *MSSIM* total est :

$$\text{MSSIM}(x, y) = \frac{1}{M} \sum_{i=1}^M \text{MSSIM}(x_i, y_i) ,$$

Où M est le nombre total de fenêtres locales dans l'image. Il est sensible aux changements perceptibles par l'œil humain, ce qui permet de quantifier le critère de qualité visuelle, qui sera aussi évalué qualitativement lors de l'analyse des résultats.

3 Résultats

Cette section présente les résultats obtenus lors du développement des différentes méthodes mentionnées dans la section 2. Ainsi, les sous-sections présentent respectivement les résultats des algorithmes de compression sur l'image de référence en développement, les performances des algorithmes en situation d'ajout de bruits, et les performances des algorithmes traitant les images provenant de la caméra *1.6 MP CMOS compact scientific* de Thorlabs.

3.1 Algorithmes de compression



FIGURE 1 – Comparaison entre l'image RGB originale et l'image reconstruite après compression colorimétrique, illustrant une excellente préservation de la qualité d'image, malgré le sous-échantillonnage de la couleur réduisant la résolution de moitié. Utilisée seule, la méthode de compression maintient la qualité visuelle de l'image.

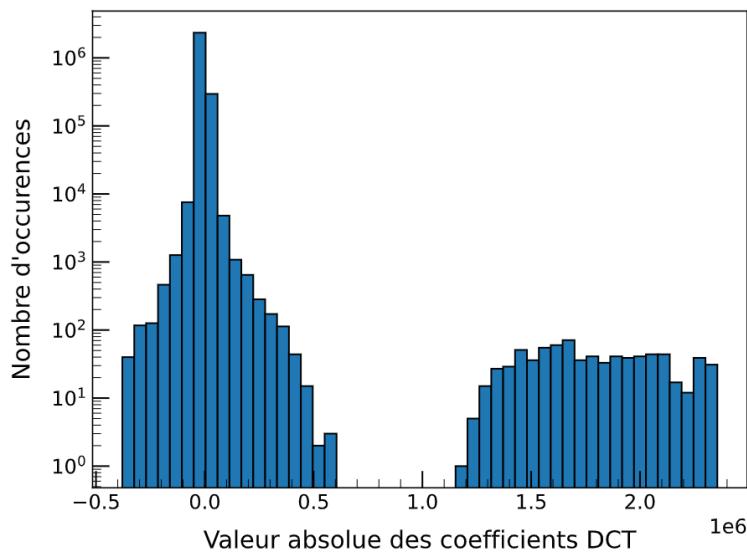


FIGURE 2 – Histogramme des coefficients DCT après compression, montrant une concentration élevée de coefficients proche de zéro, ce qui indique une compression efficace avec un minimum de perte d'information visuelle.



FIGURE 3 – Image originale comparée à l'image après la décompression DCT, révélant une fidélité visuelle élevée.

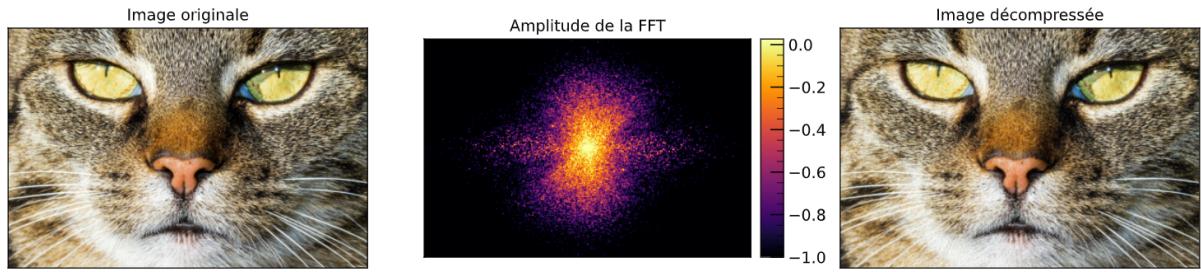


FIGURE 4 – Effets de la compression FFT sur l'image, incluant l'image originale, l'amplitude de la FFT qui montre la répartition des fréquences spatiales après la transformation et l'image reconstruite.

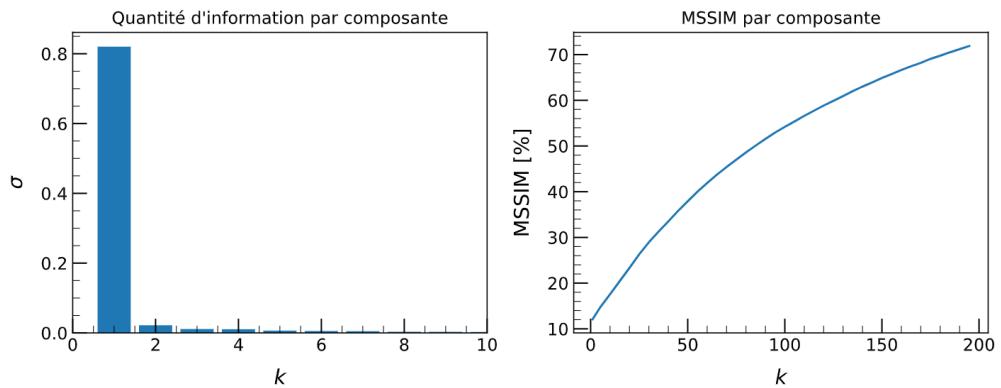


FIGURE 5 – À gauche, le graphique montre la quantité d'information conservée par nombre de valeurs singulières k , et à droite, l'évolution du MSSIM, révélant une amélioration de la qualité d'image avec l'augmentation de k , soulignant la corrélation entre la conservation des valeurs singulières et la fidélité de l'image reconstruite.

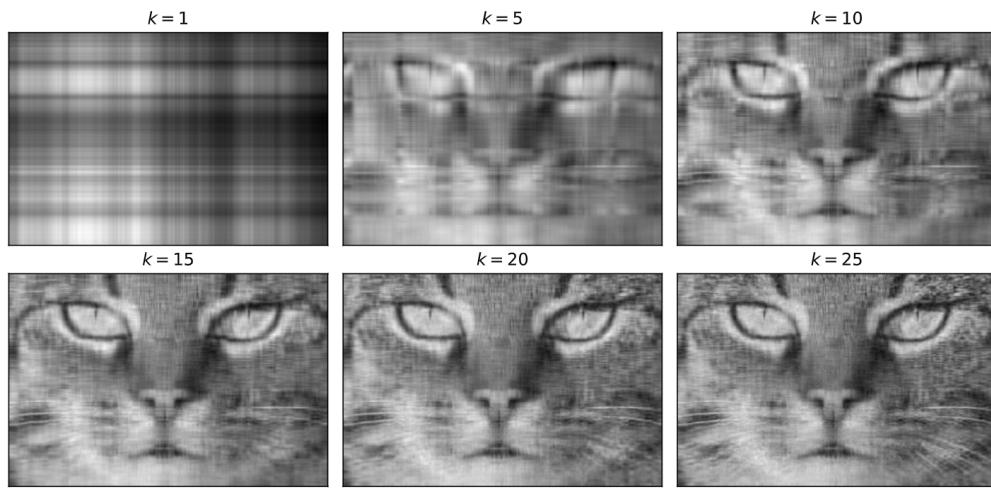


FIGURE 6 – Progression de la qualité des images reconstruites par SVD, montrant une amélioration claire de la clarté et des détails à mesure que k augmente.

TABLEAU 1 – Récapitulatif des méthodes de compression et leurs performances en ce qui concerne le taux de réduction virtuel (filtrage de signaux) et le *MSSIM*.

Méthode de compression	Taux de réduction virtuel (%)	MSSIM (%)
Colorimétrique	50	97.6
DCT	90	86.3
FFT	90	69.3
SVD	19	50.1

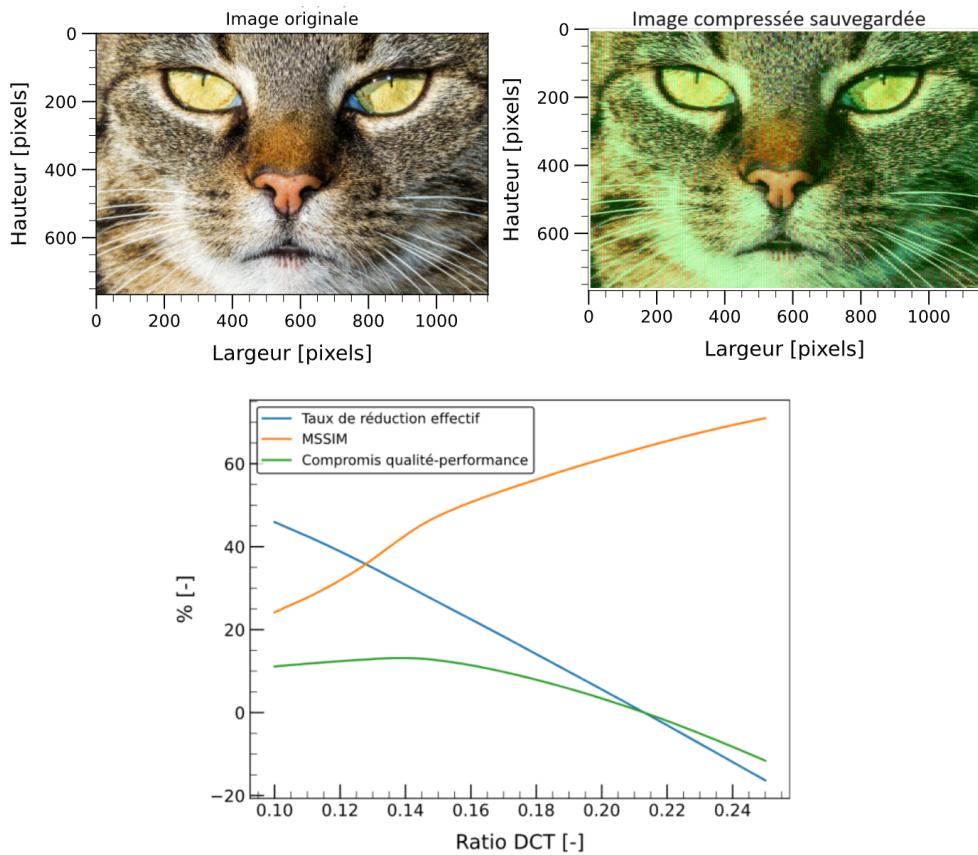


FIGURE 7 – Étude de la combinaison des méthodes de compression en faisant varier le ratio de compression DCT, maximal à 0.14. La compression colorimétrique, la compression DCT, la quantification et l'encodage Huffman permettent ensemble d'obtenir le plus grand ratio qualité performance pour une réduction de 30.787% de stockage et un *MSSIM* de 42.710%. Notons une importante limite de reconstruction de la méthode montrée lorsque le ratio DCT est de 0.214. L'image sauvegardée occupe le même espace initial, mais n'est pas fidèlement reconstruite, le *MSSIM* étant bien inférieur à 100%.

3.2 Effet de l'ajout de bruits

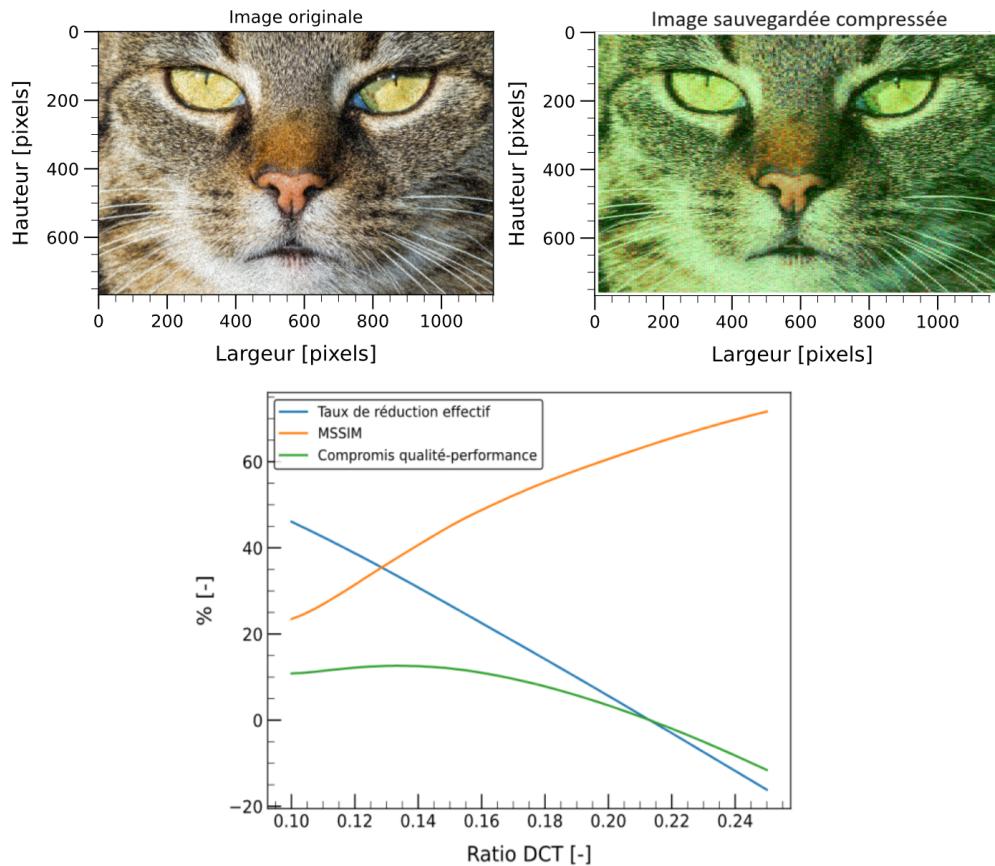


FIGURE 8 – Étude de la combinaison des méthodes de compression en faisant varier le ratio de compression DCT, maximale à 0.14 également pour le cas bruité. Le plus grand ratio qualité-performance pour une réduction de 59.712% de stockage et un *MSSIM* de 40.657 %. Les résultats obtenus sont douteux, étant donné que l'ajout de bruit poivre et sel ne semble pas avoir complètement disparu, et que le facteur de compression semble être plus élevé parce que l'image bruitée est plus volumineuse initialement.

3.3 Traitement des acquisitions avec détecteur

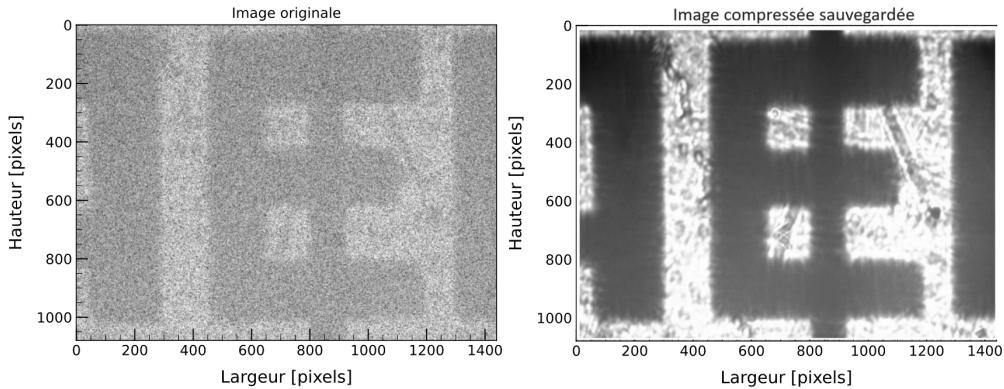


FIGURE 9 – Étude de la combinaison des méthodes de compression en faisant varier le ratio de compression DCT, maximale à 0.2 pour le cas avec acquisition de caméra. Lorsqu'il n'y a pas de couleurs à considérer, les résultats sont nettement plus satisfaisants. Ces résultats bénéficient aussi de la version de code la plus récente. Ainsi, le plus grand ratio qualité-performance donne un *MSSIM* 97.019 % et une réduction de stockage de 48.159 %.

4 Discussion

Cette section présente les discussions résultantes de chaque processus de compression présenté dans les section 3. Les forces et les limites de chaque méthode sont détaillées ci-dessous.

4.1 Algorithme de compression par colorimétrie

L'approche de compression par colorimétrie a démontré une efficacité notable dans le projet. La méthode a atteint un taux de réduction virtuel de 50%. En raison de son taux de réduction virtuel élevé comparé à d'autres techniques, et de son MSSIM élevé de 97.6%, la compression colorimétrique est souvent utilisée en parallèle avec d'autres techniques de compression d'images. Cette combinaison stratégique permet d'exploiter la sensibilité moindre de l'œil humain aux détails de couleur, permettant ainsi une compression plus poussée sans altérer significativement la perception globale de l'image. L'intégration de la compression colorimétrique avec d'autres techniques améliore non seulement l'efficacité globale du processus, mais contribue aussi à une meilleure gestion de l'équilibre entre la taille du fichier et la qualité de l'image, rendant cette méthode idéale pour les systèmes où la conservation des ressources et la qualité de l'image sont toutes les feux prioritaires.

4.2 Algorithme de compression DCT

L'approche utilisée pour la compression DCT dans ce projet s'est avérée efficace pour compresser des images en réduisant leur taille, tout en maintenant une qualité acceptable, comme en témoigne le taux de réduction virtuel de 90% et un MSSIM de 86.3%. Cette performance s'explique par la capacité de la DCT à transformer efficacement l'image du domaine spatial au domaine fréquentiel, ce qui permet une meilleure quantification et une compression plus efficace. La quantification qui

suit la transformation DCT est particulièrement efficace pour les zones de l'image où la variation de luminance est faible, permettant de réduire ou d'éliminer les coefficients DCT sans impacter de façon notable l'aspect de l'image. Cela montre que, même si la DCT est une technique de compression avec perte, elle conserve une qualité relativement élevée. L'efficacité de la quantification est largement dépendante des seuils choisis pour la réduction des coefficients DCT. Le graphique présent à la figure 7 illustre clairement cette dépendance. En effet, à mesure que le ratio de compression DCT augmente, le MSSIM s'améliore. Cette corrélation suggère que la qualité perçue peut rester relativement haute, en particulier lorsque les seuils de quantification sont soigneusement ajustés. Bien que la DCT soit très efficace pour les zones de l'image avec une faible variation de luminance, elle peut être moins performante dans les zones de haute complexité, où la suppression des coefficients de haute fréquence est plus susceptible de dégrader l'image. Ces zones nécessitent une approche plus nuancée pour la quantification, ce qui suggère l'utilisation de méthodes de quantification adaptative. Ces méthodes ajusteraient les seuils de quantification en fonction des caractéristiques locales de l'image, offrant ainsi une compression plus personnalisée et efficace. En comparaison, la méthode FFT, illustrée à la figure 4, montre une très grande similarité dans la transformation des données de l'image du domaine spatial au domaine fréquentiel. Cependant, étant donné la similitude des résultats et la performance supérieure de la DCT en termes de MSSIM, comme indiqué dans le tableau 1, la DCT a été privilégiée pour la suite du projet.

4.3 Algorithme de compression SVD

Le premier graphique de la figure 5 montre la quantité d'information contenue dans chaque composante singulière. La première composante porte, d'après le graphique, la majorité de l'information, car une chute rapide des valeurs est visible dès la deuxième composante. Cela signifie que la première valeur singulière capture un élément significatif de la structure globale de l'image, tandis que les valeurs singulières suivantes apportent beaucoup moins d'informations additionnelles. Cette propriété est exploitable en compression d'images. L'objectif principal est de réduire la quantité de données, en retenant uniquement les composantes les plus significatives. Le deuxième graphique de la figure 5 représente l'évolution du *MSSIM* en fonction du nombre de composantes utilisées pour reconstruire l'image. L'augmentation du *MSSIM* devient moins importante après les premières composantes, confirmant que chaque composante supplémentaire apporte de moins en moins d'améliorations qualitatives perceptibles à la reconstruction de l'image. La figure 6 illustre également cet effet : l'augmentation de la qualité d'image est en phase avec les valeurs k , mais l'image semble toujours floue, même pour des k plus élevées, comparativement à l'image originale en nuances de gris. Cependant, la croissance lente du *MSSIM* en fonction du nombre de composantes semble contradictoire avec le premier graphique, lequel souligne que la majorité de l'information est comprise dans les 10 premiers vecteurs propres. Or, même si l'ajout de nouvelles composantes apporte moins d'améliorations perceptibles, la courbe continue de croître, ce qui indique que la qualité perçue de l'image s'améliore toujours, bien que de façon marginale.

Les différences perceptibles peuvent être justifiées par différents facteurs [4].

1. La première composante singulière de la SVD capture la plus grande information structurale dans l'ensemble des données. Cependant, la qualité perceptive n'est pas seulement une question de variance. Les composantes ultérieures, bien qu'elles contribuent moins à la reconstruction, peuvent inclure de l'information visuellement importante pour reconnaître des textures et des détails fins.

-
2. Les images réelles ont souvent des régions avec des informations redondantes ou homogènes. La SVD est efficace pour détecter et compresser ces redondances. Cependant, le MSSIM, en se focalisant sur la comparaison de structures locales, peut montrer une amélioration insignifiante lorsque des composantes supplémentaires qui codent des redondances sont ajoutées.
 3. La perception humaine est très sensible à certains types de changements visuels et moins à d'autres. Par exemple, l'œil est plus sensible aux changements dans les basses fréquences (grandes structures) que dans les hautes fréquences (détails fins), signifiant que l'ajout de composantes représentant des détails fins pourrait avoir un impact négligeable sur le MSSIM.
 4. Le MSSIM tient compte de la sensibilité au contraste de la vision humaine, qui peut ne pas être directement proportionnelle à la variance représentée par les composantes singulières. Une petite quantité d'information ajoutée par des composantes ultérieures pourrait avoir un grand impact sur le contraste local et donc sur le MSSIM.

L'essentiel à retenir est que la méthode de compression SVD nécessite une utilisation plus ardue et moins efficace que la compression DCT. C'est pour cette raison que dans l'algorithme de compression avec combinaison de méthodes, la compression SVD n'est pas utilisée.

4.4 Algorithmes d'encodage

L'encodage des données compressées représente une étape importante dans notre processus de compression d'image. Toutefois, malgré les taux élevés de compression virtuelle obtenus grâce aux algorithmes colorimétriques et DCT, un défi majeur lors de la transition vers l'étape d'encodage a été constaté : une augmentation significative de la taille des données traitées. Cette expansion inattendue est principalement due aux transformations et aux processus intermédiaires nécessaires pour préparer les données à l'encodage, ce qui a entraîné une surcharge mémoire substantielle. Cet accroissement de la taille des données, malgré une réduction théorique significative, met en lumière les limitations des bibliothèques et objets Python utilisés dans l'implémentation. Les structures de données et les mécanismes de gestion de la mémoire de Python, bien que performants dans de nombreux contextes, peuvent se révéler moins efficaces lorsqu'il s'agit de gérer de grandes transformations de données compressées destinées à l'encodage. En particulier, l'utilisation de l'encodage Huffman, qui est normalement très efficace pour réduire la taille des données sans perte, n'a pas réussi à compenser l'augmentation de la taille due aux étapes précédentes. Cette limitation soulève un point critique sur la dépendance aux outils de développement choisis. Alors que nos méthodes de compression virtuelle promettaient des réductions significatives de la taille des données, la phase d'encodage a permis d'exposer les défis inhérents à la gestion efficace des ressources et à l'optimisation des performances dans des environnements de programmation.

4.5 Algorithme de compression à méthodes mixtes

L'algorithme de compression à méthodes mixtes exploité dans ce projet combine efficacement plusieurs techniques de compression pour optimiser à la fois la réduction de la taille des fichiers et la préservation de la qualité des images. Ce processus intègre la compression par colorimétrie, la compression DCT et la quantification appropriée, ainsi que l'encodage Huffman, pour optimiser à la fois la réduction de la taille des fichiers et la préservation de la qualité des images. Chacune de ces méthodes apporte des avantages spécifiques qui, lorsqu'elles sont combinées, offrent une solution de compression efficace.

-
- **Compression par colorimétrie** : Réduis les composantes de couleur tout en conservant la composante de luminance, ce qui diminue la taille des données sans affecter de manière significative la perception visuelle.
 - **DCT** : Très efficace dans les zones à faible variation, réduisant les coefficients de haute fréquence sans perte perceptible de détails.
 - **Encodage Huffman** : Applique une méthode de codage basée sur la fréquence des symboles pour réduire encore plus la taille du fichier.

La figure 7 présente l'image originale face à sa version compressée, ainsi qu'un graphique illustrant les relations entre le ratio DCT, le taux de réduction effectif, le *MSSIM* et l'indicateur de compromis qualité-performance. Une observation du graphique est que, pour un faible ratio DCT, bien que le taux de réduction effectif soit élevé, le *MSSIM* est plus bas, indiquant une qualité inférieure. À l'inverse, à mesure que le ratio augmente, le taux de réduction effectif diminue, mais le *MSSIM* augmente, signifiant une amélioration de la qualité, mais une diminution de la réduction. L'optimisation des paramètres de l'algorithme a permis d'identifier un point de rencontre optimal entre ces courbes, à un DCT de 0.14. À ce point, correspondant au pic de la courbe qualité-performance, un compromis idéal entre la taille de fichier et qualité image est atteint, avec une réduction de stockage de 30.787% et un *MSSIM* de 42.710%. Cette observation met en lumière la limite de reconstruction de la méthode lorsque le DCT est poussé à 0.214, où l'image occupe le même espace de stockage que l'original et présente une reconstruction non fidèle. Ces résultats démontrent l'importance d'une calibration méticuleuse du ratio DCT dans l'application de la compression multimodale pour maintenir un équilibre entre l'efficacité de la compression et la qualité visuelle. L'intégration de seuils adaptatifs ajustés automatiquement en fonction des caractéristiques spécifiques de chaque image pourrait résoudre ou du moins atténuer les limitations observées. L'adoption de telles améliorations pourrait permettre de mieux personnaliser la compression en fonction des exigences uniques de chaque cas d'utilisation, optimisant ainsi à la fois la réduction de la taille des fichiers et la préservation de la qualité de l'image.

4.6 Effet de l'ajout de bruits

L'ajout de bruit effectué est relativement négligeable, étant donné que les résultats obtenus avec l'image de référence sont déjà assez limités. Malgré une valeur de réduction de stockage presque double de celle de l'image originale, l'image bruitée est environ 1.72 fois plus volumineuse. Cela est dû à la manière dont *Python* sauvegarde les images *.tiff* générées par l'algorithme d'ajout de bruits. De plus, étant donné les similarités des graphiques des figures 7 et 10, cela pointe vers le fait que les images compressées résultantes sont extrêmement similaires. En effet, leur *MSSIM* et le ratio DCT optimisant les produits qualité-performance, tout simplement le produit du taux de réduction effectif et du *MSSIM*, sont très similaire. Ainsi, cela indique que l'erreur due à la sauvegarde *Python* est compensée. Cependant, quand une comparaison des images résultantes est effectuée, par Cependant, une simple observation visuelle, notamment des yeux du chat, permet de percevoir la présence de bruit sur les images résultantes. Ainsi, la compression s'avère inefficace pour éliminer le bruit, ce qui devient indéniable lorsque davantage de bruit est ajouté à l'image.

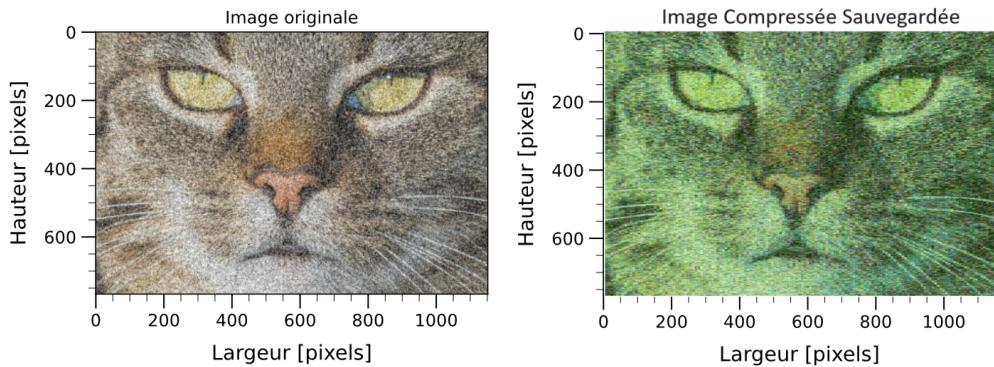


FIGURE 10 – Analyse de l’ajout supplémentaire de bruit. En conservant les mêmes paramètres de compression, le même taux de réduction effectif est conservé. Il y a une légère baisse du *MSSIM*. Par contre, le résultat confirme que l’ajout de bruit n’est pas compensé par l’algorithme de compression. Pour régler cette situation, il faudrait concevoir spécifiquement un algorithme qui se charge de la détection et de la filtration de bruits.

4.7 Grande image sur l’ensemble du projet

Les différents résultats obtenus lors de la réalisation du projet ont permis d’identifier les points les plus limitants du projet, ainsi que les forces des algorithmes à exécuter les nombreux traitements impliqués. Tout d’abord, la gestion d’objet et de types de données lors du traitement de données a été spécialement demandante. Il est impossible de conserver l’intégrité de l’image sans garder de volumineux *ndarray float64* à la fin de chaque opération. Cela nuit ultimement à l’efficacité de la compression, étant donné que le traitement de ce type de donnée est beaucoup plus long et complexe. Sachant cela, il est remarquable que le code puisse démarrer etachever l’ensemble du processus en un temps très raisonnable, soit moins d’une dizaine de secondes pour la compression mixte, même en manipulant des objets jusqu’à dix fois plus volumineux en mégaoctets.

De plus, l’algorithme s’est avéré être en mesure de compresser un fichier initialement compressé, partant d’un format *.tiff*. Bien que le taux de réduction de mémoire atteigne environ 30% et que certains défauts apparaissent, il s’agit toutefois d’une véritable compression. D’un autre côté, la simple conversion *Python* en 2 lignes de code de l’image en *.jpeg* permet de compresser 78.972% de l’espace occupé. Cela indique que les algorithmes de compression *.jpeg* sont nettement supérieurs à la combinaison de méthodes effectuées, autant en matière de qualité d’image que de compression. Il faut demeurer réaliste, l’objectif principal du projet était de réaliser les méthodes de compression et de comprendre davantage les méthodes afin de trouver des pistes de solution pour augmenter leur efficacité. Dans cette optique, le projet est un succès, puisque les méthodes principales utilisées lors de la compression *.jpeg* ont toutes été développées [5]. Alors, pour augmenter l’efficacité de ces méthodes, consulter divers autres codes de compression, développés avec la même contrainte de programmation en *Python*, serait très enrichissant et permettrait indubitablement de tirer parti des atouts d’autres algorithmes pour se rapprocher du résultat idéal [11].

4.8 Traitement des acquisitions prises avec caméra

Le projet étant en constante progression et poursuivant une vision d'amélioration continue, les résultats les plus récemment obtenus du traitement des acquisitions de caméra sont nettement supérieurs à ceux de l'image de référence présentée auparavant. Ainsi, la plupart des inconvénients traités auparavant ont été abordés et modifiés. Avec une performance de réduction d'espace de 48.159 %, l'efficacité de l'algorithme du projet est désormais comparable à celle du format *.jpeg*. La qualité et la capacité de filtrage ont considérablement augmenté, rendant cet algorithme particulièrement polyvalent et apte à l'amélioration continue.

5 Conclusion

Ce projet a exploré diverses approches de compression d'images, chacune avec ses forces et ses limites, dans le but d'optimiser la réduction de la taille des fichiers tout en préservant au mieux la qualité visuelle des images. Les méthodes étudiées, incluant la compression par colorimétrie, la DCT, la SVD, et des techniques d'encodage telles que Huffman, ont montré qu'une combinaison stratégique de ces techniques peut mener à des résultats significatifs en termes d'efficacité de compression et de maintien de la qualité. La compression par colorimétrie a prouvé son efficacité en exploitant la sensibilité moindre de l'œil humain aux détails de couleur, tandis que la DCT s'est distinguée par sa capacité à réduire la taille des données tout en conservant une qualité acceptable grâce à une gestion fine des seuils de quantification. L'algorithme SVD, bien qu'intéressant pour sa capacité à identifier les composantes significatives de l'image, s'est révélé moins efficace dans des contextes où la performance en temps réel et la simplicité algorithmique sont cruciales. En ce qui concerne les algorithmes d'encodage, des défis ont été rencontrés, notamment une augmentation de la taille des données due aux processus intermédiaires nécessaires à l'encodage. Ces défis soulignent l'importance de choisir judicieusement les outils et méthodes en fonction des spécificités du projet et mettent également en lumière les limitations propres aux environnements de développement utilisés.

En conclusion, ce projet démontre que la compression efficace des images est un équilibre délicat entre la réduction de la taille des données et la préservation de la qualité perçue. Cet équilibre, ainsi que les différents défauts identifiés au cours du développement des algorithmes, ont permis d'améliorer ultimement la capacité des algorithmes de compression. Lors de la présentation officielle du projet au Symposium, davantage de résultats et une vulgarisation approfondie, des méthodes de programmation utilisées permettront de présenter en détail les enjeux et les connaissances de programmation rencontrés et exploités.

Références

- [1] URL : <https://en.wikipedia.org/wiki/YCbCr>.
- [2] URL : <https://github.com/mVirtuoso21/JPEG-Image-Compressor/blob/main/main.py>.
- [3] URL : <https://www.programiz.com/dsa/huffman-coding>.
- [4] Zhou Wang et AL. 2004. URL : <https://doi.org/10.1109/TIP.2003.819861>.
- [5] Image ENGINEERING. *How does the JPEG compression works*. 2011. URL : <https://www.image-engineering.de/library/technotes/745-how-does-the-jpeg-compression-work#:~:text=The%20JPEG%20compression%20is%20a%20block%20based%20compression.,the%20data%20reduction%20by%20setting%20%28or%20chose%20presets%29..>
- [6] Machine learning + JAGDEESH. URL : <https://www.machinelearningplus.com/linear-algebra/singular-value-decomposition/>.
- [7] Mikhail KONNIK et James WELSH. *High-level numerical simulations of noise in CCD and CMOS photosensors : review and tutorial*. 2014. URL : <https://arxiv.org/pdf/1412.4031.pdf>.
- [8] W. A. PEARLMAN. *Digital signal compression : principles and practice*. University Press, 2011. URL : <https://doi.org/10.1017/CBO9780511984655>.
- [9] Dima PETRUK. 2016. URL : <https://stackoverflow.com/questions/22937589/how-to-add-noise-gaussian-salt-and-pepper-etc-to-image-in-python-with-opencv>.
- [10] THORLABS. *1.6 MP CMOS COMPACT SCIENTIFIC CAMERAS*. URL : https://www.thorlabs.com/newgroupage9_pf.cfm?objectgroup_id=13677.
- [11] Git Hub TOPICS. *jpeg-image-compression*. 2023. URL : <https://github.com/topics/jpeg-image-compression?l=python>.