

Rapport : Traitement de Signal et Image

Compte-rendu du projet d'étude

Compression d'images par transformée de Fourier

MAM 3

Année 2024 - 2025



Polytech Nice Sophia

Rédigé par :

Elsa Catteau, Charlotte Prouzet, Safia Zaari Jabri et Katell Nio

Sommaire

1	Introduction et objectifs	2
2	Plan de travail et carnet de bord	3
2.1	Lundi	3
2.2	Mardi	3
2.3	Mercredi	3
2.4	Jeudi	4
2.5	Vendredi	4
3	Explications théoriques	4
3.1	Transformée de cosinus discrète (DCT) d'une image	4
3.2	Compression	4
3.3	Décompression	5
3.4	Filtrage des hautes fréquences et débruitage	5
3.5	Taux de compression	5
4	Réalisations pratiques	6
4.1	Détermination de P	6
4.2	Une composante de couleur	6
4.3	Résultats	7
4.3.1	Comparaison erreur et taux de compression en fonction de la fréquence	7
4.3.2	Comparaison des images en fonction de la fréquence	9
4.3.3	Comparaison des images avec et sans quantification	9
5	Problèmes rencontrés	10
6	Conclusion	11

1 Introduction et objectifs

Durant cette semaine, nous avons appris qu'une image numérique peut être représentée sous forme d'un tableau multidimensionnel, correspondant à une décomposition en trois composantes de couleur RGB. Autrement dit, une image est formée d'un grand nombre de pixels chacun composé de trois canaux représentant respectivement l'intensité des couleurs rouge, verte et bleue. Ainsi la composition des ces trois canaux permettent de représenter toutes les couleurs, y compris les nuances de gris.

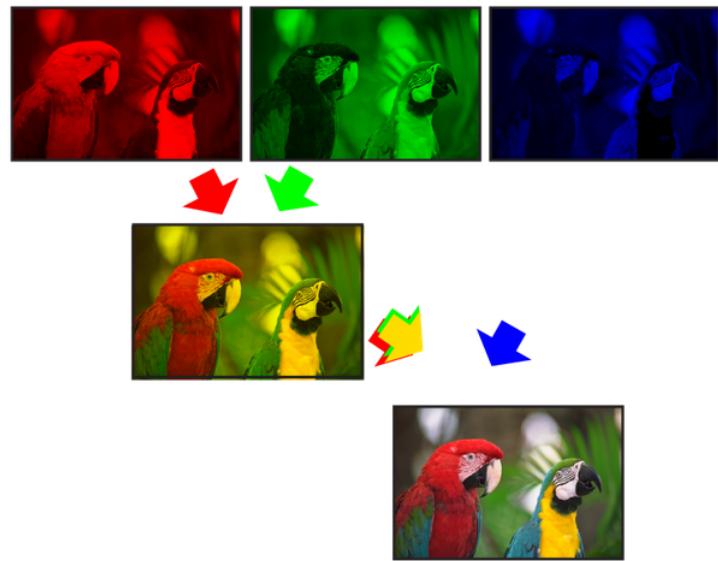


Figure 1: Source : Belgian Earth Observation.

L'objectif de cette semaine a donc été de développer un code Python capable de lire une image donnée, de la compresser puis de la décompresser en utilisant la transformée de Fourier. L'idée était de mettre en œuvre un algorithme permettant de réduire la taille des fichiers tout en maintenant une qualité visuelle satisfaisante.

2 Plan de travail et carnet de bord

Nous avions pour objectif de finir la réalisation de notre code avant le mercredi afin d'ensuite pouvoir l'optimiser tout en rédigeant le rapport. La présentation orale ayant lieu le vendredi, nous voulions nous consacrer pleinement à sa réalisation le jeudi. Pour finir, nous avions prévu de faire des groupes de deux, au moins pour l'initialisation ainsi que la compression.

Voici maintenant comment s'est déroulée notre semaine :

2.1 Lundi

- Matin :

- Découverte du projet avec lecture approfondie du cours donné (Tout le monde) ;
- Détermination de la matrice de passage P, à la main (Safia) ;
- Lecture de l'image et récupération de la matrice contenant les valeurs RGB de chaque pixel sur Python (Charlotte, Elsa, Katell) ;
- Tronquer l'image à des multiples de 8 en x et y (Charlotte, Elsa, Katell) ;

- Après-midi :

- Garder qu'une seule couleur pour avoir une matrice 2D (Elsa, Katell) ;
- Codage de la compression (Safia, Charlotte) ;
- Transformer les intensités en entiers entre 0 et 255, puis centrer pour se ramener entre -128 et 127 (Elsa, Katell) ;
- Réalisation de tests fonctionnels sur le code (Tout le monde) ;

2.2 Mardi

- Matin :

- Correction des erreurs et optimisation de la compression (Charlotte, Elsa) ;
- Codage de la décompression (Elsa) ;
- Faire l'initialisation pour les 3 couleurs différentes (Katell) ;
- Filtrage des hautes fréquences et débruitage (Charlotte, Safia) ;
- Recréation de la matrice tridimensionnelle à partir des 3 canaux de couleurs (Katell) ;
- Calcul des normes et de l'erreur (Elsa) ;
- Réalisation de tests fonctionnels sur le code (Tout le monde) ;

- Après-midi :

- Optimisation du code (Elsa, Katell) ;
- Tester le code sur différentes images et comparer les résultats (Charlotte) ;
- Rédaction du rapport (Tout le monde) ;
- Début des diapos pour la présentation (Safia) ;

2.3 Mercredi

- Après-midi :

- Réalisation de tests fonctionnels sur le code (Tout le monde) ;
- Rédaction du rapport (Tout le monde) ;
- Réalisation de la présentation (Tout le monde) ;

2.4 Jeudi

- Matin :

- Réalisation de la présentation (Tout le monde) ;
- Entraînement à l'oral et répartition du temps de parole (Tout le monde) ;
- Relecture du rapport (Tout le monde) ;

2.5 Vendredi

Nous avons réalisé notre soutenance de projet de 15 min.

3 Explications théoriques

3.1 Transformée de cosinus discrète (DCT) d'une image

On souhaite travailler avec les transformées de Fourier pour étudier et compresser l'information contenue dans une image. Par symétrisation et périodisation, on peut prolonger l'image infiniment dans les deux directions, périodique et paire. Une fonction périodique peut être décomposée en fonction cosinus et sinus et si la fonction est paire on utilise uniquement des fonctions cosinus.

On souhaite travailler sur des blocs de l'image de taille 8×8 , ce qui donne une période de 16 dans chaque direction. Avant d'appliquer la DCT (Transformée Discrète en Cosinus), qui est une restriction de la DFT (Transformée de Fourier Discrète) aux fonctions paires, on centre les valeurs de l'image autour de 0. Cela revient à soustraire 128 aux intensités de chaque pixel, pour obtenir des valeurs entre -128 et 127.

Si un bloc 8×8 est représenté par une matrice $M = (M_{i,j})$ de dimension 8×8 , la DCT de ce bloc s'exprime en dimension 2, pour $0 \leq k, l \leq 7$:

$$D_{k,l} = \frac{1}{4} C_k C_l \sum_{i=0}^7 \sum_{j=0}^7 M_{i,j} \cos\left(\frac{(2i+1)k\pi}{16}\right) \cos\left(\frac{(2j+1)l\pi}{16}\right)$$

où $C_0 = \frac{1}{\sqrt{2}}$ et $C_k = 1$ si $k > 0$. D est de la même taille que la matrice M , c'est-à-dire 8×8 .

La DCT-II permet d'obtenir un opérateur orthogonal, la transformation adjointe (transposée) donne alors la transformée inverse. La transformation de la formule précédente équivaut à un changement de base orthonormée, on peut donc la réécrire sous cette forme :

$$D = P M P^T$$

où P est une matrice orthogonale contenant les coefficients de la DCT.

Les entrées de la matrice D indiquent la fréquence des changements d'intensité lumineuse. Les coefficients dans le coin supérieur gauche représentent les basses fréquences (variations lentes), tandis que ceux en bas à droite correspondent aux hautes fréquences (variations rapides).

3.2 Compression

La compression passe par une étape de quantification, dans laquelle on ne va garder que les modes les plus importants (les basses fréquences). En effet, on considère les hautes fréquences comme du bruit. Afin de procéder à cette étape, nous effectuons la division, terme à terme, de notre matrice D (voir paragraphe précédent) par la matrice de quantification Q décrite de la manière suivante :

$$Q = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 13 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

Comme les termes en bas à droite de la matrice Q sont élevés, la division de D par Q permettra d'annuler les termes en bas à droite, relatifs aux hautes fréquences. L'essentiel des informations sera alors conservé dans une matrice creuse, dont le nombre de valeurs non-nulles sera nettement inférieur à celui de notre matrice initiale. Plus le nombre de valeurs non-nulles est réduit, plus le taux de compression de l'image est important, permettant ainsi une réduction de l'espace mémoire occupé par l'image.

3.3 Décompression

Une fois l'image compressée, nous procédons à l'étape de la décompression, qui vise à récupérer le visuel de notre image initiale. Nous appliquons alors l'inverse des opérations effectuées lors de la compression, à savoir la multiplication de chacun de nos blocs 8×8 (matrice D) par la matrice Q , puis le changement de base inverse à D :

$$\text{new_bloc} = P^T D P$$

où M correspond au bloc 8×8 après compression et décompression.

3.4 Filtrage des hautes fréquences et débruitage

De manière à enlever du bruit sur une image, nous pouvons ajouter une étape à notre code : celle du filtrage des hautes fréquences et du débruitage. Cette étape consiste à mettre à 0 tous les coefficients de la matrice D dont les indices (i, j) vérifient $i + j \geq F$, où F correspond à la fréquence de coupure que l'on définira.

Cette étape peut s'implémenter dans le code de deux manières :

- Soit en complément de l'étape de quantification, juste après avoir obtenu le nouveau bloc 8×8 résultant de la division terme à terme de D par Q .
(voir "CodeCourt")
- Ou alors seule, en supprimant la quantification (division par Q à la compression, puis remultipli-
cation par Q à la décompression).
(voir "CodeCourt_filtrage_seul")

3.5 Taux de compression

Dans ce projet, nous cherchons à compresser au maximum une image de manière à réduire l'espace mémoire occupé par cette dernière. Nous allons donc ajouter à notre code les calculs d'obtention du taux de compression et de l'erreur.

Afin d'obtenir le taux de compression de notre image, nous récupérons le nombre de coefficients non nuls de la matrice finale (après compression et décompression) ainsi que le nombre de coefficients présents dans notre matrice initiale. Le calcul du taux de compression (en pourcentage) est alors le suivant :

$$\text{taux de compression} = \left(1 - \frac{\text{nb de coeff non nuls dans la matrice finale}}{\text{nb total de coeff dans la matrice initiale}} \right) \times 100$$

Afin d'obtenir l'erreur relative, nous effectuons le calcul suivant :

$$\text{norme relative} = \left(\frac{\|\text{Matrice initiale} - \text{Matrice finale}\|}{\|\text{Matrice initiale}\|} \right) \times 100$$

En complément, nous pouvons aussi calculer la norme relative pour chacun des trois canaux de couleur. Le calcul reste alors le même, mais il est important de ne pas oublier de recentrer les valeurs contenues dans chacune des matrices entre 0 et 255.

4 Réalisations pratiques

4.1 Détermination de P

Avant de construire notre code, nous avons d'abord cherché à déterminer la matrice de passage P. Celle-ci est obtenue par identification à partir du produit matriciel de trois matrices, de la manière suivante : on considère les trois matrices suivantes $P = (p_{k,l})$, $M = (m_{k,l})$ et $Q = (q_{k,l})$ toutes de dimension 8×8 . On a dans un premier temps

$$MQ = \sum_{j=0}^7 m_{i,j} q_{j,l}$$

Puis dans un second temps

$$PMQ = \sum_{i=0}^7 p_{k,i} \times \left(\sum_{j=0}^7 m_{i,j} q_{j,l} \right) = \sum_{i=0}^7 p_{k,i} \times \sum_{j=0}^7 m_{i,j} q_{j,l} = \sum_{i=0}^7 \sum_{j=0}^7 p_{k,i} m_{i,j} q_{j,l}$$

Ensuite, par identification avec

$$D_{k,l} = \sum_{i=0}^7 \sum_{j=0}^7 \frac{1}{4} C_k C_l m_{i,j} \cos\left(\frac{(2i+1)k\pi}{16}\right) \cos\left(\frac{(2j+1)l\pi}{16}\right)$$

On en déduit que

$$p_{k,i} = \frac{1}{2} C_k \cos\left(\frac{(2i+1)k\pi}{16}\right)$$

$$q_{j,l} = \frac{1}{2} C_l \cos\left(\frac{(2j+1)l\pi}{16}\right)$$

Ainsi, pour $0 \leq k, l \leq 7$:

$$P = (p_{k,l})_{0 \leq k, l \leq 7} = \left(\frac{1}{2} C_k \cos\left(\frac{(2l+1)k\pi}{16}\right) \right)_{0 \leq k, l \leq 7}$$

$$Q = (q_{k,l})_{0 \leq k, l \leq 7} = \left(\frac{1}{2} C_l \cos\left(\frac{(2k+1)l\pi}{16}\right) \right)_{0 \leq k, l \leq 7} = P^T$$

avec $C_0 = \frac{1}{\sqrt{2}}$ et $C_k = 1$ si $k > 0$ (idem pour C_l)

4.2 Une composante de couleur

Tout d'abord, nous avons affiché l'image `pns.original.png` avec une seule couleur, en mettant les autres à 0. On a donc obtenu les trois images ci-dessous :



Figure 2: Image avec la composante bleue seulement



Figure 3: Image avec la composante verte seulement



Figure 4: Image avec la composante rouge seulement

4.3 Résultats

On a ensuite obtenue l'image compressée pour chaque canal de couleur mais aussi pour l'image globale. On a remarqué que lorsque nous enregistrons l'image compressée, elle utilise beaucoup moins d'espace de stockage que l'image originale. De plus, le taux de compression dépend de la fréquence de coupure F , plus celle-ci est élevée, plus le taux de compression diminue. Ainsi, plus le taux de compression est élevé, moins l'image utilise d'espace de stockage. Toutefois, plus la compression est élevée, plus on perd en netteté. On a donc souhaité déterminer la fréquence optimale, c'est-à-dire la fréquence pour laquelle l'erreur est faible mais le taux de compression reste élevé.

Une fois le code de la décompression implémenté pour chaque couleur, nous les avons rassemblés pour obtenir l'image finale.

4.3.1 Comparaison erreur et taux de compression en fonction de la fréquence

i) Comparaison des valeurs obtenues grâce au code avec quantification et filtrage (code "CodeCourt")

- Pns_original.png

Fréquence de coupure	Taux de compression (en %)	Erreur (en %)
$F = 2$	96.06	15
$F = 4$	88.51	9
$F = 6$	78.38	7
$F = 15$ (pas de filtrage)	44.42	25

- Pns_noisy.png

Fréquence de coupure	Taux de compression (en %)	Erreur (en %)
$F = 2$	97.10	20
$F = 4$	91.64	16
$F = 6$	84.39	15
$F = 15$ (pas de filtrage)	58.95	20

On remarque ici que le taux de compression ainsi que l'erreur relative sont plus importants sur l'image contenant du bruit que sur l'image originale. On peut expliquer cette différence par le fait que l'essentiel de l'information d'une image avec bruit se trouve dans les hautes fréquences (en bas à droite de la matrice). Au contraire, l'essentiel de l'information de l'image sans bruit se trouve dans les basses fréquences (en haut à gauche de la matrice). Ainsi, lorsque l'on applique le filtrage avec une fréquence de coupure F similaire pour les deux images (avec et sans bruit), les coefficients dans les hautes fréquences deviennent nuls. Pour une image avec bruit, les informations contenues dans les hautes fréquences s'annulent, et sont alors perdues. En revanche, pour une image sans bruit, il n'y a pas de pertes car l'essentiel des informations est contenu dans les basses fréquences. Ainsi, lorsque l'on retient la partie entière de la division de la matrice D par la matrice de quantification Q , les résultats diffèrent pour une image avec bruit et une image sans. Pour une image avec bruit, comme il n'y a pas beaucoup d'informations dans les basses fréquences, certains coefficients vont tomber à 0 tandis qu'ils resteront supérieurs à 1 pour une image sans bruit (puisque l'essentiel de son information est contenu dans les basses fréquences). Comme il y a plus de 0 dans la matrice de l'image avec bruit que dans celle sans bruit, on peut en déduire que le taux de compression est plus élevé. Concernant l'erreur, elle est aussi plus élevée pour une image avec bruit car il y a eu une perte d'informations plus importante que pour une image sans bruit.

- trefles.png

Fréquence de coupure	Taux de compression (en %)	Erreur (en %)
F = 2	95.5	31
F = 4	85.7	22
F = 6	72.9	16.5
F = 15 (pas de filtrage)	38.24	34

Sur l'image des trèfles, le taux de compression est inférieur aux images précédentes, tandis que l'erreur est supérieure. Cela s'explique par le fait qu'il y a davantage de "texture" sur cette image que sur les précédentes. Il y a donc davantage d'informations (des valeurs plus grandes) dans les basses fréquences, ce qui implique l'apparition de moins de 0 lorsque l'on divise chacun de nos blocs par la matrice de quantification Q. Le taux de compression est donc plus petit, tandis que l'erreur augmente en raison de la perte de plus d'information.

- vagues.png

Fréquence de coupure	Taux de compression (en %)	Erreur (en %)
F = 2	96.32	1.5
F = 4	91	2.3
F = 6	82.37	4.9
F = 15 (pas de filtrage)	48.79	27

Au contraire, sur l'image des vagues, le taux de compression est supérieur aux images précédentes, tandis que l'erreur est plus faible. En effet, l'image est moins texturée. Les valeurs stockées dans les basses fréquences sont donc plus petites, ce qui génère plus de 0 lorsque l'on divise chacun de nos blocs par la matrice de quantification Q. Le taux de compression est donc plus grand, tandis que l'erreur diminue.

- ii) Comparaison entre les valeurs obtenues grâce au code avec quantification et filtrage (`code_Court`) et les valeurs obtenues grâce au code avec filtrage seul (`CodeCourt_filtrage_seul`)

- Pns_noisy.png (Code avec quantification et filtrage)

Fréquence de coupure	Taux de compression (en %)	Erreur (en %)
F = 2	97.10	20
F = 4	91.64	16
F = 6	84.39	15
F = 15 (pas de filtrage)	58.95	20

- Pns_noisy.png (Code avec filtrage seul)

Fréquence de coupure	Taux de compression (en %)	Erreur (en %)
F = 2	96.62	19.5
F = 4	88.96	15.8
F = 6	76.53	14.4
F = 15 (pas de filtrage)	26.98	0.4

Si l'on compare maintenant le code complet (quantification avec filtrage) et le code avec filtrage seul pour une image bruitée (Pns_noisy.png), on observe que le taux de compression ainsi que l'erreur diminuent du premier code au second. Ce résultat est cohérent puisque dans le code dit "complet", l'étape de quantification vient en complément du filtrage, permettant l'apparition de nouveaux 0 par rapport au code avec filtrage seul. L'erreur diminue car la perte d'informations est moins grande, et finit par tendre vers 0 lorsque $F > 14$. En effet, les blocs étant chacun de taille 8×8 , la matrice répertoriant la somme des indices i et j est la suivante :

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{pmatrix}$$

La méthode du filtrage avec la fréquence de coupure F consiste à mettre à 0 tous les coefficients dont l'indice $i + j \geq F$. Ainsi, comme $i + j \leq 14$, pour tout i et pour tout j tels que $0 \leq i, j \leq 7$, alors pour des valeurs de $F > 14$, il n'y aura ni filtrage des hautes fréquences, ni débruitage.

4.3.2 Comparaison des images en fonction de la fréquence

Nous avons ensuite comparé notre image à l'originale pour différentes fréquences. Voici nos résultats obtenus :



Figure 5: Image Originale



Figure 6: Image obtenue avec une fréquence de 3



Figure 7: Image obtenue avec une fréquence de 6

De loin, ces images peuvent sembler similaires. Regardons de plus près, grâce à un zoom, la différence entre ces images :



Figure 8: Image Originale



Figure 9: Image obtenue avec une fréquence de 3



Figure 10: Image obtenue avec une fréquence de 6

Analyse de nos résultats : On remarque donc qu'une fréquence plus élevée permet d'avoir une image plus nette. Cela est dû au fait que plus la fréquence est basse, plus le nombre de zéros dans la matrice compressée est important et donc le nombre d'informations perdues est élevé lors de la décompression, ce qui rend l'image floue. Cependant, une fréquence plus haute diminue le taux de compression.

4.3.3 Comparaison des images avec et sans quantification

Nous avons ensuite voulu tester pour des images différentes une compression avec quantification + filtrage à une compression avec seulement filtrage. Voici nos résultats obtenus pour une fréquence de 6 :

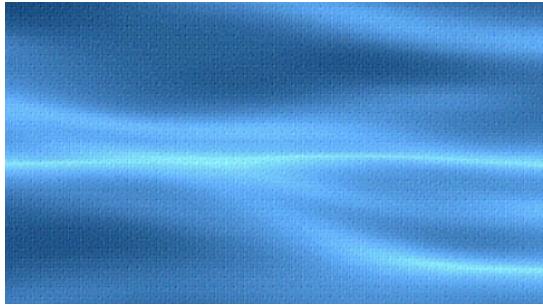


Figure 11: Image avec quantification + filtrage



Figure 12: Image avec filtrage seulement



Figure 13: Image avec quantification + filtrage



Figure 14: Image avec filtrage seulement



Figure 15: Image bruitée avec quantification + filtrage



Figure 16: Image bruitée avec filtrage seulement

Analyse de nos résultats : On remarque donc que les images sont plus nettes avec une compression faite par filtrage seul. Cela s'explique par le fait que la quantification divise les valeurs de la DCT, ce qui introduit une perte d'information supplémentaire, car les petites variations (particulièrement dans les hautes fréquences) sont supprimées ou fortement réduites. On le voit d'autant plus avec la vague qui renvoie une image très similaire à l'original pour une compression avec filtrage seulement. Néanmoins, une compression avec quantification permet un meilleur taux de compression.

5 Problèmes rencontrés

Durant cette semaine de projet, nous avons rencontré certaines difficultés, notamment dans le calcul de l'erreur. En effet, nous n'arrivions pas à comprendre pourquoi, lorsqu'on faisait une compression avec quantification + filtrage, l'erreur augmentait de nouveau à partir d'une certaine valeur de fréquence. On en a finalement déduit que c'était dû à l'erreur accumulée sur les hautes fréquences, car lorsque F augmente, plus de coefficients haute fréquence (qui sont souvent faibles en valeur) sont inclus, mais leur quantification amplifie les erreurs dues à l'arrondi.

6 Conclusion

Durant ce projet, nous avons exploré la compression d'image en utilisant la transformée en cosinus discrète (DCT) appliquée à des blocs 8×8 . L'objectif était de comprendre l'impact du filtrage des hautes fréquences et de la quantification sur la qualité de l'image reconstruite et le taux de compression. Nous avons mesuré la qualité via l'erreur relative en norme L2 et évalué les compromis entre qualité et compression. Les résultats montrent que la compression d'image repose sur un équilibre délicat :

Filtrage des hautes fréquences : Permet d'éliminer les détails fins et le bruit, réduisant ainsi la taille des données sans altérer significativement les informations essentielles.

Quantification : Réduit davantage la taille en approximant les coefficients, mais peut introduire des erreurs perceptibles, surtout dans les hautes fréquences.

Finalement, le choix entre les deux dépend de la demande. Pour une image nette, il vaut mieux opter pour un filtrage seul, mais si l'objectif est une compression efficace, la combinaison des deux est préférable. Il est aussi possible de faire varier la valeur de la fréquence afin d'obtenir un bon compromis entre la qualité de l'image et sa compression. Dans notre cas, nous avons observé le meilleur compromis pour une fréquence de 6, montrant ainsi une image nette avec une erreur relative faible.