Projet MAM3 –
Compression d'image par transformée de Fourier

CHOUKROUN Julien

DAVID Samy

GOURDON Jessica

SAGNES Luc

BOULBE Cédric

AUROUX Didier

Année 2020-2021

Sommaire

I. Introduction

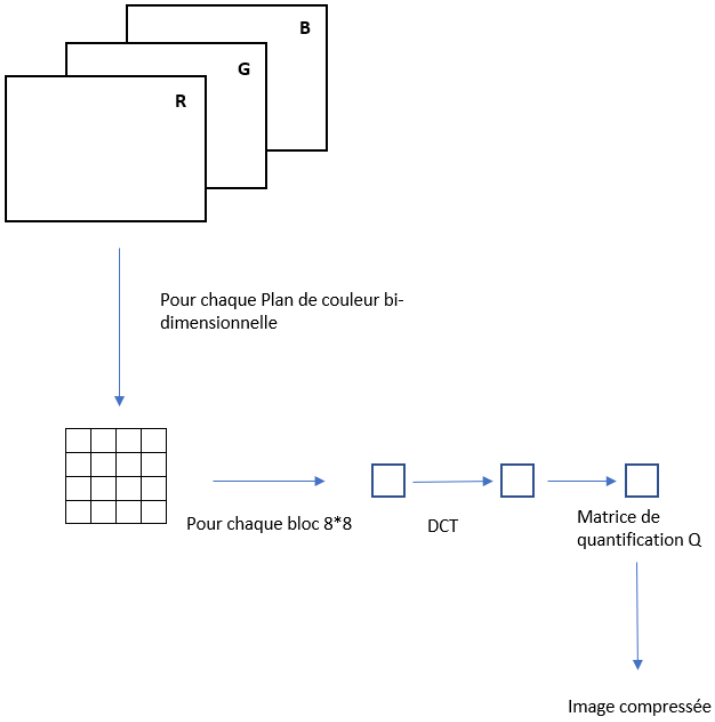
Les images font partie des éléments qui constituent le plus gros d'une page Web. Le format et la taille des illustrations d'un site Web ont en outre un grand impact sur les temps de chargement et, de ce fait, aussi sur le classement dans les moteurs de recherche.
La compression d'image est une application de la compression de données sur des images numériques. Cette compression a pour utilité de réduire la taille du fichier afin de pouvoir l'emmagasiner sans occuper beaucoup d'espace ou le transmettre rapidement.
La compression d'image peut être effectuée avec perte de données ou sans perte. La compression sans perte est souvent préférée dans les schémas, dessins techniques, icônes, bandes dessinées. La compression avec perte, est utile pour les transmissions à bas débit, mais dégrade la qualité de l'image restituée.
L'objectif ici est d'appliquer la compression d'image avec perte de données grâce à la méthode du codage par transformation. Pour ce faire, nous utilisons la transformée de cosinus discrète.

Voici notre plan de travail :

Jour	Matin	Après-Midi
Lundi	Réunion et explication Formation du groupe Téléchargement de python et librairies	Troncage de l'image en multiple de 8 Conservation d'une composante Transformation des intensités en entier Début du rapport
Mardi	Définition de la matrice de passage en fréquentiel de la DCT2 P	Application matrice de quantification Filtrer les hautes fréquences Stockage des nouveaux blocs dans la matrice compresser Taux de compression Décompression
Mercredi	Comparaison de la matrice obtenue avec la matrice originale	Continuer le rapport
Jeudi	Tests Finaux	Terminer le rapport Faire la présentation

II. Rappels

La transmission des informations est d'autant plus facile que le nombre d'informations est faible. Il est donc avantageux de réduire la taille d'un fichier pour l'envoyer plus rapidement et plus facilement. Une technique employée dans de nombreux domaines est la compression des informations qui permet en effet, une réduction importante de la quantité de données.
Voici l'algorithme permettant la compression d'une image :



Une fois après avoir implémenté et codé la compression, il n'est pas difficile d'aborder la décompression. Il suffit de faire le chemin inverse de la compression en appliquant la DCT inverse. La décompression est un processus plus rapide que la compression car il n'y a plus de divisions à arrondir et de termes à négliger.

III. Explication du code

1) Initialisation

Dans un premier temps nous devons lire l'image, pour ce faire nous avons utilisé le module PIL. Le but était de partir d'une image de stocké dans une grande matrice à 3 dimensions, et d'attribuer à chaque pixel sa composante en rouge, vert et bleu. Il faut ensuite redimensionner l'image afin d'avoir une image qui a pour longueur et largeur un multiple de 8 pour pouvoir traiter l'image par blocs de 8 par 8. Par la suite nous avons créée 3 matrices contenant les composantes de chaque couleur. Ces composantes étant des entiers compris entre 0 et 255 nous avons réalisé de soustraction afin de les ramener à des entiers compris entre -128 et 127. En effet pour appliquer la transformée discrète de Fourier il nous faut des valeurs de l'image centrées autour de 0.
A partir de la formule de D nous avons pu isoler les coefficients de la matrice de changement de base pour pouvoir effectuer la transformation de Fourier discrète.

2) Compression

Pour chaque matrice de couleurs, nous avons créé des sous-matrices correspondants à des blocs de 8x8 pixels. Nous avons effectué notre changement de base sur chaque bloc de matrice de couleurs. Puis nous avons divisé ses sous-matrices par la matrice de quantification Q de la norme de compression JPEG. Le but étant de ne garder que les fréquences les plus importantes (souvent les plus petites). Nous obtenons des valeurs réelles, il ne nous restait alors plus qu'à prendre leur partie entière.
On peut également rajouter un filtrage de haute fréquence où l'on applique la valeur 0 à une grande partie des coefficients de haute fréquence pour réduire au maximum le bruit.
On rassemble ensuite les blocs 8x8 pour obtenir notre matrice compressée. Enfin nous avons calculé notre taux de compression, expérimentalement nous avons trouvé un taux de 94% de compression.

3) Décompression

Pour effectuer la décompression, nous avons réalisé la manipulation inverse à savoir : pour chaque bloc 8x8, nous multiplions par la matrice Q terme à terme. On peut noter que les coefficients nuls resteront nuls. Nous avons, par conséquent, une diminution de la qualité par rapport à l'image initiale. Nous avons donc effectué le changement de base inverse. Puis, nous avons réassemblé la matrice finale qui sera par la suite décompressée et débruitée.

4) Post-processing

Nous avons ainsi pu comparer nos deux matrices : la matrice initiale et la matrice finale que nous avons créé. Pour cela, nous nous sommes servis de la norme relative. Expérimentalement, nous avons obtenu une norme de 0,07. L'écart entre les 2 matrices étant faibles, nous pouvons en conclure que nous n'avons perdu que peu d'informations.

IV. Analyse et Limite

1) Analyse

Nous avons comparé notre résultat avec une DCT implémentée par une libraire toute faite qui est scipy.fftpack. Cette dernière réalise la compression et la décompression toute seule. Nous pouvons observer que l'image finale obtenue par la DCT implémentée par scipy est de meilleure qualité que l'image finale obtenue par notre DCT expérimentale. Cela est sûrement dû à une meilleure précision de la librairie scipy.

Image en noir et blanc :

Image décompressée

Image originale

Pour cette dernière nous obtenons un taux de compression de 97%.

Image basse fréquence :

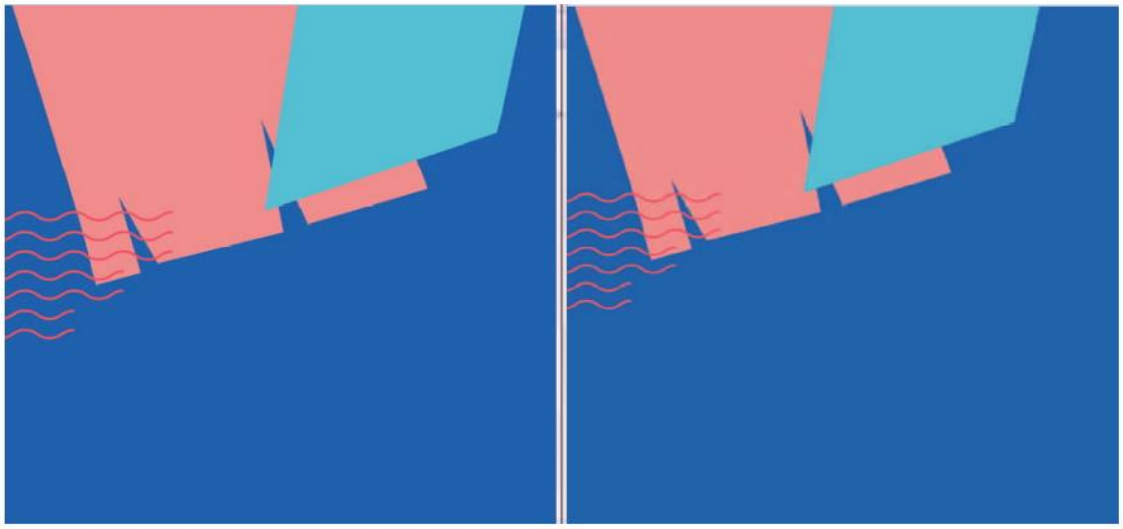


Image décompressée Image originale

Pour l’image en basse fréquence nous obtenons également un taux de compression de 97%.

Image haute fréquence :

Image décompressée Image originale

Pour l’image en haute fréquence on obtient un taux de compression de 87%.
On suppose qu’on obtient un taux de compression plus faible du fait que nous supprimons les hautes fréquences ce qui débouche sur une perte de données.

2) Limites

Durant ce projet nous nous sommes heurtés a des difficultés. En effet nous avons pour certain du nous familiariser avec un nouveau langage de programmation : Python.
En travaillant avec des tableaux numpy nous avons vu de nouvelle fonction spécifique qui ont pu perturber, par exemple une fois P calculé pour vérifier nous avons réalisé l’opération P*tP et nous ne

retombions pas sur la matrice identité car avec les tableaux numpy, il nous fallait utiliser numpy.dot autrement nous avions une multiplication terme à terme.
Quand nous avons commencé à avoir des résultats nous nous sommes retrouvés avec une image qui avait à certains endroits des pixels de couleurs qui ne correspondaient pas à l'image. Cela était dû à l'utilisation de Image.fromarray.
Dans le cas de valeurs débordants de l'intervalle [0,255] tel que -2 il fallait veiller à ce qu'il soit traité comme un 0 et non, par périodicité, comme un 254. Il a donc fallu que l'on vérifie si les valeurs dépassaient cet intervalle, dans ce cas si leurs valeurs étaient en dessous de 0, elles prenaient la valeur 0 et si elles étaient au-dessus de 255 elle prenait la valeur 255.

V. Conclusion

Le principe de compression d'image JPEG est basé sur des principes mathématiques simples et permet d'obtenir un taux de compréhension assez important dont les modifications ne sont pas décelées par l'humain sans effectuer un zoom sur l'image. Cette compression est donc très utilisée.
Notre projet restitue bien une image décompressée et respecte bien les consignes données. Différentes améliorations auraient pu être envisagées telle qu'une interface plus sophistiquée mais en raison du délai nous n'avons pu nous consacrer à une tâche pareille.
Ce projet nous a malgré tout permis de développer notre esprit d'équipe, nous avons pu acquérir de nouvelles compétences telle que la maîtrise du langage Python (ou plutôt la découverte de ce langage).
Le projet nous a enfin montrer l'application que pouvait avoir les mathématiques dans la vie courante, grâce à la transformée de Fourier.