Compression d'images
TP1 : JPEG (8h)
Python

Les fichiers nécessaires sont accessibles sur Moodle.

L'évaluation se fera sur la base du travail en séance et d'un compte-rendu détaillé délivré à la fin de la dernière séance. Ce compte-rendu devra comporter :

- un exposé clair de la problématique,
- un détail des voies explorées au cours des TPs,
- une synthèse présentant clairement les éléments validés, les éléments restant à tester, ceux à améliorer et ceux éventuellement manquants,
- une archive numérique contenant l'ensemble des codes produits.

Réalisation d'un codeur/décodeur JPEG simplifié

JPEG est l'acronyme de Joint Photographic Experts Group. C'est une norme définissant le format d'enregistrement et l'algorithme de décodage pour une représentation numérique compressée d'une image. JPEG normalise uniquement l'algorithme et le format de décodage. Le processus d'encodage est quant à lui laissé libre à la compétition des industriels et des universitaires. La seule contrainte est que l'image produite doit pouvoir être décodée par un décodeur respectant le standard.

Le format JPEG permet de compresser les données d'une image. Il existe un processus de compression sans pertes (ou réversible) qui permet de revenir exactement aux valeurs de l'image avant encodage. Son taux de compression est inférieur à celui d'une compression avec pertes (ou irréversible). Dans les cas où la taille de l'image est un critère plus important que la qualité, on privilégiera une compression avec pertes.

1 Structure d'une image (non compresée)

L'image de départ est un tableau de M sur N pixels. Pour une image en niveau de gris, chaque pixel est codé sur un octet (valeurs comprises entre 0 et 255).

Pour une image couleur, chaque pixel est codé sur trois octets, qui représentent les intensités des trois composantes couleur : le rouge, le vert et le bleu.

Conversion en luminance/chrominance

Une autre représentation que le RGB est souvent utilisé : le YCbCr, qui correspond à la luminance Y (intensité du pixel en niveau de gris) et deux chrominances (une rouge Cr et une bleue Cb). L'image est alors constitué par trois tableaux d'octets, associés respectivement aux trois grandeurs Y, Cr, Cb. Les formules permettant de passer d'une représentation à l'autre sont les suivantes :

$$
\begin{aligned}
Y &= 0.299R + 0.587G + 0.114B \\
Cb &= -0.1687R - 0.3313G + 0.5B + 128 \\
Cr &= 0.5R - 0.4187G - 0.0813B + 128 \\
R &= Y + 1.14020(Cr - 128) \\
G &= Y - 0.34414(Cb - 128) - 0.71414(Cr - 128) \\
B &= Y + 1.77200(Cb - 128)
\end{aligned}
$$

Remarque : R, G, B sont compris entre 0 et 255. Ces formules peuvent donner les valeurs YCbCr qui sortent de cet intervalle, et qui devront être saturées si nécessaire afin de les garder comprises entre 0 et 255.

Deux formats différents sont fréquemment utilisés :

• dans le format le plus simple, les trois tableaux Y, Cr, Cb ont les mêmes dimensions, M et N, égales aux dimensions de l'image originale.

• un format plus efficace en terme de compression (et donc plus souvent utilisé) consiste à sous-échantillonner les deux signaux de chrominance d'un facteur 2 (ou 4). Plus précisément, avec le facteur 2, le tableau Y reste de taille [M,N] mais les tableaux Cr et Cb sont de dimension [M/2,N/2] (la valeur de chrominance est associée à quatre pixels voisins).

2 Encodage d'une data unit

Le codeur JPEG travaille sur des data unit qui sont des blocs de taille 8x8 d'une image. Chaque composante est découpée en carrés de 8x8 pixels. Si les dimensions de la composante ne sont pas des multiples de 8, l'image est complétée par duplication de la dernière ligne (ou colonne) jusqu'à obtenir le multiple de

8 immédiatement supérieur. Chaque carré 8x8 est ensuite traité indépendamment, en décrivant l'image de gauche à droite et de haut en bas.

2.1 Centrage

Les échantillons des carrés 8x8 sont des nombres compris entre 0 et 255. La première opération à réaliser est un centrage afin d'obtenir des valeurs comprises entre -128 et 127 (en retirant 128 à chaque valeur). Les valeurs centrées seront notées $f(i,j)$ avec $i,j \in [0,\ 7]$.

2.2 Transformée en cosinus

Après avoir réalisé le centrage, il faut calculer la transformée en cosinus discrète (DCT) de chaque carré 8x8. Cette transformation donne une nouvelle matrice 8x8 de coefficients, appelée $F(u,v)$, $u,v \in [0,\ 7]$. Cette transformée est une variante de la transformée de Fourier. Elle décompose un bloc, considéré comme une fonction numérique à deux variables, en une somme de fonctions cosinus oscillant à des fréquences différentes. Chaque bloc est ainsi décrit en une carte de fréquences et en amplitudes plutôt qu'en pixels et coefficients de couleur. Pour le décodage, on peux retrouver les $f(i,j)$ à partir de la transformée 2D inverse des $F(u,v)$. Les équations de cette transformée ainsi que de la transformée inverse :

$$F(u,v) = C_u C_v \sum_{i=0}^{7} \sum_{j=0}^{7} f(i,j) cos\left(\frac{(2i+1)u\pi}{16}\right) cos\left(\frac{(2j+1)v\pi}{16}\right) \tag{1}$$

$$f(i,j) = \sum_{u=0}^{7} \sum_{v=0}^{7} C_u C_v F(u,v) cos\left(\frac{(2i+1)u\pi}{16}\right) cos\left(\frac{(2j+1)v\pi}{16}\right) \tag{2}$$

avec $C_0 = \frac{1}{\sqrt{(8)}}$ et $C_u = \frac{1}{2}$ si $u \neq 0$.

L'application de la DCT est une opération théoriquement sans perte d'informations : les coefficients initiaux peuvent être retrouvés en appliquant la DCT inverse au résultat de la DCT.

Ces transformation en deux dimensions sont séparables car elles peuvent être réalisées en appliquant des transformations en une dimension, successivement sur les lignes et les colonnes.

2.3 Quantification

La quantification est l'étape qui permet de gagner le plus de place (la DCT n'effectue aucune compression). La DCT a retourné, pour chaque bloc, une matrice de 8x8 nombres. La quantification consiste à diviser point à point cette matrice par une matrice de quantification également 8x8. Soit Q la matrice de quantification. Le bloc 8x8 après compression sera obtenu par :

$$\hat{F}(u,v) = round\left(\frac{F(u,v)}{Q(u,v)}\right) \tag{3}$$

Le but est d'atténuer les hautes fréquences car l'oeil humain y est très peu sensible. Ces fréquences ont des amplitudes faibles, et elles sont souvent ramenées à 0 après la quantification. Ces coefficients sont situés dans la matrice en bas à droite. Le but va être de ne garder que quelques informations essentielles (concentrées dans le coin en haut à gauche) pour représenter le bloc. Le reste de la matrice sera essentiellement composée de 0, ce qui va permettre d'utiliser un codage RunLength afin de gagner de la place.

Les matrices de quantification sont les éléments centraux de la compression avec pertes parce que c'est la quantification qui permet de régler les pertes, et donc le taux de compression. En toute rigueur, ces matrice devraient être calculées pour chaque image, en tenant compte du taux de compression désiré et des propriétés de l'image et de l'oeil. En pratique, ce calcul est complexe, et l'on pourra dans ce projet utiliser les matrices précalculées.

Pour la quantification d'une composante RGB ou de la composante luminance, la matrice Q est :

16	11	10	16	24	40	51	61
12	12	14	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

Pour la quantification d'une chrominance, la matrice Q est :

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Remarque : dans un premier temps du projet, faire la quantification en utilisant ces matrices de quantification. Ensuite, tester la compression en modifiant les matrices de quantification avec un facteur de qualité. Soit fq le facteur de qualité (entre 1 et 100), définir s : si $fq < 50$, $s = 5000/fq$, sinon $s = 200 - 2 * fq$ et puis calculer la nouvelle matrice de quantification : $Q2 = floor((s * Q + 50)/100)$.

2.4 Parcours en zigzag, RLC, et codage de Huffman

Comme dit dans la section précédente, après la quantification, beaucoup de coefficients de la matrice $\hat{F}$ sont nuls et ils localisés en bas à droite de cette matrice. Le nombre de bits moyen de ces coefficients est donc très réduit. Afin d'exploiter cette propriété, le bloc 8x8 sera lu avec un parcours zigzag (Figure 4) afin de construire de longues plages de 0 (afin d'optimiser au mieux l'utilisation d'un RLC sur le symbole 0).

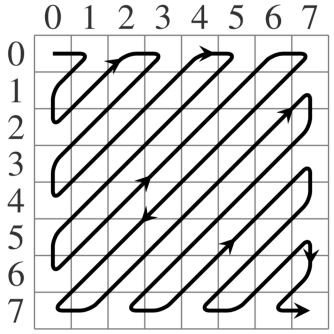

FIGURE 1 – Ré-ordonnancement Zigzag.

Une fois le vecteur contenant les 64 valeurs du bloc 8x8 construit, il faut effectuer un codage RunLength pour obtenir le vecteur Vrlc : dès qu'une plage de 0 est détectée, il faut préciser la longueur de cette plage, précédée du nombre 257.

Une fois les vecteurs codant le RLC codés, il faut effectuer un codage de Huffman pour les Vrlc.

Remarque

En pratique, dans un bloc 8x8, on distingue deux types de coefficient : le coefficient $\hat{F}(0,0)$ est appelé coefficient DC, et les autres 63 coefficients sont appelés coefficients AC (qui sont très faibles ou nuls après quantification). Les coefficients DCs et ACs sont codés différemment.

DC : la valeur DC d'un bloc correspond à sa valeur moyenne. Elle est la première valeur du bloc, située à la position (0, 0). En faisant l'hypothèse qu'elle est généralement voisine de celles des blocs voisins (vrai sauf en cas de changement brusque de couleurs ou de zones), la différence entre deux valeurs DC de blocs voisins de la même composante (luminance ou chrominance) est plutôt faible. Ainsi, au lieu d'encoder les valeurs DC issues de la quantification, on préfère encoder cette différence. On encode donc les DCs par le codage différentiel (DPCM Differential Pulse Code Modulation) et le code de Huffman.

AC : Les coefficients AC sont donc les 63 valeurs restantes dans le bloc 8 x 8. L'étape de quantification et le ré-ordonnancement ont eu pour but de mettre à 0 les hautes fréquences. On encode les ACs par le codage RLE et le code de Huffman.

3 Décodage d'une data unit

Le décodage d'une data unit va défaire tous les étapes précédentes les unes après les autres. La première étape consistera à décoder le code de Huffman de la data unit afin de reconstruire le vecteur Vrlc. De là, on peut reconstruire les 64 valeurs de bloc 8x8, puis inverser l'opération de quantification et de DCT.

4 Implémentation d'un codeur/décodeur JPEG simplifié

Les blocs nécessaires à la mise en oeuvre du codeur/décodeur simplifié détaillé précédemment sont les suivants :

- Découpage en bloc de taille 8x8
- Centrage, DCT et quantification de chaque bloc
- RunLength coding
- Codage de Huffman
- Les fonctions inverses de chacun des points précédents

Dans un premier temps, commencer par travailler avec des images en noir et blanc (encoder les DCs et ACs de la même manière). Par la suite, si tous les blocs fonctionnent correctement, passer à une image couleur en vérifiant d'abord avec la représentation RGB puis la représentation luminance/chrominances. Ensuite, encoder les DCs et ACs en utilisant deux tableaux de codage différentes (voir le remarque).

Conseil : Calculer et afficher le résultat de différentes étapes (les résultats du DCT et de la quantification de différents blocs).

Calculer le taux de compression, les erreurs, le PSNR.