TP - Codages JPEG et MPEG

Dans ce TP de Compression, Streaming, Interaction Vidéo (CSIV), vous serez amenés à effectuer un codage JPEG et un codage MPEG d'une image. Un rendu des codes est attendu pour le **lundi 20 novembre 2023**. Le TP est divisé en deux parties : la première pour le codage JPEG qui regroupe les exercices 1 à 5 et la seconde pour le codage MPEG qui comprend les exercices 6 à 8. Pour toutes les fonctions que vous aurez à compléter, les prototypes sont indiqués en commentaires dans l'entête de chaque fichier. Toutes les images de test utilisées dans le TP sont contenues dans le fichier `Donnees_TP_MPEG-2.mat` afin de vous laisser la surprise des visuels pendant les tests des différentes fonctions à implémenter.

Partie 1 - Compression et décompression JPEG

JPEG (sigle de *Joint Photographic Experts Group*) est une norme qui définit le format d'enregistrement et l'algorithme de décodage pour une représentation numérique compressée d'une image fixe. Les différentes étapes de codage et décodage sont répertoriées dans le schéma suivant :

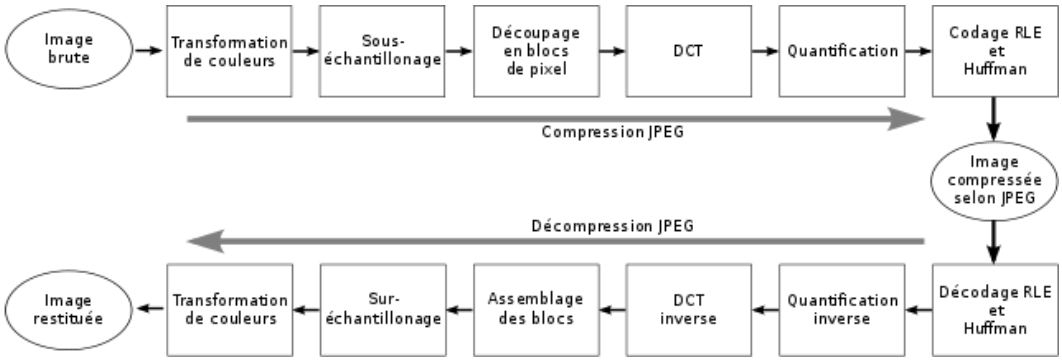

FIGURE 1 – Les différentes étapes du codage et décodage JPEG pour une image couleur.

Pour des questions de temps, nous nous intéresserons seulement à des images en niveaux de gris, et le codage RLE + Huffman sera remplacé par un simple codage entropique, pour donner une idée du poids final de l'image compressée. Libre à vous de rajouter ces étapes en fonction de votre temps, mais ce n'est en rien obligatoire pour le rendu.

Exercice 1 - Codage entropique d'un signal

Une fois n'est pas coutume, la première étape à réaliser pour la compression sera le codage entropique (soit la dernière étape du schéma de la FIGURE 1). Il s'agit ici de calculer le nombre de bits moyen le plus petit possible afin de pouvoir transmettre le signal sans perte, à partir de la formule de calcul de l'entropie :

$$H(I) = -\sum_{i=1}^{n} f_i \log_2(f_i) \tag{1}$$

où les f_i représentent les différentes fréquences d'apparition des niveaux de gris dans l'image. Pour ce faire, complétez la fonction `CodageEntropique` qui permettra de calculer l'entropie et le poids d'un signal quelconque. En effet, la fonction `entropy` de Matlab ne fonctionne que sur des images en niveaux de gris mais nous aurons besoin par la suite de généraliser cela à un vecteur à la fin des différentes étapes du codage JPEG. Pour tester la fonction, lancez le script `Exercice_1` (le résultat est comparé avec la fonction `entropy` de Matlab seulement ici pour vérifier qu'il y a bien égalité), et quelqu'un apparaîtra pour vous informer si vous avez réussi ou non. **Attention :** Cette fonction doit être généralisable à des coefficients quelconques (pas forcément entre 0 et 255).

Exercice 2 - Transformée en Cosinus Discrète par blocs d'une image

Si l'on revient dans le bon ordre des étapes de la compression, la première est la Transformée en Cosinus Discrète (ou DCT en anglais) 2D par blocs de taille 8×8. Cette transformée prend la forme suivante :

$$\mathrm{DCT}_{2D}(i,j) = \frac{C(i)C(j)}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} \mathrm{bloc}(x,y) \cos\left[\frac{(2x+1)i\pi}{16}\right] \cos\left[\frac{(2y+1)j\pi}{16}\right] \tag{2}$$

où les coefficients $C(i)$ et $C(j)$ valent $1/\sqrt{2}$ en 0 et 1 sinon. Cette DCT 2D peut être réécrite comme la composition de deux DCT 1D, ce qui donne :

$$\mathrm{DCT}_{2D}(i,j) = \frac{C(j)}{2} \sum_{y=0}^{7} \left(\frac{C(i)}{2} \sum_{x=0}^{7} \mathrm{bloc}(x,y) \cos\left[\frac{(2x+1)i\pi}{16}\right] \right) \cos\left[\frac{(2y+1)j\pi}{16}\right] \tag{3}$$

Cette séparation permet d'effectuer un calcul plus rapide des différents coefficients de la transformée comme vu en cours. Complétez la fonction `DCT2DParBlocs` en utilisant, pour chaque bloc de taille 8×8, la fonction `dct2` déjà implémentée dans Matlab. Le script `Exercice_2` permet de tester la fonction sur une image. Cette image en question a subi une DCT inverse (ou IDCT), ce qui fait qu'en lui appliquant une DCT on retrouve l'image initiale. **Attention :** Cette fonction servira aussi pour la DCT inverse, d'où la variable `sens` en entrée.

Bonus : Complétez la sous-fonction `DCT2Rapide` qui permet d'effectuer la DCT sur un bloc de taille 8×8 en utilisant les redondances des coefficients vues en cours. Ajoutez-là ensuite dans la fonction `DCT2DParBlocs` en possibilité supplémentaire par rapport à `dct2`.

Exercice 3 - Quantification par blocs d'une image

Une fois la DCT effectuée, on s'intéresse à la quantification, étape irréversible car consistant à diviser les valeurs contenues dans la DCT et ensuite prendre l'arrondi. Dans le cas du codage JPEG, la quantification n'est pas uniforme : chaque coefficient d'un bloc va être divisé par une valeur différente. Pour obtenir cette quantification par bloc, on part de la table de quantification de la luminance donnée par la norme, et correspondant à un facteur de qualité $F_Q = 50$:

$$T^q_{(l)}(50) = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix} \tag{4}$$

On remarque que les coefficients de haute fréquence sont beaucoup plus atténués que ceux des basses fréquences. Ceci est dû au fait que l'œil humain y est moins sensible. Pour un facteur de qualité quelconque (compris entre 0 et 97), la nouvelle table de quantification s'obtient de la manière suivante :

$$T^q_{(l)}(F_Q) = \begin{cases} \left\lfloor \dfrac{(100 - F_Q) \times T^q_{(l)}(50)}{50} \right\rceil & \text{si } F_Q \in]50, 97] \\ \left\lfloor \dfrac{50 \times T^q_{(l)}(50)}{F_Q} \right\rceil & \text{si } F_Q \in]0, 50[\end{cases} \tag{5}$$

Une fois la table de quantification calculée, pour obtenir le bloc de l'image quantifiée bloc_Q, on prend à nouveau l'arrondi du rapport entre le bloc d'origine et la table $T^q_{(l)}(F_Q)$:

$$\mathrm{bloc}_Q = \left\lfloor \frac{\mathrm{bloc}}{T^q_{(l)}(F_Q)} \right\rceil \tag{6}$$

Complétez la fonction `QuantificationDCT` qui, à partir d'un facteur de qualité, calcule la table de quantification associée (pour récupérer cette table, complétez la sous-fonction `ChoixTableQuantification` dans le même fichier) et effectue la quantification de la DCT bloc par bloc. De même que précédemment, le script `Exercice_3` permet de tester la fonction sur une image qui a subi une quantification inverse. **Attention :** Cette fonction servira aussi pour la quantification inverse, d'où la variable `sens` en entrée. La variable `canal` permet quant à elle de changer de table pour un canal de chrominance ou de résidu (voir cours MPEG-2 et suite du TP).

Exercice 4 - Compression d'une image au format JPEG

Toutes les étapes de compression ont été réalisée. Complétez maintenant la fonction `CompressionJPEG` qui doit reprendre toutes les fonctions utilisées dans les exercices précédents pour effectuer toute la partie supérieure du pipeline de la FIGURE 1. Il reste cependant une autre fonction à implémenter, et qui fait la particularité du codage JPEG. Il s'agit de la séparation des coefficients AC et DC dans la sous-fonction `CoefficientsACDC`. Cette fonction doit séparer le coefficient basse fréquence situé en position `[1,1]` dans chaque bloc, et récupérer les coefficients restants, dits AC, suivant le parcours en zigzag suivant :

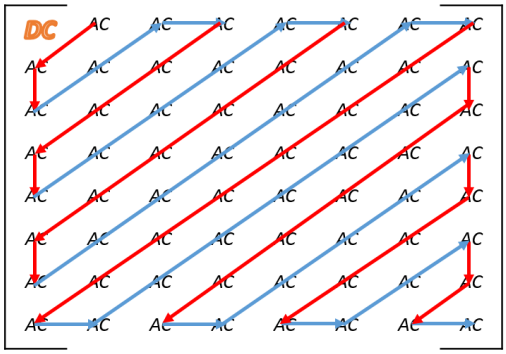

FIGURE 2 – Parcours en zigzag pour récupérer les coefficients AC d'un bloc de taille 8×8.

La récupération des coefficients AC d'un bloc est effectuée avec la sous-fonction `ParcoursBlocZigzag` qui vous est fournie. Il faut ensuite éliminer les 0 à la fin du vecteur que l'on ne souhaite pas coder et rajouter un symbole 'EOB' (pour *End Of Block*) à la fin du vecteur. Ici ce symbole sera remplacé par la valeur numérique 1000. La fonction `CompressionJPEG` pourra ainsi estimer le taux de compression associé au codage (en sortie), que vous pourrez tester avec le script `Exercice_4`.

Exercice 5 - Décompression JPEG et analyse de la qualité de reconstruction

Pour évaluer la qualité de la compression, il est nécessaire d'effectuer une reconstruction de l'image à partir des coefficients quantifiés de la DCT. Complétez la fonction `DecompressionJPEG` qui effectue la quantification inverse des coefficients, suivie de la DCT inverse (ou IDCT), définie comme suit :

$$\mathrm{IDCT}_{2D}(x,y) = \sum_{j=0}^{7} \left(\frac{C(j)}{2} \sum_{i=0}^{7} \frac{C(i)}{2} \mathrm{bloc}(i,j) \cos\left[\frac{(2x+1)i\pi}{16}\right] \right) \cos\left[\frac{(2y+1)j\pi}{16}\right] \tag{7}$$

où les coefficients $C(x)$ et $C(y)$ valent $\sqrt{2}$ en 0 et 1 sinon. La quantification inverse consiste quant à elle à multiplier les blocs par la table de quantification utilisée lors de l'étape de quantification. La fonction `idct2` est présente dans Matlab pour effectuer la DCT inverse. Lancez ensuite le script `Exercice_5` qui va effectuer le codage, puis le décodage et afficher le résultat suivant les différents facteurs de qualité F_Q possibles. Diminuez la valeur de la variable `F_Qualite_Max` pour que l'exécution s'arrête à un facteur de qualité plus faible et ainsi pouvoir observer les différents artefacts occasionnés par la compression, et la valeur du PSNR associé.

Bonus : Complétez la sous-fonction `IDCT2Rapide` qui permet d'effectuer la DCT inverse sur un bloc de taille 8×8 en utilisant les redondances des coefficients vues en cours. Ajoutez-là ensuite dans la fonction `DCT2DParBlocs` en possibilité supplémentaire par rapport à `idct2`.