COMPRESSION DES IMAGES

TP2 : Compression JPEG

Nous allons implémenter sous Matlab les différentes étapes de la compression JPEG vues en cours. Notez qu'à aucun moment nous ne stockerons sur le disque le fichier `.jpg` résultant, mais que nous pourrons par contre visualiser les effets de la compression sur l'image. Nous travaillerons dans un premier temps avec l'image `lena.tif`.

1) Charger l'image sous Matlab. La première étape du JPEG consiste à travailler dans le plan des luminance et chrominances. Effectuer cette conversion à l'aide de la fonction fournie `rgb2ycbcr()`. Vérifier rapidement que notre œil n'est pas sensible à un sous échantillonnage d'un facteur 2 de la chrominance. Pour la suite, utiliser la fonction `double()` sur l'image en luminance/chrominance de façon à travailler avec un niveau suffisant de précision.

2) Découper ensuite l'image en blocs disjoints de dimension 8x8 pour la luminance, et en blocs 4x4 pour les chrominances, et en calculer les transformées en cosinus. Vérifier que la propriété de concentration de l'énergie de cette transformation est vérifiée.

3) L'étape suivante consiste à quantifier chacun des coefficients de ces transformées. Construire deux matrices de quantification (une pour la luminance, une seconde pour les deux chrominances), telles que les éléments Q(i,j) situés sur la ième ligne et jème colonne de ces matrices vérifient :
   Q(i,j) = 1+F(i+j-1), où F désigne le facteur de qualité (F>0).

4) Effectuer la quantification de chacun des blocs.
   Attention : *La partie entière `floor(1.3)=1, floor(0.2)=0 et floor(-1.4)` vaut –2 et non pas –1 ; il faut dont faire `sign(x)*floor(abs(x))` pour avoir le bon résultat*

5) Effectuer ensuite la déquantification. Comparer les valeurs déquantifiées obtenues avec les « vraies » valeurs obtenues après calcul des transformées.

6) Enfin, afficher l'image compressée. Pour cela, il faudra interpoler les valeurs manquantes des deux chrominances (pour revenir, depuis des blocs 4x4, à des blocs 8x8) avec la commande : `bloc8x8 = kron(bloc4x4,ones(2,2))`. Utiliser ensuite de manière successive les fonctions `uint8()` et `ycbcr2rgb()` pour revenir à une image RGB codée sur 8 digits.

7) Visualiser les effets de la compression. Tracer l'erreur moyenne en fonction du facteur F. A partir de quelle valeur de F percevez vous une différences entre l'image d'origine et sa version compressée ?

8) En pratique, la majorité des valeurs quantifiées des transformées de chacun des blocs est compressée selon une méthode combinant RLE et Huffman après

lecture en zig-zag. Il est donc nécessaire de connaître ici les fréquences d'apparition des différentes valeurs quantifiées. La fonction `freq=freq_bloc(blocY, blocCb, blocCr, freq)` actualise ces fréquences d'apparition pour l'ensemble de l'image. Insérer cette fonction dans votre code. En déduire les probabilités d'apparition `proba` de chacune des valeurs comprises entre 0 et 255.

9) Utiliser la fonction `L=compress(proba)` qui calcule pour vous les mots de code associés aux valeurs quantifiées des transformées ainsi que la longueur moyenne `L` des mots de code. En déduire le taux de compression.

D'autres images sont à votre disposition. Tester votre code et les effets de la compression sur chacune des ces images. Conclure quant à l'efficacité de la compression JPEG selon le type d'image.

lena.tif

paysage.tif

symboles.tif

logo.tif