TP 4 - Compression d'images

Pour le 29/05/14

Les fichiers `http://www.di.ens.fr/~waldspurger/tds/13_14_s2/encode.m` et `http://www.di.ens.fr/~waldspurger/tds/13_14_s2/decode.m` définissent une fonction d'encodage et une fonction de décodage.
La fonction `encode` prend en entrée une suite d'entiers (dont certains peuvent éventuellement être négatifs) et renvoie un code associé à cette suite, sous la forme d'une chaîne de caractères ne contenant que des 0 et des 1.
La fonction `decode` prend en entrée un code calculé par la fonction `encode` et renvoie la suite d'entiers dont provient le code. Sur de longues entrées, elle peut avoir un temps d'exécution assez long (trente secondes).
L'algorithme d'encodage utilisé est l'algorithme de compression de Huffman.

[Pour les personnes qui préfèrent Python : je suis désolée mais je n'ai pas trouvé d'équivalent Python de ces deux fonctions. Il y a des codes disponibles qui réalisent un codage de Huffman mais ceux que j'ai vus n'acceptent en entrée que des chaînes de caractères ; pour pouvoir les utiliser, il faudrait les modifier pour qu'ils acceptent en entrée des listes d'entiers arbitrairement grands.]

1. Dans cette question et la suivante, on définit une fonction `compress`, qui prend en entrée un paramètre $\delta > 0$ et une image `im`, de taille $2^J \times 2^J$ pour un certain $J \in \mathbb{N}$.
a) Calculer la transformée en ondelettes de Haar de `im`.
[Vous pouvez reprendre le code que vous avez écrit pour le TP précédent ou utiliser celui qui figure dans le corrigé de ce même TP.]
b) Concaténer les 2^{2J} coefficients de la transformée en un vecteur `v` à une seule dimension.
c) Calculer `v_quant`, le vecteur contenant les coefficients de `v` arrondis au multiple de δ le plus proche.
d) Calculer `im_comp`, le code associé à `v_quant`$/\delta$ par la fonction `encode`.

2. a) Calculer un tableau qui contient le nombre d'occurences de chaque entier dans `v_quant`$/\delta$.
[Indication : vous pouvez utiliser la commande `n_occ = hist(v_quant/delta,centers)`, pour un choix approprié de `centers`.]
b) En déduire l'entropie `H` associée à la suite `v_quant`$/\delta$.
c) Faire renvoyer `im_comp` et `H` à la fonction `compress`.

3. Écrire une fonction `decompress` qui prend en entrée un $\delta > 0$ et une chaîne de caractères `im_comp` (calculée par la fonction `compress`) et renvoie une approximation `im_rec` de l'image dont provient `im_comp`.

4. a) Charger l'image `http://www.di.ens.fr/~waldspurger/tds/13_14_s2/batiment_2.png` dans une variable `im`. La convertir en un tableau de flottants à deux dimensions.
b) Afficher sur trois figures différentes l'image `im` et les deux `im_rec` obtenues pour $\delta = 100$ et $\delta = 30$.
[Utiliser pour l'affichage des images les mêmes commandes que dans le TP précédent, ainsi que la commande `axis equal`, qui évite que les images ne soient déformées.]

5. Effectuer le calcul suivant pour $\delta = 2, 5, 10, 20, 50, 100, 200, 500$.
a) Calculer la chaîne `im_comp` et l'entropie `H` renvoyées par `compress`.
b) Calculer le nombre moyen de bits par pixel dans `im_comp`, c'est-à-dire :

$$R = \frac{\text{nombre de caractères de } \texttt{im\_comp}}{\text{nombre de pixels de } \texttt{im}}$$

c) À partir de `im_comp`, calculer `im_rec` et en déduire l'erreur quadratique sur `im`, c'est-à-dire :

$$D = \sum_{k_1,k_2} |\texttt{im\_rec}[k_1, k_2] - \texttt{im}[k_1, k_2]|^2$$

d) Afficher sur une quatrième figure les graphes de $\log_2(D)$ en fonction de R et de $\log_2(D)$ en fonction de H.

6. a) D'où vient la différence entre les deux courbes que vous venez de tracer ?
b) Vous devez observer que la variation de $\log_2(D)$ en fonction de H est à peu près affine pour les plus petites valeurs de D. À quel résultat du cours est-ce dû ?
c) Pourquoi ce résultat ne s'applique-t-il plus pour de grandes valeurs de D ?

7. a) À partir de quelle valeur de R la différence entre `im` et `im_rec` devient-elle difficilement visible ?
b) Si on avait utilisé pour coder l'image la méthode la plus naïve possible, en concaténant les représentations binaires des valeurs de chaque pixel, quel nombre de bits par pixel aurait-on obtenu ?
[La valeur de chaque pixel est un entier compris entre 0 et 255.]

Pour me faciliter la correction, pensez à :
- inclure dans votre code toutes les instructions d'affichage demandées
- me rendre une copie contenant les réponses aux questions posées, soit par mail dans un format lisible (pdf ou scans de bonne qualité d'un texte manuscrit) soit sous la forme d'une feuille déposée dans mon casier (dans l'entrée de l'espace Cartan)

Merci !