Traitement Numérique des Images
TELECOM Nancy 2ème Année Apprentissage

TD « Compression d'images »

…

Objectif

Il s'agit ici de mesurer certaines caractéristiques des images numériques (histogramme, entropie), afin de retrouver les principales propriétés des codes évoqués en cours et d'évaluer l'efficacité de la compression sur ces images. Vous utiliserez pour cela le logiciel Matlab. Dans le répertoire compression, vous n'utiliserez pour ce TD que les images « lena512.bmp », « lena512.ascii » et « souris.bmp ».

Exercice 1 – Fonction entropie(I)

1- Programmer une fonction entropie(I), I étant la matrice d'une image lue avant l'appel à la fonction. NB : ne pas oublier de transformer I au format 'double' dès le début de la fonction.

2- A l'aide de la fonction précédente, calculer l'entropie de l'image *lena512.bmp*.

3- Ajouter une constante à l'image lena512.bmp (attention à ne pas sortir de la dynamique de l'image de départ) et calculer l'entropie. Commenter.

4- Calculer l'entropie de l'image *souris.bmp*.

5- Commenter les différences entre les entropies des images *lena512.bmp* et *souris.bmp* et les performances de compression auxquelles on peut s'attendre. Confirmer ces différences en affichant les histogrammes des deux images.

Exercice 2 – Fonction psnr(I,J)

1- Programmer une fonction ***psnr(I,J)*** qui calcule le PSNR entre I et J, I et J étant les matrices de deux images lues avant l'appel à la fonction. I représente l'image d'origine et J l'image « traitée » (compressée) dont on veut mesurer le rapport signal à bruit par rapport à l'origine. NB : ne pas oublier de transformer I et J au format 'double' dès le début de la fonction.

2- A l'aide de la fonction précédente, calculer le PSNR entre

- l'image *souris.bmp* et elle-même ;
- l'image *souris.bmp* et *souris.bmp* bruitée avec la fonction Matlab ***imnoise***. Que peut-on conclure quand le bruit augmente ?

Exercice 3 – Quantificateur scalaire uniforme (QSU)

1- Ecrire une fonction ***QSU(I,nb_niveaux)*** qui quantifie une image quelconque de manière scalaire uniforme, et retourne l'image quantifiée, le débit binaire maximum du quantificateur, ainsi que le débit entropique obtenu après quantification.

Le QSU est décrit par la formule suivante :

$$Q(x) = \left(\text{int}\left[\frac{x-d}{q} \right] + \frac{1}{2} \right) * q + d \qquad (1)$$

où d représente la valeur minimale des niveaux de l'image d'origine, q le pas de quantification, et int une fonction qui rend la partie entière d'un nombre ("floor" sous Matlab).

Dans le cas général des images sur 256 niveaux de gris dans le domaine d'origine, on prend :

d = 0, et $q = \frac{255}{\text{nb\_niveaux}}$.

Attention : la formule (1) ne fonctionne pas pour le plus grand niveau de gris (255). Elle conduit en effet ce niveau à être quantifié dans un intervalle suivant qui n'existe pas. Il faudra donc forcer cette valeur à être quantifiée dans le dernier intervalle.

2- Tester votre programme sur l'image souris.bmp, avec différents taux de compression. A chaque fois, afficher et commenter l'histogramme de l'image quantifiée, relever le PSNR et les débits binaires. Tracer les courbes PSNR(débit maximum) et PSNR(débit entropique) sur la même figure.

3- A l'aide de la fonction ***imwrite***, compresser l'image souris.bmp au format JPEG (.jpg) à différents taux de compression. Comparer visuellement à taux de compression identiques (ou très proches) l'image souris compressée JPEG et l'image souris quantifiée avec la fonction QSU. Comment expliquer la supériorité visuelle de l'image codée JPEG ?

Exercice 4 – Codage sans pertes : algorithme de Huffman

On utilisera ici la routine ***Huffman.m*** fournie dans le répertoire.

1- Appliquer cette fonction au fichier lena512.ascii. Examiner le fichier résultat (lena512.out) en utilisant un éditeur de texte (Word par exemple). Recopier la partie utile de la table de codage (concernant les symboles présents dans l'image lena512.ascii), de même que les caractéristiques fournies (longueur du fichier, entropie, …)

2- Vérifier la propriété de préfixe du code de Huffman. Commenter la longueur des mots de code.

3- Créer à l'aide du bloc notes un fichier ascii « essai1ascii.txt » contenant les 26 lettres de l'alphabet les unes à la suite des autres.

Sauvegarder ce fichier et effectuer les questions 1.1 et 1.2. Commenter les résultats obtenus.

4- Reprendre la question 1.3 avec un fichier essai2ascii.txt contenant 26 lettres mais tel que la lettre 'a' ait une probabilité égale à ½.