Format de compression JPEG.

Introduction

Le but de ce TP est d'implémenter une partie d'un codeur Jpeg c'est à dire un programme qui compresse une image au format Jpeg. Nous ne ferons pas en pratique la partie liée au codage car elle fait intervnenir des notions que vous n'avez pas encore vu mais nous essairons d'estimer la taille du fichier créé par une telle procédure.
La compression Jpeg repose sur une transformée en DCT locale sur des carrés 8 par 8. Au lieu de coder les niveaux de gris d'une image on code ses coefficients dans la base de DCT locale. Plus précisément on va modifier les coefficients de manière à utiliser le moins d'octets possible. Pour cela on utilisera un codeur entropique.
Dans toute la suite on supposera que les deux dimensions $N1$ et $N2$ des images considérées sont des multiples de 8 et que les images sont en niveau de gris.

1. Pour une image $N1 \times N2$ en niveau de gris dont les niveaux de gris sont codés par un entier de 0 à 255 quel est le nombre de bits nécessaire au codage si on utilise une méthode élémentaire ?

1 DCT locale et DCT locale inverse

2. Ecrire une fonction *DCTLocale*

   `function D=DCTLocale(I)`

   qui prend en entrée une matrice I et qui renvoie une matrice D de même taille qui contient les DCT des imagettes de I obtenues en découpant l'image en blocs 8 par 8. On retirera au début de la fonction la valeur 128 à tous les coefficients de I.
3. Ecrire la fonction InvDCTLocale

   `function Irec=InvDCTLocale(D)`

   qui reconstruit la matrice à partir de la matrice contenant la DCT locale.
4. Faites un test en utilisant Lenna (Charger Wavelab)

   `>>I=readImage('Lenna');`
5. Afficher la différence entre l'image originale I et l'image reconstruite $Irec$. Pourqui cette différence n'est pas nulle ?

2 Rudiments de théorie de l'information

6. La théorie du codage nous assure que le nombre de bits moyen minimal pour coder une suite d'éléments $(a_n)_{1\leqslant n\leqslant N}$ d'un alphabet, dont chacun des éléments est indépendant et tirés au hasard avec une probabilité $P(a)$ est bornée inférieurement par

$$E = \sum_{n=1}^{N} -P(a_n)\log_2(P(a_n))$$

   On appelle Entropie cette valeur. On peut montrer qu'on peut construire des codes dont le nombre moyen de bits est aussi proche de cette valeur que l'on souhaite.
   Si on suppose qu'on doit coder une suite aléatoire de quatre symboles $\{a,b,c,d\}$ où pour chaque élément de la suite on a une probabilité $P(a) = P(b) = P(c) = P(d) = \frac{1}{4}$ de choisir l'un des quatre éléments, quelle est la valeur de l'entropie ? Dans ce cas proposer un code en bits associés à chacun des éléments.
7. Si $P(a) = \frac{1}{2}$, $P(b) = \frac{1}{4}$ et $P(c) = P(d) = \frac{1}{8}$ quelle est la valeur de l'entropie ?
8. Calculer la longueur moyen du code binaire d'un symbole en utilisant le code suivant : a est codé par 0, b par 10, c par 110 et d par 111. Un tel code est appelé préfixe et est donc décodable.
9. Ecrire un programme Matlab

   `E=Entropie(S)`

   qui calcule l'entropie d'une suite de valeurs. On pourra utiliser la commande *unique* pour déterminer l'ensemble des valeurs prises par S. La probabilité étant remplacée par la fréquence d'apparition des différentes valeurs.
10. Tester le code précédent sur le vecteur $[2,3,8,2,7,2,2,3,2]$ et vérifier que la valeur obtenue est bien la bonne.
    Pour effectuer la compression d'une image on va modifier la suite de coefficients en essayant de faire diminuer son entropie tant que possible. Le but de la quantification est d'approcher le tableau des coefficients de DCT par un vecteur associé proche dont l'entropie sera beaucoup plus faible que celui du tableau de départ.

3 Quantification

11. La transformation opérée par la DCT locale étant théoriquement inversible on ne va pas gagner quand chose en terme de compresssion. L'essentiel de la compression est réalisé en quantifiant les coefficients de la DCT, c'est à dire en les arrondissant.
    A l'aide de la commande *round* écrire une fonction *QuantificationUniforme*

    `function IQ=QuantificationUniforme(I,q)`

    qui reconstruit l'image I après avoir arrondi (on dit aussi quantifié) tous les coeffcients de la DCT locale au multiple de q le plus proche.
12. On peut évaluer la qualité d'une reconstruction d'image en calculant son PSNR. La formule du PSNR est donnée par
$$PSNR = 10\log_{10}\left(\frac{N1\times N2\times 255^2}{\|I-Irec\|^2}\right)$$
    Plus le PSNR est grand plus l'image reconstruite est de bonne qualité.
    Déterminer le PSNR de la reconstruction utilisant une quantification uniforme avec un pas de 10.
13. En pratique on utilise un pas de quantification différent pour chaque coefficient car l'oeil n'est pas sensible de la même manière aux modifications des différents coefficients. La table de quantification stantard de Jpeg est donnée par la matrice suivante : http ://www.math.u-bordeaux.fr/ dossal/Enseignements/L3-TheoSignal/QuantificationJpeg.m Elle correspond à un facteur de qualité Q=50.
    Ecrire un programme

    `Irec=Quantification(I,T)`

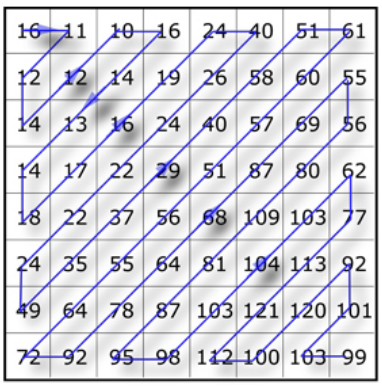
    qui prend en entrée une image I (toujours de taille $N1\times N2$) et qui renvoie une image $Irec$ obtenue par quantification des coefficients de DCT en utilisant la table T.
14. Calculer le PSNR associé à cette reconstruction.
15. Faire un test avec une autre image.
16. On peut construire différentes tables de quantification en Jpeg associés à différents facteurs de qualité Q. Pour construire une table associée à un niveau de qualité $Q\in[50,100]$, on peut utiliser une table de quantitication obtenue par la formule
$$T_Q = \text{round}(\frac{T(100-Q)}{50})$$
    Donner la table associée à une qualité $Q=80$.
17. Compresser Lenna avec cette table et donner le PSNR associé.
18. Pour $Q\in[0,50]$ la formule est différente :
$$T_Q = \text{round}\left(\frac{50T}{Q}\right).$$
    Effectuer une compression de Lenna avec un facteur de qualité $Q=20$ et donner le PSNR associé.
19. Dans la suite nous n'aurons pas besoin de regarder précisément la valeurs des coefficients de DCT mais uniquement leurs quotients obtenus pas la division par la *Table de quantification*. Ecrire une fonction Matlab Quantification2

    `DQ=Quantification2(I,T)`

    Qui ne renvoie que les valeurs des quotients de la DCT locale par la table de quantification.

4 Calcul du volume théorique des données.

Si on visualise quelques blocs 8×8 d'une DCT locale quantifiée, on voit que nombre de coefficients sont nuls et que la plupart d'entre eux sont petits. On va essayer de mesurer la place que peuvent prendre ces bloque de DCT.
Dans un premier temps notons qu'on ne code pas un tableau 8×8 mais un vecteur obtenu par réarrangements des coefficients le long d'un serpentin.



Ce réarrangement est calculé par le code suivant :

```
function V=ReshapeJpeg2(D)
V=zeros(1,64);
for k=1:32
    n=ceil(sqrt(2*k+1/4)-0.5);
    temp=k-n*(n-1)/2;
    if mod(n,2)<1
    i=temp;
    else
        i=n+1-temp;
    end
    j=n+1-i;
    [k,i,j]
    V(k)=D(i,j);
    V(65-k)=D(9-j,9-i);
end
```

En effet une fois les coefficients réordonnés, la grande majorité des 0 se trouve à la fin du vecteur. Ainsi pour coder cette séquence on code la suite des premiers coefficients et à la place du premier 0 derrière lequel il n'y a que des 0 on code un message de fin de block (EOB).
Ainsi le tableau

$$\begin{array}{ccccc} 7 & 0 & 0 & -2 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ -4 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{array} \tag{1}$$

sera codé par la suite $7,0,1,3,-1,0,-2,0,0,-4,3,EOB$. La question qui se pose maintenant est de coder efficacement ces suites de valeurs. On va coder séparément la première valeur des autres.

5 Codage des premiers coefficients

20. Calculer le nombre de valeurs possiblement prise par ce premier coefficient en fonction du pas de quantification. Donner le nombre de bits nécessaires pour le coder de manière élémentaire.
21. Ecrire un programme Matlab

    `function V=PremiersCoefs(I)`

    qui à partir d'une image renvoie le vecteur des premiers coefficients de DCT.
22. Ecrire un programme *Entropie Premiers coefs* qui calcule pour une image, l'entropie de la suite formée des premiers coefficients.
23. Quelles sonts les valeurs obtenues pour Lenna avec des compressions de facteurs de qualité $Q \in \{20,40,50,60,80\}$.

5.1 Codage des autres coefficients

24. Ecrire un programme qui à un tableau 8×8 renvoie le vecteur décrit en début de section se terminant par EOB en otant le premier coefficient. On pourra remplacer l'élément EOB (end of block) par le réel 1000. Cela n'aura pas d'influence sur le calcul de l'entropie.

    `function V=VecteurLocal(D)`
25. Ecrire un programme qui à une image associe le vecteur formé de la concaténation de tous les vecteurs obtenus sur l'ensemble des blocs.

    `function V=VecteurGlobal(I)`
26. Ecrire un programme qui renvoie le nombre de bits moyen minimal (donc basé sur les valeurs des entropies calculées) d'une image comprimée avec un facteur de qualité Q si on suppose connue la table de quantification utilisée, c'est à dire la table de référence T connue, ainsi que le PSNR.

    `function [Nbbits, P]= Compression(I,Q)`
27. Ecrire un programme qui affiche le PSNR en fonction du nombre total de bits utilisés. On pourra prendre $Q \in \{20,30,40,50,60,70,80\}$.