Ecole Centrale de Lyon **Multimédia : Concepts et technologies**
S7 - INF a 1-FH Mohsen ARDABILIAN, Emmanuel DELLANDREA

BE - La transformation en cosinus discrète (DCT, inverse DCT) et son application à la compression d'image

L'objectif de cette séance est d'appliquer la transformation DCT pour compresser des images. Puis, de procéder à la décompression, en appliquant la transformation inverse. Et enfin, de calculer l'erreur entre l'image de départ et les images après la décompression pour des facteurs de quantification allant de 1 à 25. Pour chaque facteur de quantification, on calculera la compression effectuée en relevant le nombre de coefficients DCT différents de zéro.

Les étapes de la compression sont les suivantes :

1. Lecture d'une image bitmap
2. Récupération de la matrice des coefficients RGB
3. Calcul des coefficients DCT de la matrice par blocs de 8x8 pixels
4. Quantification
5. Ecriture des coefficients DCT dans un fichier

Pour la décompression, il suffit d'inverser les étapes précédentes.

Un ensemble incomplet de fonctions matlab implémente les étapes de la compression. Cet ensemble, qui vous est fourni, comprend les fonctions suivantes :

```
- function [duree_traitement] = main_compression(qualite, nom_source_bmp, nom_destination_jpg)
    o function [imageRGBdouble] = lecture_image_bmp(nom_fichier)
    o function [M] = conversion_spatial_frequantiel(matrice, qualite)
        ▪ function [res] = ajoute(m,valeur)
        ▪ function [Q] = mat_quant(Fq)
    o function [] = ecriture_jpg(JPG, nom, qualite)
```

La fonction DCT :

```
function [a] = m_DCT2(m)
```

est absente.

Un ensemble complet de fonctions matlab implémente les étapes de la décompression. Cet ensemble, qui vous est aussi fourni, comprend les fonctions suivantes :

```
- function [duree_traitement] = main_decompression(nom_destination_jpg , nom_source_bmp)
    o function [JPG,qualite] = lecture_jpg(nom)
    o function [M] = conversion_frequentiel_spatial(matrice, qualite)
        ▪ function [res] = ajoute(m,valeur)
        ▪ function [Q] = mat_quant(Fq)
        ▪ function [a] = m_IDCT2(m)
    o function [] = ecriture_bmp(BMP, nom)
```

La transformation inverse est obtenue par la formule :

$$pixel(x,y) = \frac{1}{\sqrt{2N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} c(i)c(j)DCT(i,j)\cos\left(\frac{(2x+1)i\pi}{2N}\right)\cos\left(\frac{(2y+1)j\pi}{2N}\right),$$

$$c(k) = \frac{1}{\sqrt{2}} \text{ si } k = 0, \text{ et } 1 \text{ si } k > 0$$

Le code matlab de cette transformation inverse avant l'optimisation est le suivant :

```
function [a] = m_IDCT2(m)
```

```
%Fonction calculant la DCT inverse d'une matrice
%arguments d'entree:
% - m matrice 8*8 sur laquelle sera appliquée la dct inverse
%argument de sortie:
% - a matrice 8*8 correspondant à la dct inverse appliquée sur m
%recuperation de la taille de la matrice m
[N,N] = size(m);
%declaration d'une matrice nulle de meme dimension que m
a = zeros(N,N);

%Calcul des coefficients de la tranformes en dct
%i,j sont les indices de la nouvelle matrice a
%u,v sont les indices de la matrice m
for i=0:N-1
       for j=0:N-1
              for u=0:N-1
                     for v=0:N-1
                            %Test permettant de determiner la valeur des variables C1 et C2
                            if u==0
                                   c1=1/sqrt(2);
                            else
                                   c1=1;
                            end
                            if v==0
                                   c2=1/sqrt(2);
                            else
                                   c2=1;
                            end
  a(i+1,j+1)=a(i+1,j+1)+1/sqrt(2*N)*c1*c2*m(u+1,v+1)*cos(pi*u/N*(i+1/2))*cos(pi*v/N*(j+1/2));
                     end
              end
       end
end
```

I. Implémentez la fonction :

```
function [a] = m_DCT2(m)
```

en utilisant la formule de la DCT :

$$DCT(i,j)=\frac{1}{\sqrt{2N}}c(i)c(j)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}pixel(x,y)\cos\left(\frac{(2x+1)i\pi}{2N}\right)\cos\left(\frac{(2y+1)j\pi}{2N}\right),$$

$$c(k)=\frac{1}{\sqrt{2}}\ si\ k=0,\ et\ 1\ si\ k>0.$$

Une fois cette fonction implémentée, testez la avec la matrice :

```
Im = [1 2 3 4 5 6 7 8 ;
      8 7 6 5 4 3 2 1 ;
      1 2 3 4 5 6 7 8 ;
      8 7 6 5 4 3 2 1 ;
      1 2 3 4 5 6 7 8 ;
      8 7 6 5 4 3 2 1 ;
      1 2 3 4 5 6 7 8 ;
      8 7 6 5 4 3 2 1 ] ;
```

Puis, appliquez la DCT inverse pour vérifier que votre code est correct. Si votre code est correct l'application de la DCT et DCT inverse vous rend votre matrice Im.

II. Calculez la différence entre une image bitmap avant la compression et après la décompression pour des facteurs de quantification allant de 1 à 25.

III. Pour chaque facteur de quantification, calculez le nombre de coefficients DCT différents de zéros.

IV. Tracez les courbes d'erreurs (moyenne et la norme), ainsi que les taux de compression, en fonction du facteur de quantification.