TP3 - Image-comptes-rendus-madali-nabil

May 3, 2019

1 TP3 après le son l'image !

Un image est un rectangle de pixel, ou matrice avec le premier pixel en haut à gauche.

2 Partie I : Fourier

2.1 1) Pour débuter en noir et blanc

Chaque pixel est une valeur entière ou plus exactement un entier 8, 16, 24 ou 32 bits signé. Usuellement on utilise le format 8 bits qui permet un niveau de gris allant de 0 à 255 (0 noir - 255 blanc).

2.1.1 Quelques outils

- imfinfo - Return information about image file (MATLAB Toolbox).
- imread - Read image file (MATLAB Toolbox).
- imwrite - Write image file (MATLAB Toolbox).

Matlab est capable de lire et de décoder les fichiers images JPEG, TIFF, BMP, PNG, HDF, PCX ou XWD. Une image sous Matlab peut être représentée sous plusieurs formes, mais toujours sous forme d'une matrice. Avant de traiter une image dans Matlab, il faut la lire et décoder son format afin de la transformer en une matrice de valeurs.

```
In [87]: %plot -s 1600,800
```

```
In [88]: imfinfo('histeq.png')
```

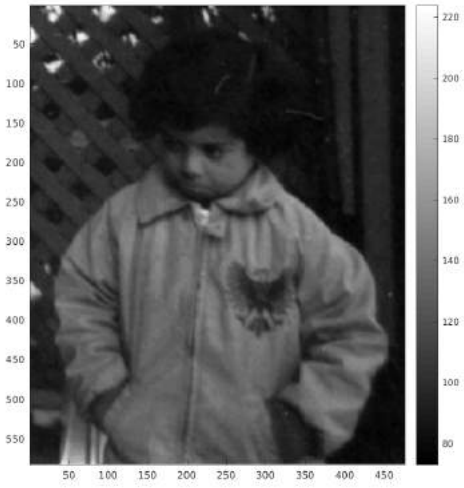
```
ans =

  struct with fields:

                Filename: '/home/nabil.madali/histeq.png'
            FileModDate: '11-Apr-2019 17:07:03'
                FileSize: 82579
                  Format: 'png'
           FormatVersion: []
                   Width: 476
```

```
                    Height: 582
                 BitDepth: 8
                ColorType: 'grayscale'
          FormatSignature: [137 80 78 71 13 10 26 10]
                 Colormap: []
                Histogram: []
           InterlaceType: 'none'
            Transparency: 'none'
  SimpleTransparencyData: []
         BackgroundColor: 1
         RenderingIntent: []
          Chromaticities: []
                   Gamma: 0.4545
              XResolution: 5669
              YResolution: 5669
          ResolutionUnit: 'meter'
                 XOffset: []
                 YOffset: []
              OffsetUnit: []
         SignificantBits: []
            ImageModTime: '9 Jan 2017 17:37:11 +0000'
                   Title: []
                  Author: []
             Description: []
               Copyright: []
            CreationTime: []
                Software: []
              Disclaimer: []
                 Warning: []
                  Source: []
                 Comment: []
               OtherText: {2Œ2 cell}
```

In [89]:
```
I = imread('histeq.png');
whos I
```

```
  Name        Size              Bytes  Class    Attributes

  I         582x476            277032  uint8
```

In [90]:
```
imagesc(I)
colorbar
colormap(gray)
axis('image');
axis on
```



2.2 2) La couleur

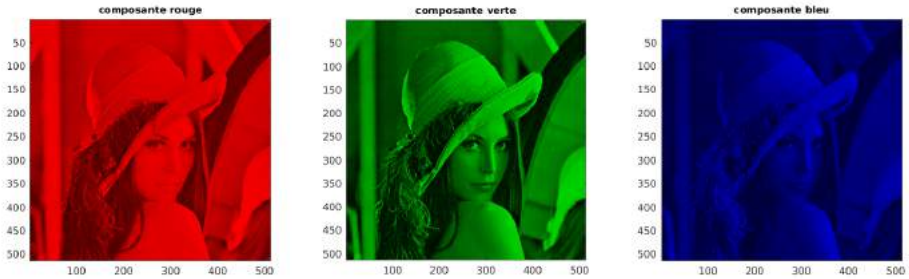
La couleur est une donnée importante pour une image, elle modifie la perception que l'on a de l'image. L'espace de représentation standard décompose une image en trois plans de couleur: le rouge, le vert et le bleu - Red/Green/Blue RGB en anglais. Les couleurs finales sont obtenues par synthèse additive de ces trois couleurs primaires. Il existe cependant des problèmes qui peuvent nécessiter de changer d'espace de couleur pour percevoir différemment l'image. Il y a des images où la couleur importe peu, par exemple des photographies de cellules vivantes (pseudo-transparentes), des images radar, des images satellites... Dans ce cas, l'espace RGB n'est plus utilisé. On lui préfère d'autres espaces comme HSV Hue/Saturation/Value ou YCbCr Luminance/Chrominance bleue/Chrominance rouge.

L'image est décrite par 3 matrices d'entiers 8 bits ou plus précisément sur matlab une hypermatrise ou tableau à 3 entrées la troisème entrée allant de 1 à 3 pour la couleur rouge, vert et bleu.

```
In [91]: Lena=imread('http://vision.gel.ulaval.ca/~jflalonde/cours/4105/h14/tps/results/tp2/tomt

         whos Lena

  Name        Size                Bytes  Class    Attributes

  Lena      512x512x3            786432  uint8
```

```
In [92]: imagesc(Lena)
         axis('image');
```

In [93]:
```
n = size(Lena,1);
figure
f1 = cat(3, Lena(:,:,1), zeros(n), zeros(n)) ;
f2 = cat(3, zeros(n), Lena(:,:,2),  zeros(n)) ;
f3 = cat(3, zeros(n), zeros(n), Lena(:,:,3) ) ;
subplot(1,3,1); imagesc ( f1 );axis('image'); ; title('composante rouge');
subplot(1,3,2); imagesc ( f2 );axis('image'); ; title('composante verte');
subplot(1,3,3); imagesc ( f3 );axis('image'); ; title('composante bleu');
```



2.2.1 Représentation YCbCr (ou YUV)

La transformation du systeme de représentation précédent RGB vers YCbCr Luminance/Chrominance bleue/Chrominance rouge est donné par

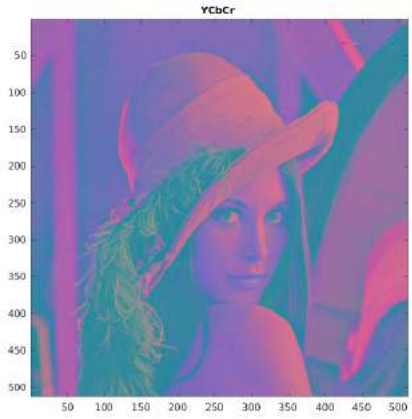
$$\begin{cases} Y & = 0.299R + 0.587G + 0.114B \\ Cb & = -0.1687R - 0.3313G + 0.5B + 128 \\ Cr & = 0.5R - 0.4187G - 0.0813B + 128 \end{cases}$$

L'ajout de 128 à Cb et Cr permet d'obtenir des octets dont les valeurs varient entre 0 et 255.
Représenter Léna sous cette décomposition

```
In [94]: T = [0.299 ,0.587 ,0.114; -0.1687 , - 0.3313 ,  0.5; 0.5 ,- 0.4187 , - 0.0813]
         offset = [0,128,128]
```

```
T =

    0.2990    0.5870    0.1140
   -0.1687   -0.3313    0.5000
    0.5000   -0.4187   -0.0813


offset =

     0   128   128
```

```
In [95]: ycbcr = zeros(size(Lena),class(Lena));

In [96]: for p = 1:3
          ycbcr(:,:,p) = imlincomb(T(p,1),Lena(:,:,1),T(p,2),Lena(:,:,2), T(p,3),Lena(:,:,3),off
         end

In [97]: imagesc (ycbcr)
         subplot(1,2,1); imagesc ( Lena );axis('image'); ; title('RGB');
         subplot(1,2,2); imagesc ( ycbcr );axis('image'); ; title('YCbCr ');
```

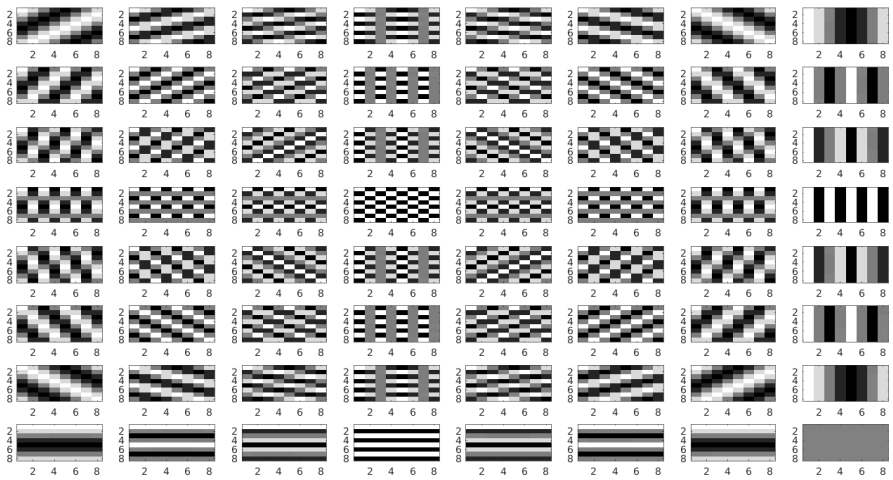


2.3 3) Transformée de Fourier 2d et transformée en Cosinus Discret

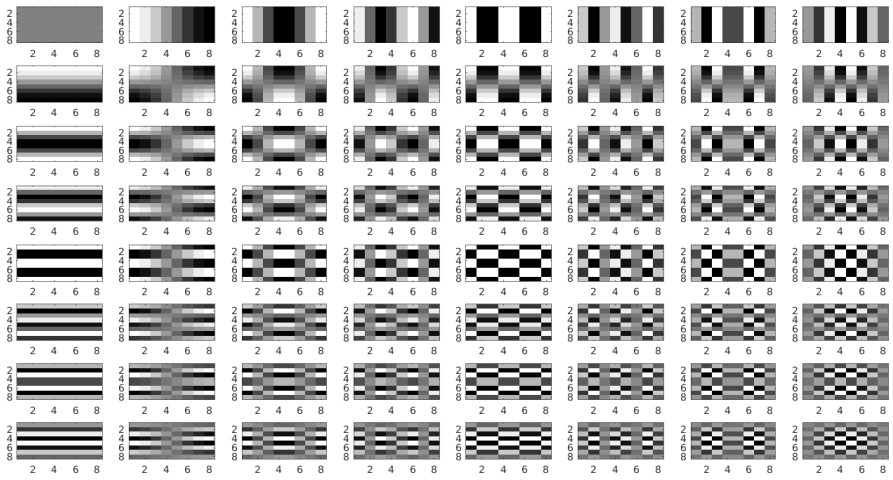
A l'aide de la fonction idct2 (inverse fast fourier transform 2d) sur une image 8x8 (niveaux de gris) représenter les 64 images correspondants aux 64 transformeées possibles valant 1 en un point de la grille et 0 sinon.

Réaliser l'opération analogue avec ifft2.

```
In [98]: N=8 ;
         cpt=1;
         for m = 1:8
         for n = 1:8
         [x,y]=meshgrid(0:(N-1),0:(N-1));
         subplot(8,8,cpt); imagesc(real(exp(-1j*2*pi*(m*x/N + n*y/N))));colormap(gray)
         cpt=cpt+1;
         end
         end
```



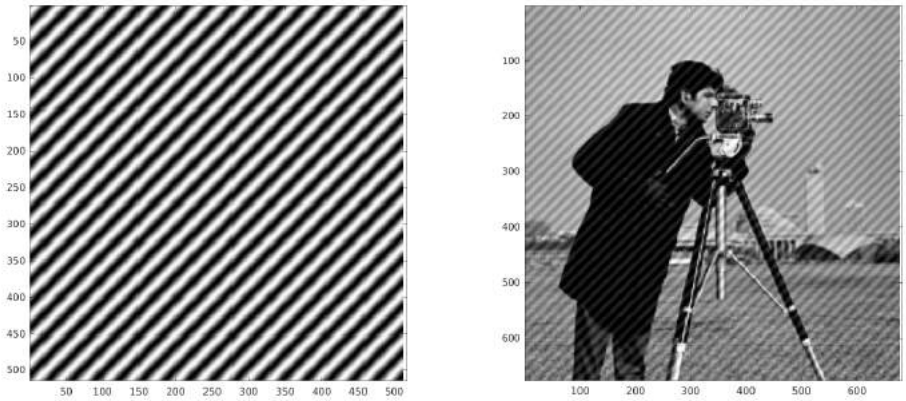
In [99]:
```
cpt=1;
for m = 1:8
for n = 1:8
tmp=zeros(8,8);
tmp(m,n)=1;
subplot(8,8,cpt); imagesc (  idct2(tmp ));colormap(gray)
cpt=cpt+1;
end
end
```



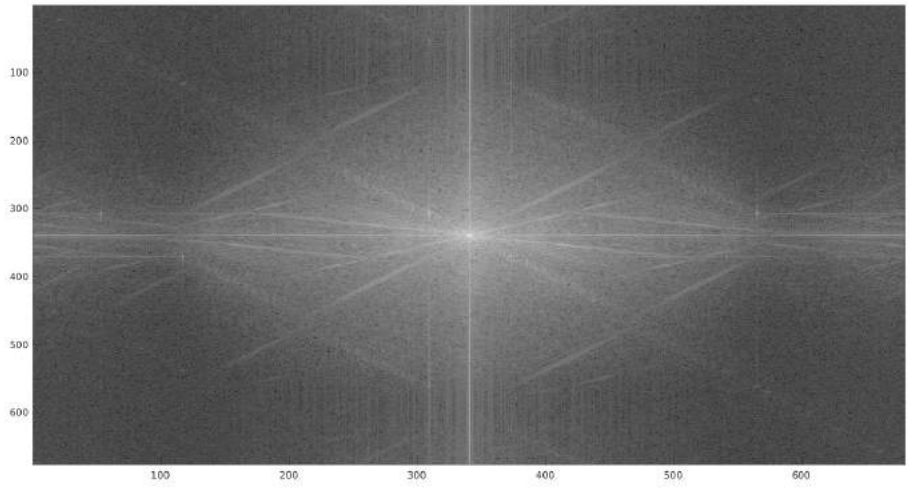
2.4 4) Débruitage par Fourier

Soit les 2 images suivantes, à l'aide la transformée de fourrier fft2 et de son inverse iftt2 "débruitez" la seconde images. On prendra le soin de représenter la transformée de fourrier des images. On pourra utiliser une représentation 3d de la transformée de Fourier en échelle logarithme pour l'amplitude.

In [101]:
```
% images disponible dossier partagé
Im1=rgb2gray(imread('sinus_diag.png'));
Im2=rgb2gray(imread('imag_bruite.png'));
subplot(1,2,1); imagesc(Im1);axis('image');colormap(gray)
subplot(1,2,2); imagesc(Im2);axis('image');
```

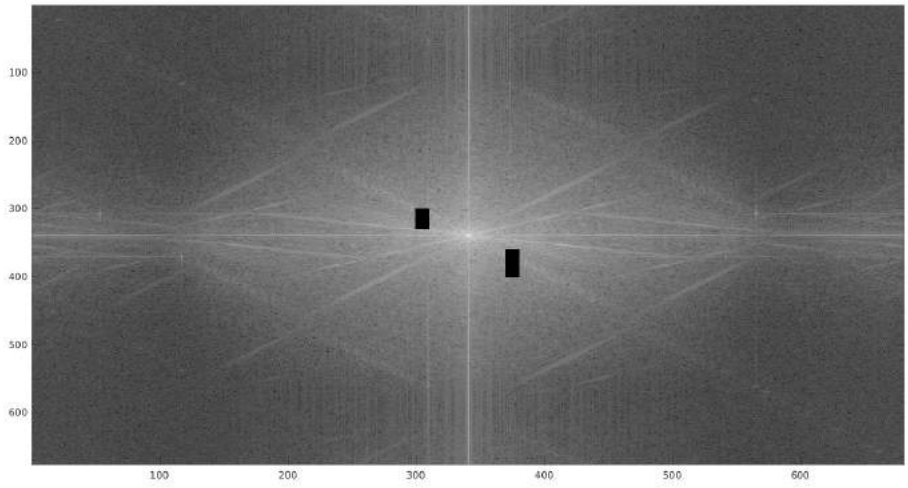


In [102]:
```
bf=fftshift(fft2(Im2));
figure,imagesc(log(abs(bf)));colormap(gray)
```



In [103]:
```
for m =300:330
    for n =300:310
        bf(m,n) = 0;
    end
end
for m = 360:400
     for n =370:380
        bf(m,n) = 0;
    end
end
```

```
figure,imagesc(log(abs(bf)));colormap(gray)
```




```
In [104]: ptnfx = real(ifft2(ifftshift(bf)));
          imshow(ptnfx,[]);
```

```
Warning: Image is too big to fit on screen; displaying at 67%
> In images.internal.initSize (line 71)
  In imshow (line 328)
```

2.5 5) Filtre passe bas ou passe haut

Reprendre l'image de Léna, à l'aide de la transformée de Fourier discrète sur les 3 couleurs filtrer : * Les "hautes" fréquences (filtre passe bas) * Les "basses" fréquences (filtre passe haut) Afficher les résultat obtenus

Remarque : On sélectionnera une zone carrée (à prendre ou enlever) sur la fft de l'image (pas shiftfft2) !

```
In [105]: Im3=imread('Lena_couleur_bruitee.jpg');
          imagesc(Im3);axis('image');
```

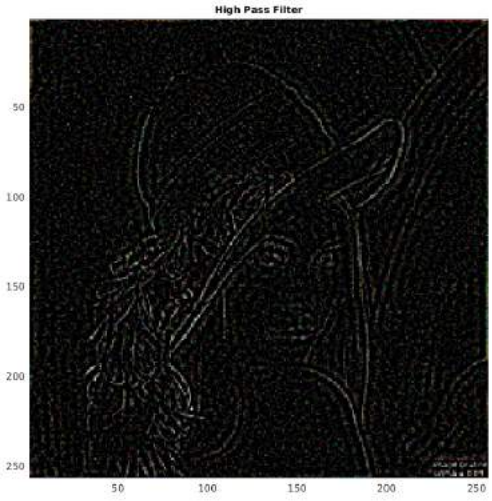

```
In [106]: bf1=fftshift(fft2(Im3(:,:,1)));
          bf2=fftshift(fft2(Im3(:,:,2)));
          bf3=fftshift(fft2(Im3(:,:,3)));

In [107]: [x,y]=meshgrid(-128:217,-128:127);
          z=sqrt(x.^2+y.^2);
          c=(z<40);

          cfl1=bf1.*c(1:256,1:256);
          cfl2=bf2.*c(1:256,1:256);
          cfl3=bf3.*c(1:256,1:256);

In [108]: O=real(ifft2(ifftshift(cfl1)));
          O=(O/max(max(O)))*255;
          Im3(:,:,1)=O;

          O=real(ifft2(ifftshift(cfl2)));
          O=(O/max(max(O)))*255;
```

```
Im3(:,:,2)=O;

O=real(ifft2(ifftshift(cfl3)));
O=(O/max(max(O)))*255;
Im3(:,:,3)=O;
```

In [145]:
```
imagesc(Im3);axis('image');
title('Low Pass Filter')
```

In [147]:
```
[x,y]=meshgrid(-128:217,-128:127);
z=sqrt(x.^2+y.^2);
c=(z>40);

cfl1=bf1.*c(1:256,1:256);
cfl2=bf2.*c(1:256,1:256);
cfl3=bf3.*c(1:256,1:256);

O=real(ifft2(ifftshift(cfl1)));
O=(O/max(max(O)))*255;
Im3(:,:,1)=O;

O=real(ifft2(ifftshift(cfl2)));
O=(O/max(max(O)))*255;
Im3(:,:,2)=O;

O=real(ifft2(ifftshift(cfl3)));
O=(O/max(max(O)))*255;
Im3(:,:,3)=O;
```

```
In [148]: imagesc(Im3);axis('image');
          title('High Pass Filter')
```



2.6 6) Débruitage

Peut on débruiter cette image à l'aide d'un filtre passe bas ?

```
In [112]: Im3=imread('Lena_couleur_bruitee.jpg');
          imagesc(Im3);axis('image');
```

```
In [113]: Im3(:,:,1)=imgaussfilt(Im3(:,:,1), 0.8);
          Im3(:,:,2)=imgaussfilt(Im3(:,:,2),  0.8);
          Im3(:,:,3)=imgaussfilt(Im3(:,:,3),  0.8);

In [114]: imagesc(Im3);axis('image');
```

2.7 7) Interpolation d'image par FFT

On souhaite "agrandir" la taille d'un image en doublant sa taille et sa résolution. On imagine alors la méthode suivante : 1. Décomposition de l'image F sur une base de Fourier discrète 2D, en utilisant l'algorithme de FFT 2D.On obtient alors une matrice de coefficients F de taille même taille que l'image. 2. Plongement de la matrice F dans une matrice G de taille double en ajoutant des zéros correspondant à des coefficients de haute fréquence. 3. FFT 2D inverse de la matrice G pour obtenir une image G de taille double.

```
In [115]: Im4=im2double(imread('cameraman.tif'));colormap(gray);
          imagesc(Im4);axis('image');title('Original Image');colormap(gray);
```

In [116]:
```
K = 2;
I = im2double(imread('cameraman.tif'));
S = size(I);

F = fft2(I);
F2 = repmat( K*K*F , K , K);

Mask = zeros(K*S);
Mask(1:S(1)/2+1 , 1:S(2)/2+1) = 1;
Mask(end-S(1)/2 : end , 1:S(2)/2+1) = 1;
Mask(1:S(1)/2+1 , end-S(2)/2 :end ) = 1;
Mask(end-S(1)/2 : end , end-S(2)/2 :end ) = 1;

F2 = Mask.*F2;

I2 = real( ifft2( F2 ));

imagesc(I2);axis('image');title('Rescale Image');colormap(gray);
```

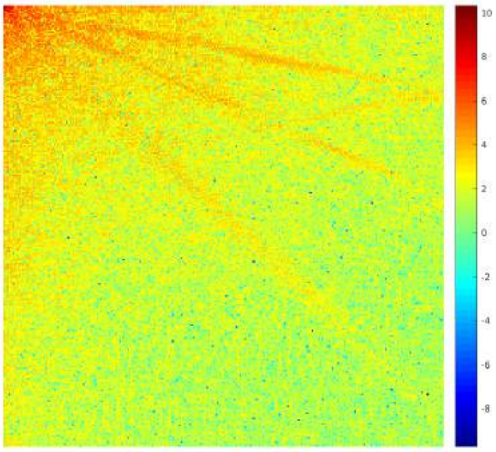
2.8 8) Compression d'image

Nous allons maintenant effectuer une compression de type JPEG. Cette compression utilise la transformée en Cosinus Discret (fonction matlab dct2 et idct2).

2.8.1 8.1) Regarder le résultat obtenu en effectuant la DCT de l'image complète.

Remarquez que l'origine des fréquences se trouvent en haut à gauche de l'image et mettre à zéro les valeurs qui sont en dessous de l'antidiagonale. Calulez la DCT inverse et afficher l'image obtenue. Estimez l'erreur induite par la mise à zéro en faisant la différence avec l'image originale.

```
In [117]: I = imread('cameraman.tif');
          J = dct2(I);
          figure
          imshow(log(abs(dct2(I))),[])
          colormap(gca,jet(64))
          colorbar
```

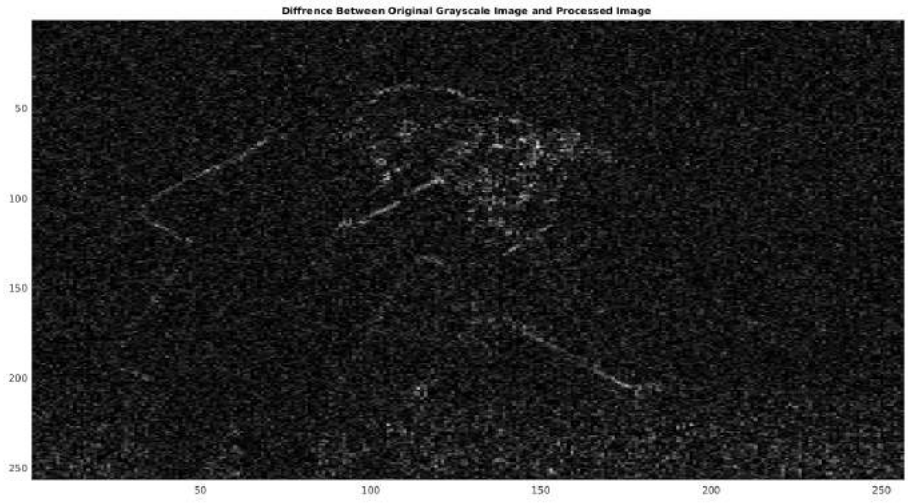


In [118]:
```
J(abs(J) < 10) = 0;
```

In [119]:
```
K = idct2(J);
```

In [120]:
```
figure
imshowpair(I,K,'montage')
title('Original Grayscale Image (Left) and Processed Image (Right)');
```

In [121]:
```
figure,imagesc(abs((im2double(I)*256)-K));colormap(gray)
title('Diffrence Between Original Grayscale Image and Processed Image ');
```



2.8.2 8.2) Méthode naïve de compression

Regarder le résultat obtenus en supprimant 10%, 20% ... des coefficients les moins significatifs de la transformée.

```
In [122]: %%file ImgCom.m
          function [out] = ImgCom (infile,bx,by)
          a = rgb2gray(imread(infile));
           dvalue = double(a);
              dx=size(dvalue,1);
              dy=size(dvalue,2);
              % if input image size is not a multiple of block size image is resized
              modx=mod(dx,bx);
              mody=mod(dy,by);
              dvalue=dvalue(1:dx-modx,1:dy-mody);
              % the new input image dimensions (pixels)
              dx=dx-modx;
              dy=dy-mody;
              % number of non overlapping blocks required to cover
              % the entire input image
              nbx=size(dvalue,1)/bx;
              nby=size(dvalue,2)/by;

              % the output compressed image
              matrice=zeros(bx,by);
              % the compressed data
              m_u=zeros(nbx,nby);
              m_l=zeros(nbx,nby);
              mat_log=logical(zeros(bx,by));
```

```
            posbx=1;
            for ii=1:bx:dx
                posby=1;
                for jj=1:by:dy
                    % the current block
                    blocco=dvalue(ii:ii+bx-1,jj:jj+by-1);
                    % the average gray level of the current block
                    m=mean(mean(blocco));
                    % the logical matrix correspoending to the current block
                    blocco_binario=(blocco>=m);
                    % the number of pixel (of the current block) whose gray level
                    % is greater than the average gray level of the current block
                    K=sum(sum(double(blocco_binario)));
                    % the average gray level of pixels whose level is GREATER than
                    % the block average gray level
                    mu=sum(sum(double(blocco_binario).*blocco))/K;
                    % the average gray level of pixels whose level is SMALLER than
                    % the block average gray level
                    if K==bx*by
                        ml=0;
                    else
                        ml=sum(sum(double(~blocco_binario).*blocco))/(bx*by-K);
                    end
                    % the COMPRESSED DATA which correspond to the input image
                    m_u(posbx,posby)=mu;                              %---> the m_u matrix (see
                    m_l(posbx,posby)=ml;                              %---> the m_l matrix
                    mat_log(ii:ii+bx-1,jj:jj+by-1)=blocco_binario;  %---> the logical matrix
                    % the compressed image
                    matrice(ii:ii+bx-1,jj:jj+by-1)=(double(blocco_binario).*mu)+(double(~blocc
                    posby=posby+1;
                end
                posbx=posbx+1;
            end

            % display the logical matrix

            out=uint8(matrice);

        end
```

```
Created file '/home/nabil.madali/ImgCom.m'.
```

In [123]:
```
imshowpair(rgb2gray(imread('lena.png')),ImgCom ('lena.png',2,2),'montage');
title('Original Grayscale Image  and Compressed Image With  2*2 Window');
```


```
In [124]: imshowpair(rgb2gray(imread('lena.png')),ImgCom ('lena.png',10,10),'montage');
          title('Original Grayscale Image  and Compressed Image With   10*10 Window');
```


```
In [125]: imshowpair(rgb2gray(imread('lena.png')),ImgCom ('lena.png',20,20),'montage');
          title('Original Grayscale Image  and Compressed Image  With  20*20 Window');
```

2.8.3 8.3) Répétez l'opération précédentes par blocs de 8x8 (Compression Jpeg)

Ce type de compression repose sur plusieurs étapes : * découper l'image en blocs de taille identique (8x8 en pratique). * calculer la DCT de chaque bloc. * représenter le résultat de la DCT 2D en regroupant tous les blocs par ordre de coefficient. * supprimer les coefficients qui ont la plus petite amplitude (10%, 20% on controle le taux de compression). * calculer les DCT inverse de chaque bloc. * regrouper les blocs dans une même image.

Observez l'irrégularité spatiale de la répartition de l'information.

Remarque Pour la compression Jpeg on transpose d'abord l'image dans une représentaion YCrCb puis on diminue par deux la taille des matrices contenant les coefficients codant la chrominance de l'image (Cr et Cb). Pour 4 pixels, on n'en garde qu'un seul représentant la moyenne de ceux-ci, on appelle cela le "Downsampling". Voir https://fr.wikipedia.org/wiki/JPEG

In [126]:
```
%%file JpgCom.m
function [out] = JpgCom (infile,quality)
I =rgb2gray(imread(infile));
I1=I;
[row coln]= size(I);
I= double(I);

I = I - (128*ones(size(I)));



Q50 = [ 16 11 10 16 24 40 51 61;
     12 12 14 19 26 58 60 55;
     14 13 16 24 40 57 69 56;
     14 17 22 29 51 87 80 62;
```

```
     18 22 37 56 68 109 103 77;
     24 35 55 64 81 104 113 92;
     49 64 78 87 103 121 120 101;
     72 92 95 98 112 100 103 99];

 if quality > 50
     QX = round(Q50.*(ones(8)*((100-quality)/50)));
     QX = uint8(QX);
 elseif quality < 50
     QX = round(Q50.*(ones(8)*(50/quality)));
     QX = uint8(QX);
 elseif quality == 50
     QX = Q50;
 end
```

```
DCT_matrix8 = dct(eye(8));
iDCT_matrix8 = DCT_matrix8';

dct_restored = zeros(row,coln);
QX = double(QX);

for i1=[1:8:row]
    for i2=[1:8:coln]
        zBLOCK=I(i1:i1+7,i2:i2+7);
        win1=DCT_matrix8*zBLOCK*iDCT_matrix8;
        dct_domain(i1:i1+7,i2:i2+7)=win1;
    end
end

for i1=[1:8:row]
    for i2=[1:8:coln]
        win1 = dct_domain(i1:i1+7,i2:i2+7);
        win2=round(win1./QX);
        dct_quantized(i1:i1+7,i2:i2+7)=win2;
    end
end


for i1=[1:8:row]
    for i2=[1:8:coln]
        win2 = dct_quantized(i1:i1+7,i2:i2+7);
        win3 = win2.*QX;
        dct_dequantized(i1:i1+7,i2:i2+7) = win3;
    end
```

```
        end

        for i1=[1:8:row]
            for i2=[1:8:coln]
                win3 = dct_dequantized(i1:i1+7,i2:i2+7);
                win4=iDCT_matrix8*win3*DCT_matrix8;
                dct_restored(i1:i1+7,i2:i2+7)=win4;
            end
        end
        I2=dct_restored;




        K=mat2gray(I2);
        out=K;
        end
```

```
Created file '/home/nabil.madali/JpgCom.m'.
```

In [127]:
```
imshowpair(rgb2gray(imread('lena.png')), JpgCom ('lena.png',90),'montage');
title('Original Image  and Compressed Image  With  10% Compression');
```

In [128]:
```
imshowpair(rgb2gray(imread('lena.png')), JpgCom ('lena.png',70),'montage');
title('Original Image  and Compressed Image  With  30% Compression');
```

In [129]:
```
imshowpair(rgb2gray(imread('lena.png')), JpgCom ('lena.png',50),'montage');
title('Original Image  and Compressed Image  With  50% Compression');
```

In [130]:
```
imshowpair(rgb2gray(imread('lena.png')), JpgCom ('lena.png',30),'montage');
title('Original Image  and Compressed Image  With  70% Compression');
```

3 Partie II : Ondelettes

Suivant le même principe que pour le son on peut par "tensorisation" d'ondellettes 1d faire naturellement des dimensions supérieurs. Par contre au lieu d'avoir 2 niveaux de décomposition (Approximation et détail) on en obtient 4 comme produits de approximation/détail en x et en y.

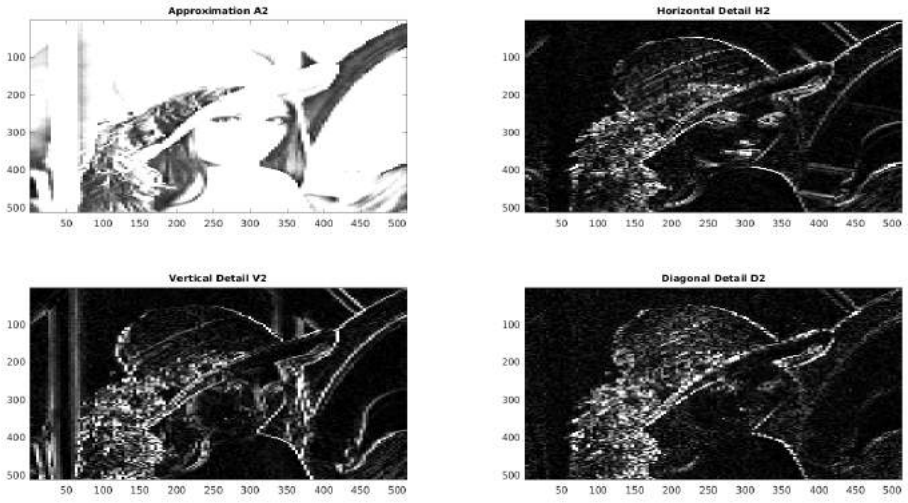
3.1 1) Prise en main dwt2

3.1.1 a) Décomposition 2 niveaux

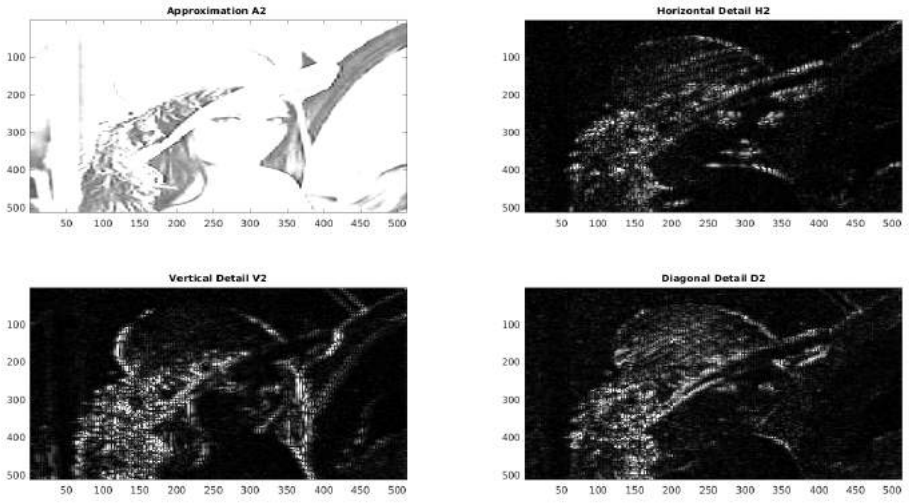
A l'aide de la commande matlab dwt2 faire une décomposition 2 niveaux de l'image suivant et sur une même image (taille identique) représenter les 4 résultats obtenus. A fin de rendre le résultat plus lisible on pourra "amplifier" ou normaliser les éléments de détail... Utiliser différentes ondelettes, Haar, db4 ...

```
In [131]: X=rgb2gray(imread('http://vision.gel.ulaval.ca/~jflalonde/cours/4105/h14/tps/results/t
          imagesc(X);
          axis('image');colormap(gray)
```


```
In [132]: [C,S] = wavedec2(X,2,'haar');

          cA2 = appcoef2(C,S,'haar',2);
          cH2 = detcoef2('h',C,S,2);
          cV2 = detcoef2('v',C,S,2);
          cD2 = detcoef2('d',C,S,2);
          cH1 = detcoef2('h',C,S,1);
          cV1 = detcoef2('v',C,S,1);
          cD1 = detcoef2('d',C,S,1);

          A2 = wrcoef2('a',C,S,'haar',2);
          H1 = wrcoef2('h',C,S,'haar',1);
          V1 = wrcoef2('v',C,S,'haar',1);
          D1 = wrcoef2('d',C,S,'haar',1);
          H2 = wrcoef2('h',C,S,'haar',2);
          V2 = wrcoef2('v',C,S,'haar',2);
          D2 = wrcoef2('d',C,S,'haar',2);

          colormap(gray)

          subplot(2,2,1);image((wcodemat(A2,192)));title('Approximation A2')
          subplot(2,2,2);image((wcodemat(H2,192)));title('Horizontal Detail H2')
          subplot(2,2,3);image((wcodemat(V2,192)));title('Vertical Detail V2')
          subplot(2,2,4);image((wcodemat(D2,192)));title('Diagonal Detail D2')
```



In [133]:
```
[C,S] = wavedec2(X,2,'db4');

cA2 = appcoef2(C,S,'db4',2);
cH2 = detcoef2('h',C,S,2);
cV2 = detcoef2('v',C,S,2);
cD2 = detcoef2('d',C,S,2);
cH1 = detcoef2('h',C,S,1);
cV1 = detcoef2('v',C,S,1);
cD1 = detcoef2('d',C,S,1);

A2 = wrcoef2('a',C,S,'db4',2);
H1 = wrcoef2('h',C,S,'db4',1);
V1 = wrcoef2('v',C,S,'db4',1);
D1 = wrcoef2('d',C,S,'db4',1);
H2 = wrcoef2('h',C,S,'db4',2);
V2 = wrcoef2('v',C,S,'db4',2);
D2 = wrcoef2('d',C,S,'db4',2);

colormap(gray)

subplot(2,2,1);image((wcodemat(A2,192)));title('Approximation A2 ')
subplot(2,2,2);image((wcodemat(H2,192)));title('Horizontal Detail H2')
subplot(2,2,3);image((wcodemat(V2,192)));title('Vertical Detail V2')
subplot(2,2,4);image((wcodemat(D2,192)));title('Diagonal Detail D2')
```

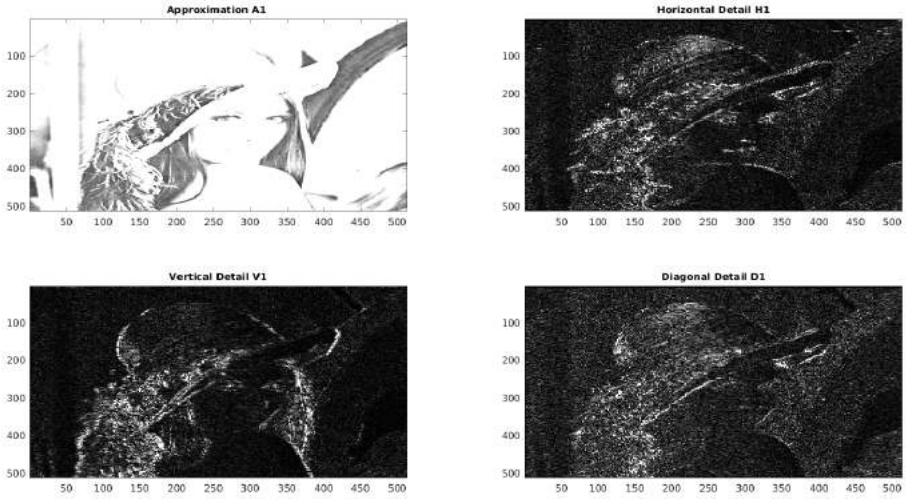


3.1.2 b) Décomposition multiniveaux

Répéter l'opération et la représentation sur plusieurs niveaux.

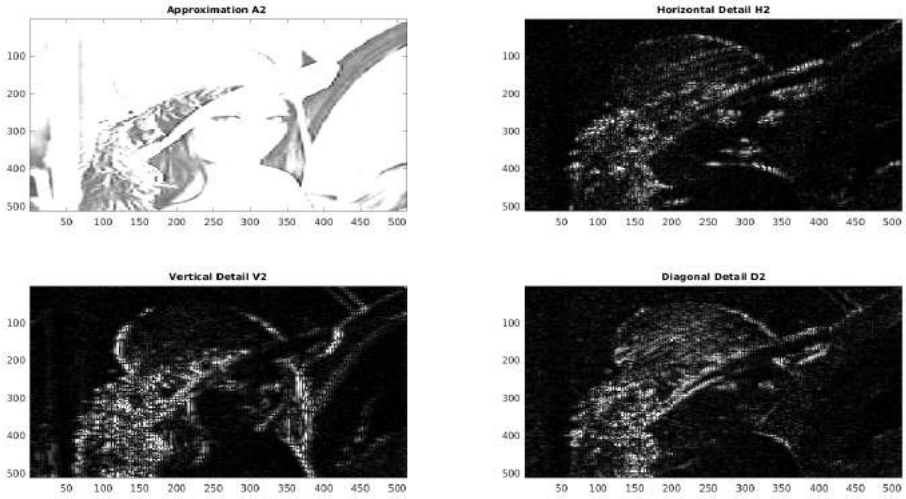
```
In [134]: [C,S] = wavedec2(X,4,'db4');

          A1 = wrcoef2('a',C,S,'db4',1);
          H1 = wrcoef2('h',C,S,'db4',1);
          V1 = wrcoef2('v',C,S,'db4',1);
          D1 = wrcoef2('d',C,S,'db4',1);

          A2 = wrcoef2('a',C,S,'db4',2);
          H2 = wrcoef2('h',C,S,'db4',2);
          V2 = wrcoef2('v',C,S,'db4',2);
          D2 = wrcoef2('d',C,S,'db4',2);

          A3 = wrcoef2('a',C,S,'db4',3);
          H3 = wrcoef2('h',C,S,'db4',3);
          V3 = wrcoef2('v',C,S,'db4',3);
          D3 = wrcoef2('d',C,S,'db4',3);

          A4 = wrcoef2('a',C,S,'db4',4);
          H4 = wrcoef2('h',C,S,'db4',4);
          V4 = wrcoef2('v',C,S,'db4',4);
          D4 = wrcoef2('d',C,S,'db4',4);

In [135]: colormap(gray)

          subplot(2,2,1);image((wcodemat(A1,192)));title('Approximation A1 ')
```

```
subplot(2,2,2);image((wcodemat(H1,192)));title('Horizontal Detail H1')
subplot(2,2,3);image((wcodemat(V1,192)));title('Vertical Detail V1')
subplot(2,2,4);image((wcodemat(D1,192)));title('Diagonal Detail D1')
```



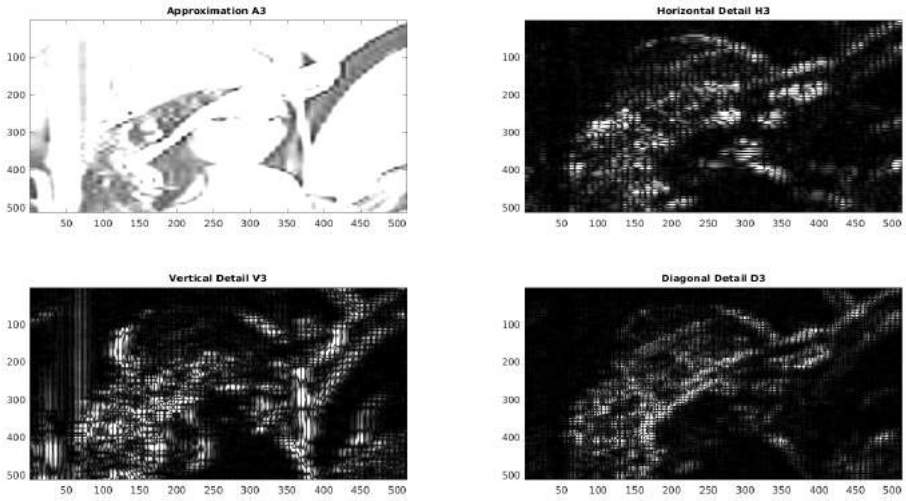
In [136]:
```
colormap(gray)
```

```
subplot(2,2,1);image((wcodemat(A2,192)));title('Approximation A2 ')
subplot(2,2,2);image((wcodemat(H2,192)));title('Horizontal Detail H2')
subplot(2,2,3);image((wcodemat(V2,192)));title('Vertical Detail V2')
subplot(2,2,4);image((wcodemat(D2,192)));title('Diagonal Detail D2')
```



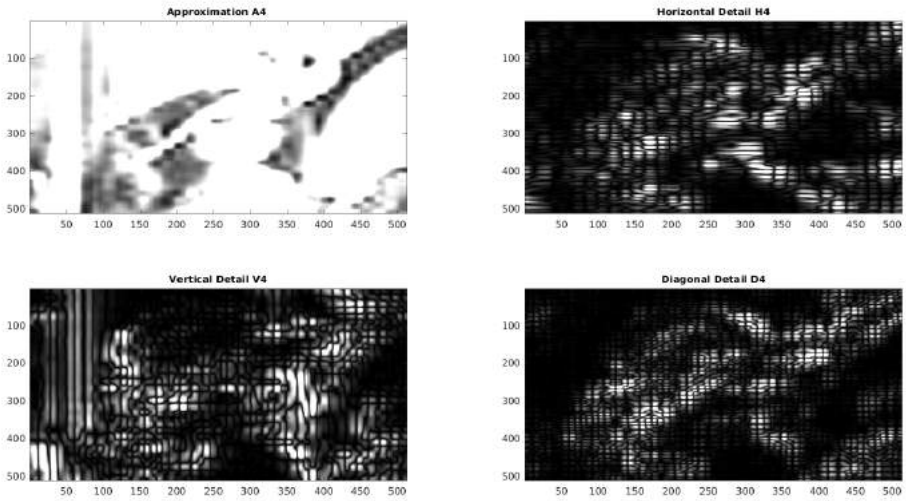
In [137]: colormap(gray)

```
subplot(2,2,1);image((wcodemat(A3,192)));title('Approximation A3 ')
subplot(2,2,2);image((wcodemat(H3,192)));title('Horizontal Detail H3')
subplot(2,2,3);image((wcodemat(V3,192)));title('Vertical Detail V3')
subplot(2,2,4);image((wcodemat(D3,192)));title('Diagonal Detail D3')
```



In [138]: colormap(gray)

```
subplot(2,2,1);image((wcodemat(A4,192)));title('Approximation A4 ')
subplot(2,2,2);image((wcodemat(H4,192)));title('Horizontal Detail H4')
subplot(2,2,3);image((wcodemat(V4,192)));title('Vertical Detail V4')
subplot(2,2,4);image((wcodemat(D4,192)));title('Diagonal Detail D4')
```



3.2 2) Débruitage

Par méthode de seuillage sur la décomposition "débruiter" l'image de Léna. Utiliser différentes ondelletes.

Mots clefs : dwt2,idwt2

```
In [139]: Lena=imread('Lena_couleur_bruitee.jpg');
          I=Lena;
          imagesc(Lena)
          axis('image');
```


```
In [140]: th='s';
          for ii=1:3
          b=I(:,:,ii);
          [a1,b1,c1,d1]=dwt2(b,'db2');
          [a2,b2,c2,d2]=dwt2(a1,'db2');

          v1=(median(abs(b2(:)))/0.6745);
          b2= wthresh(b2,th,v1);
          v2=(median(abs(c2(:)))/0.6745);
          c2= wthresh(c2,th,v2);
          v3=(median(abs(d2(:)))/0.6745);
          d2= wthresh(d2,th,v3);

          v1=(median(abs(b1(:)))/0.6745);
          b1= wthresh(b1,th,v1);
```

```
v2=(median(abs(c1(:)))/0.6745);
c1= wthresh(c1,th,v2);
v3=(median(abs(d1(:)))/0.6745);
d1= wthresh(d1,th,v3);


a1=idwt2(a2,b2,c2,d2,'db2');
c=idwt2(a1(1:129,1:129),b1,c1,d1,'db2');

I(:,:,ii)=(c/max(max(c)))*255;
end
```

```
imshowpair(Lena, I,'montage');
title('Original Image  and Denoised Image ');
```

3.3 3) Tatouage numérique (simplifié)

Le tatouage est l'art d'altérer ou de marquer un média (une image, un son, une vidéo...) de sorte qu'il contienne un message le plus souvent en rapport avec l'dentité du média et le plus souvent de manière imperceptible. Dans le cas de tatouage d'image, le message sera ń mélangé ż aux pixels sans altérations visibles de l'image. Ce message devra être détectable même après que l'image aura subi des traitements comme un filtrage rehausseur de contours, redimensionnement, changement de format etc.

Depuis la naissance du tatouage numérique moderne, dans le début des années 1990, tout le monde s'accorde sur le fait qu'un bon système de tatouage devrait être robuste (être ń résistant ż) à de simples attaques de désynchronisations comme la rotation, le changement d'échelle ou

au rognage de l'image. Plusieurs systèmes ont été proposés, fin des années 90 - début des années 2000, mais ceux-ci ne résistent pas à toutes les attaques et en particulier rares sont ceux qui résistent à l'attaque ń print-and-scan ż ou à l'attaque de rognage (cropping). De plus ils présentent de fortes faiblesses face aux attaques malveillantes qui consistent à sciemment perturber le système...

Nous allons utiliser une méthode un introduisant une image (marque) 32x32 binaire (noir et blanc ou 0 et 1) dans une image 512x512.

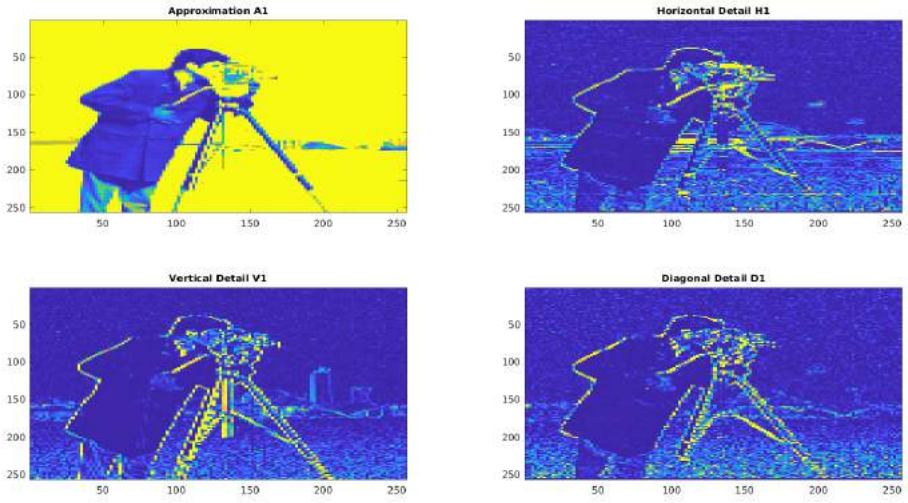
3.3.1 Algorithme :

- Effectuer une décomposition dwt2 à 3 niveaux a l'image originale.
- Diviser les matrices des coefficients LH13, HL13, LH23, HL23 en des blocks de tailles 4x4
- Modifier le premier élément du bloc par la valeur de la marque, qui peut valoir 0 ou 255... (par exemple pour notre tatouage binaire prendre 0 => 0 et 1 => 255).
- Reconstruire une image tatouée avec idwt2.

Proposer un algorithme de décodage pour retrouver la marque (image binaire 32x32) et tester sans changer de taille de l'image, la robustesse du tatouage à l'encodage jpg.

```
In [141]: Text='Bonne vacance';
          wavename = 'haar';
          data = zeros(1,length(Text));
          for i = 1 : length(Text)
              d = Text(i) + 0;
              data(i) = d;
          end

          im = imread('cameraman.tif');
          imshow(im),title('Original Image')
```

In [142]:
```
[cA1,cH1,cV1,cD1] = dwt2(im,wavename);

A1 = upcoef2('a',cA1,wavename,1);
H1 = upcoef2('h',cH1,wavename,1);
V1 = upcoef2('v',cV1,wavename,1);
D1 = upcoef2('d',cD1,wavename,1);



 subplot(2,2,1); image(wcodemat(A1,192));
 title('Approximation A1')

 subplot(2,2,2); image(wcodemat(H1,192));
 title('Horizontal Detail H1')

 subplot(2,2,3); image(wcodemat(V1,192));
 title('Vertical Detail V1')

subplot(2,2,4); image(wcodemat(D1,192));
title('Diagonal Detail D1')
```



In [143]:
```
M=max(data);

normalize = data; %

n=length(data);


cH1(1,1) = -1*(n/10);
```

```
cH1(1,2) = -1*(M/10);

[~ , y]=size(cH1);


for i = 1 : ceil(n/2)
     cV1(i,y)= normalize(i); %
end



for i = ceil(n/2) + 1 : n
     cD1(i,y)=normalize(i);
end
figure;

Restore = idwt2 (cA1,cH1,cV1,cD1,wavename);
imshow(uint8(Restore));title('Original Image With Embedding msg')
```

In [144]:
```
[cA1r,cH1r,cV1r,cD1r] = dwt2 (Restore,wavename);


n = ceil ( abs((cH1r(1,1)*10)) );
M = ceil ( abs((cH1r(1,2)*10)) );

data = zeros(1,length(n));
normalize = zeros(1,length(n));

[x y]=size(cH1r);
```

```
            for i = 1 : ceil(n/2)
                normalize(i) = cV1r(i,y);
            end


            for i = ceil(n/2)+1 : n
                normalize(i) = cD1r(i,y);
                data = normalize ;
            end


            Text1='';

            for i = 1 : length(data)
                Text1 = horzcat(Text1,floor(data(i)));
            end

        Text1
```

```
Text1 =

    'Bonne vacance '
```

In [ ]:

In [ ]: