Alexandre MICHALIK - Paul-Emile Vétu 21/06/2021

Compte rendu TP Traitement Du Signal

<u>Partie 1 :</u> Création de notre propre image

<u>Question 1 :</u> Créer une image 512px*512px totalement noire avec en son centre un carré de pixel blancs de largeur 100px.

Afin de créer un carré de noir de largeur 512px et 512px de longueur, nous allons utiliser la commande suivante :

```
image=zeros(512,512);
```

La fonction zeros(x,y) va permet de créer une matrice de longueur x et de largeur y et va mettre à 0 toutes les valeurs de sa matrice. Le 0 correspond à la valeur 0 au niveau de gris.

Afin de créer un carré de pixel blancs de largeur 100px, nous utilisons l'instruction suivante :

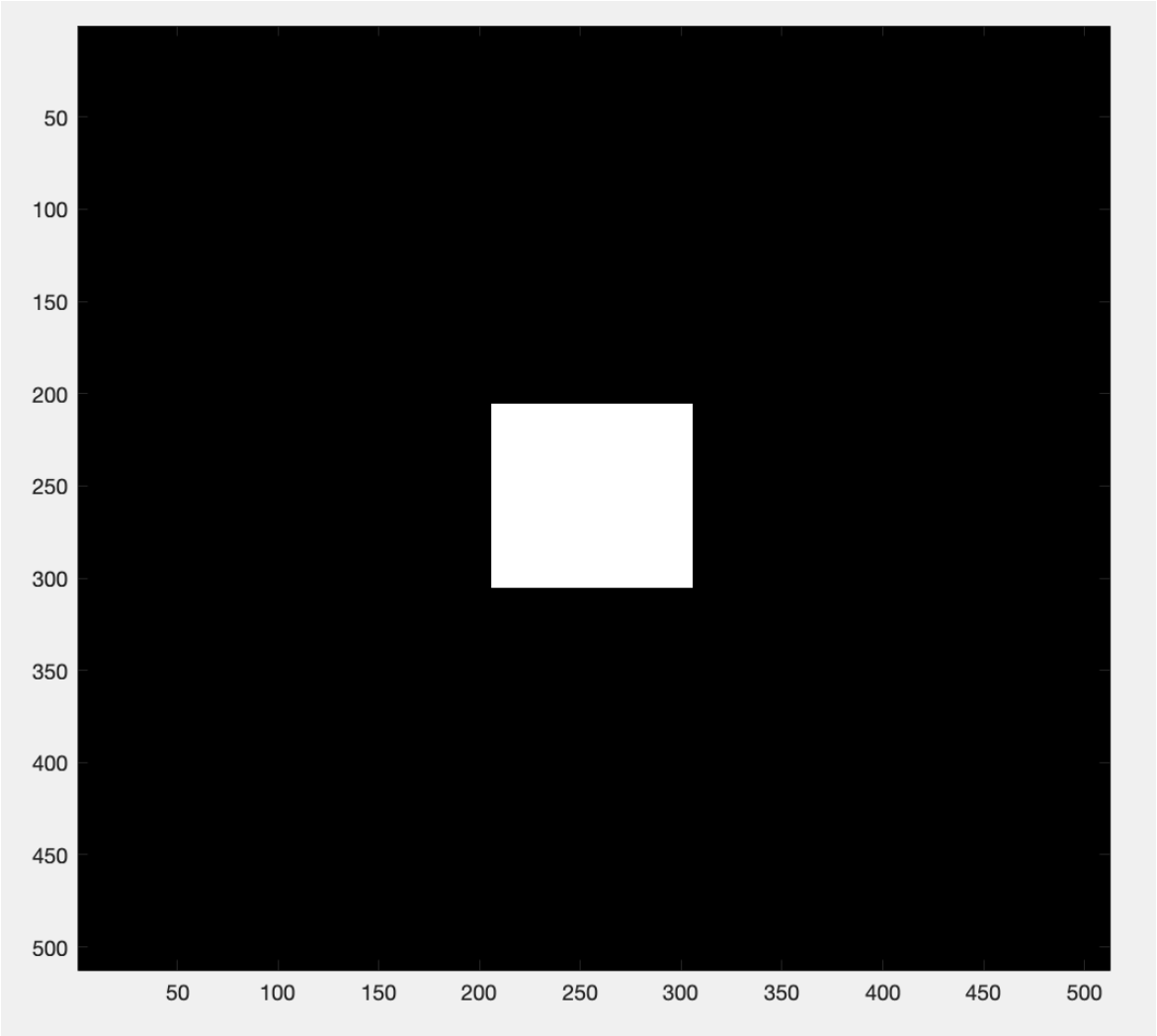
```
image(206:305,206:305)=255;
```

Pour cela nous allons définir la largeur et hauteur nécessaire pour placer ce carré au centre :

$\frac{512}{2}$= 256

256-50 = 206
256+49 = 305 car il faut compter le bit de départ.

Puis on affecte la teinte blanche qui correspond à la valeur 255.



Question 2 : Appliquer la transformée en cosinus discrète sur notre image.

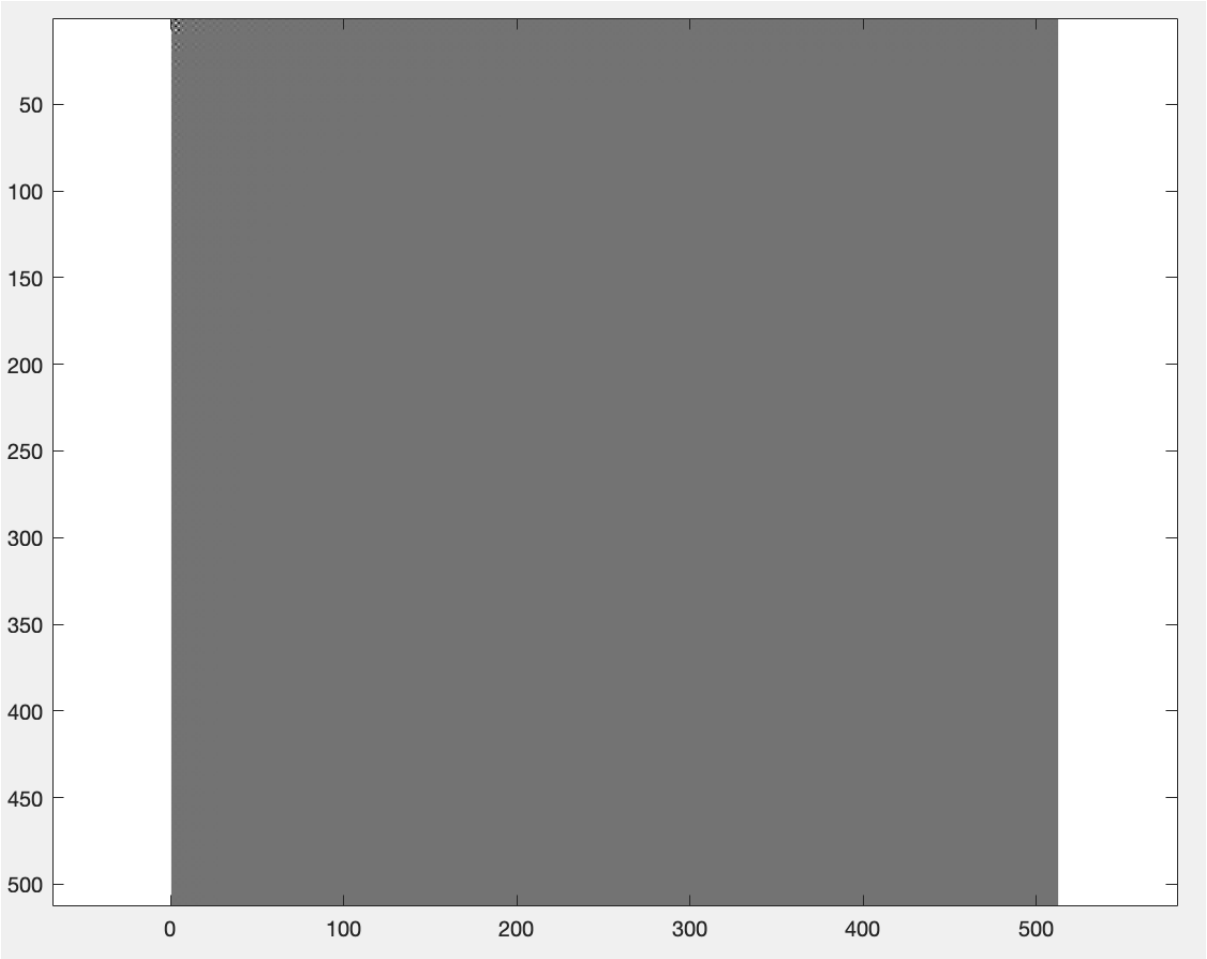
Afin d'effectuer la transformée en cosinus discrète on utilise la fonction suivante :

```
A = dct2(image);
```

Donc ici, on va affecter à la variable A la transformée en cosinus discrète de notre image noir avec notre centre blanc.

Question 3 : Visualiser le résultat de ce calcul.

Afin de visualiser la transformée en cosinus discrète, nous avons utilisés :



La transformée en cosinus discret sert d'obtenir une représentation d'un signal ou d'une image dans l'espace des fréquences.

On peut observer en haut à gauche de l'image, des carrés noir et blanc.

Question 4 : Grâce à la fonction idct2.m, reconstruire l'image initiale sur différents cas.

On va d'abord commencer par reconstruire l'image initiale et après nous ajusterons le pourcentage de plus grand coefficient que nous voulons garder.

Les différentes étapes à effectuer avant de faire cette transformée en cosinus discrète inverse :

- On retranscrit notre matrice en ligne
- On va donc obtenir un sorte de graphe ayant comme amplitude la valeur de chaque pixel
- On va trier les différentes valeurs dans l'ordre décroissant de ses valeurs
- On applique un % de valeur que nous voulons garder et mettons à 0 tous les autres pixel
- On fait la transformée inverse en cosinus discrète

```
B = A(:);
```
On affecte à la variable B, la matrice A 512*512 en transformée de cosinus discret sur une ligne.

```
B = sort(abs(B), 'descend');
```
Permet de trier la matrice de la valeur la plus grande à la valeur la plus petite.

```
C = length(B);
```
On affecte à C la longueur de la matrice B soit 262144 (512*512)

```
D = C * 0,4;
```
On multiplie par le pourcentage des plus grands coefficients de la DCT que nous voudrions garder, soit ici nous voulons garder 40% des plus grands coefficients.

```
D = round(D);
```
Nous arrondissons la valeur D pour qu'elle soit entière.

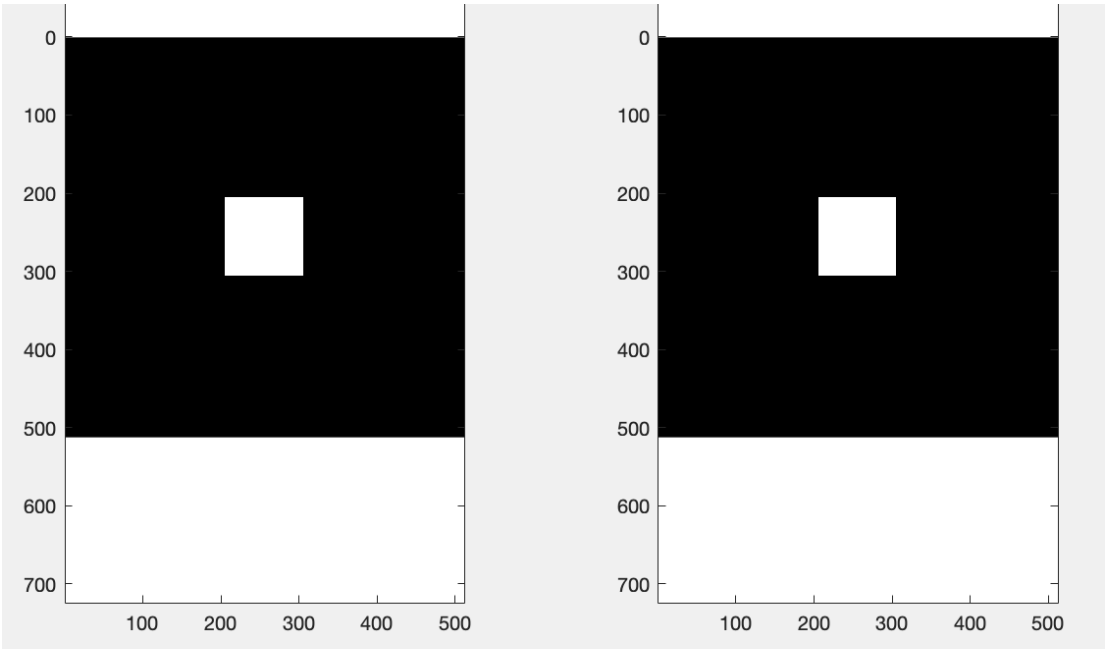
```
E=B(D);
```
Nous affectons à E, la valeur qui à partir de laquelle nous devrons affecter à 0 sur la matrice A à la prochaine opération.

```
A(abs(A)<E)=0;
```
Pour chaque valeur absolue de A inférieur à E calculé précédemment, nous lui affectons la valeur 0.

```
I = idct2(A);
```
Pour finir, nous faisons la transformée de cosinus discret inverse.

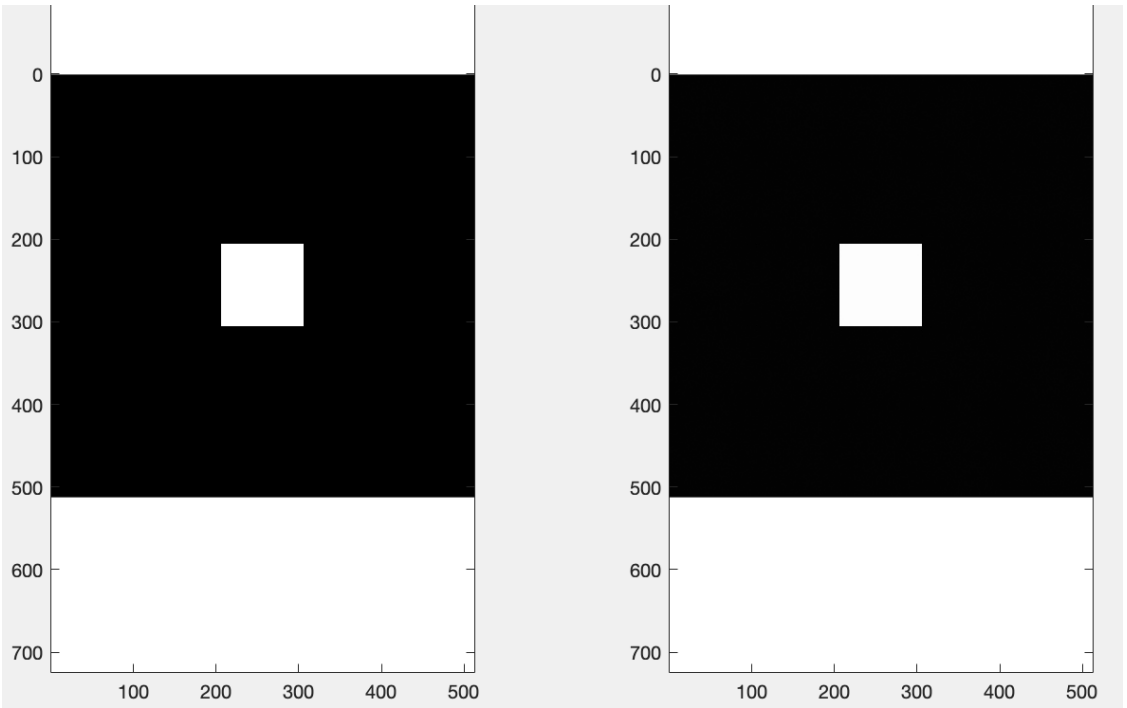
a) Les 80% plus grands coefficient de la DCT.

Pour cela on reprend le code et on affecte 0,8 à la place de 0,4.



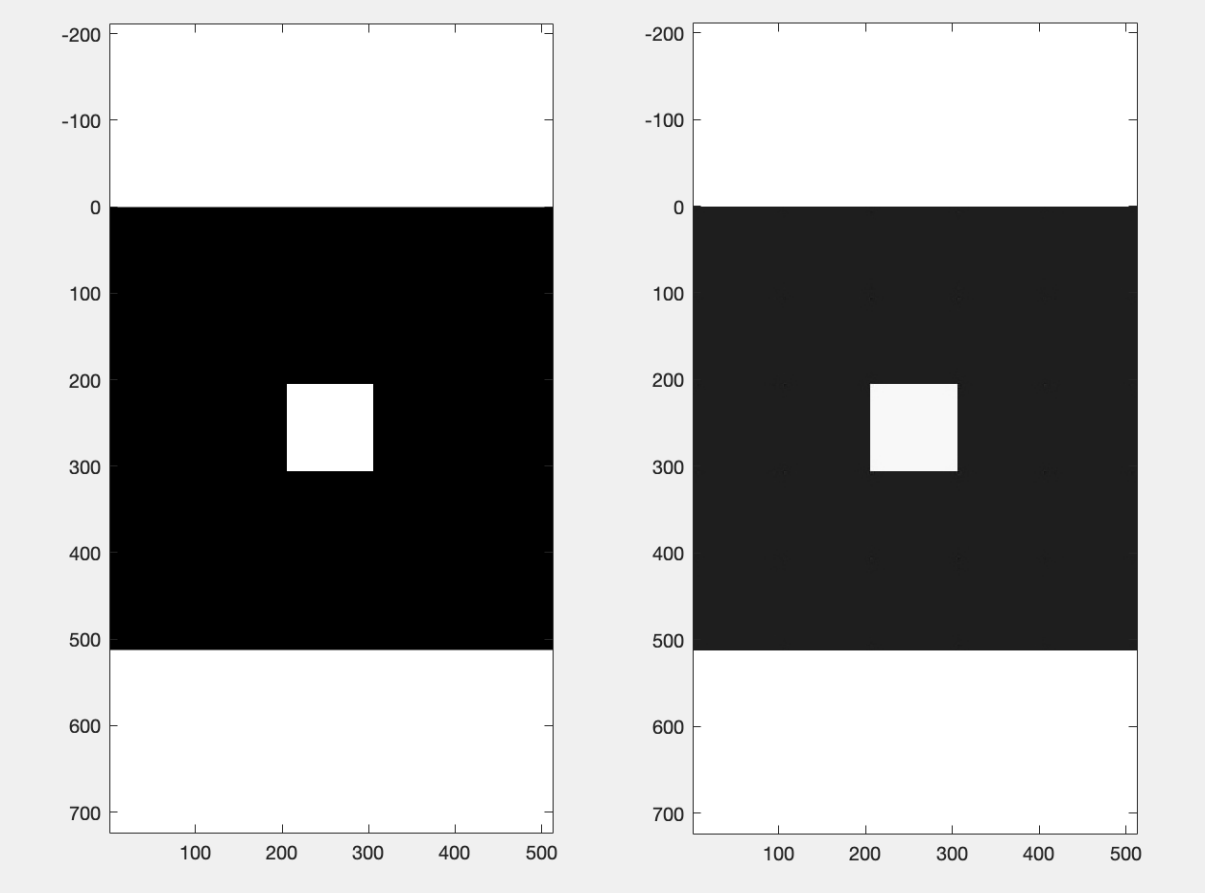
b) Les 50% plus grands coefficient de la DCT.

Pour cela on reprend le code et on affecte 0,5 à la place de 0,4.



c) Les 20% les plus grands coefficients de la DCT.

Pour cela on reprend le code et on affecte 0,5 à la place de 0,4.



Sur les images, il est assez compliqué de voir les différents résultats en fonction des différentes valeurs de plus grand coefficients de la DCT choisi. Il a partie 2, il sera beaucoup plus visible.

Sur cet exemple, même en gardant un très petit nombre de valeurs, on a une image assez convaincante.

Après avoir revu les différents cas, nous avons choisi de garder les 50% plus grands coefficients de la DCT, car nous avons observé pour le cas à 20%, une légère perte de couleur qui tend légèrement vers le gris.

La DCT sert à compresser une image. En fonction du nombre de pourcentage de valeur à garder, l'image reste plus ou moins nette.

<u>Partie 2 :</u> Application avec une image

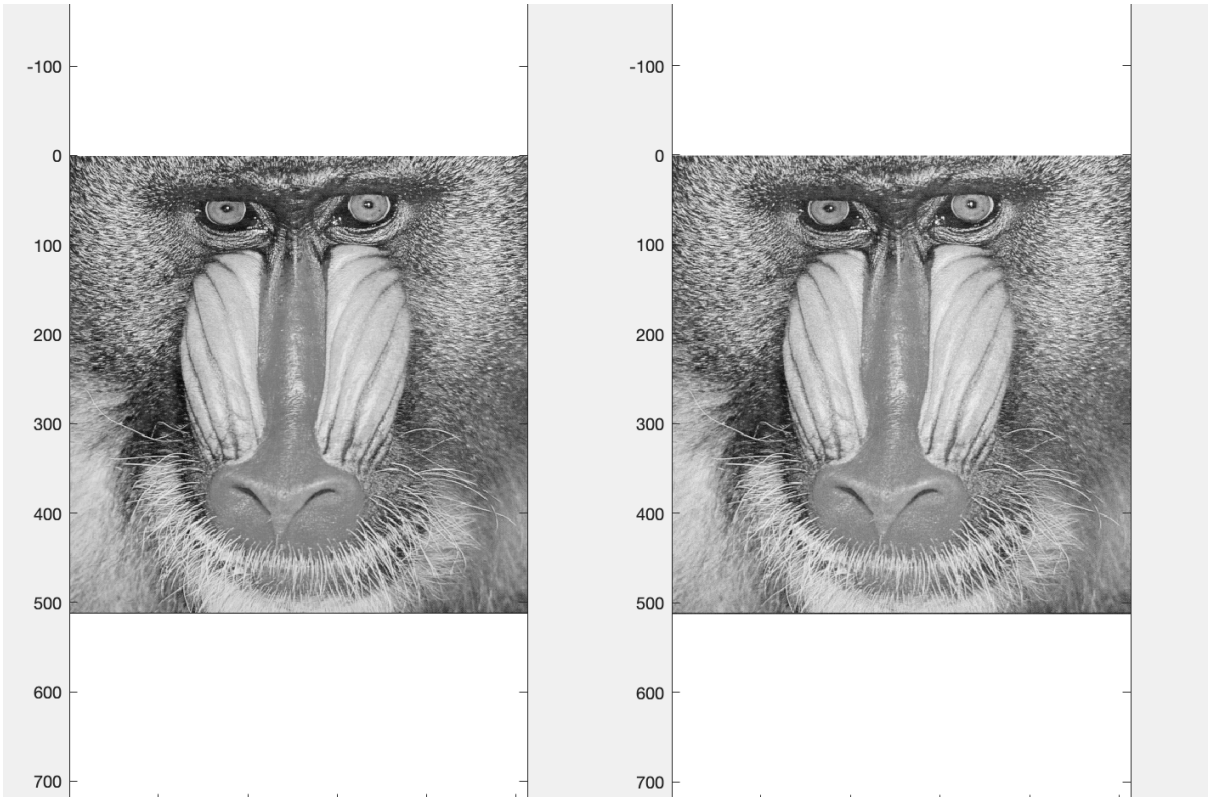
Pour le test avec une image, nous avons utilisé l'image numéro 1 avec le singe.

Nous avons ré-adapté le code de la partie 1 en remplaçant le début par :

```
image=imread('image1.ascii.pgm');
```

Cette fonction va permettre d'utiliser l'image de notre choix.

Maintenant, nous affectons une compression à 50% :



En comparant l'image originale avec l'image compressée, nous n'observons pas de différence à l'œil nu.

On peut en conclure qu'une nouvelle fois que la compression d'une image sert à gagner en taille de stockage sans forcément perdre en qualité visuelle évidente.

Code de la partie 1 :

tp.m tp2.m

```
clear all
close all

image=zeros(512,512);
image(206:305,206:305)=255;

A = dct2(image);

B = A(:);

B= sort(abs(B),'descend');
C = length(B);

D = C * 0.2;
D = round (D);

E = B(D);
A(abs(A)<E)=0;
I = idct2(A);

figure
subplot(1,2,1)
imagesc(image)
colormap('gray')
axis equal
subplot(1,2,2)
imagesc(I)
colormap('gray')
axis equal
```

Code la partie 2 :

tp.m tp2.m

```
clear all
close all

image=imread('image1.ascii.pgm');

A = dct2(image);

B = A(:);

B= sort(abs(B),'descend');
C = length(B);

D = C * 0.5;
D = round (D);

E = B(D);
A(abs(A)<E)=0;
I = idct2(A);

figure
subplot(1,2,1)
imagesc(image)
colormap('gray')
axis equal
subplot(1,2,2)
imagesc(I)
colormap('gray')
axis equal

%montage({image,I})
```