Initiation au traitement d'images avec MATLAB

Table des matières

I. Introduction

L'objectif de cette introduction au traitement d'images sous Matlab est de présenter la notion d'image et d'effectuer des opérations simples d'analyse d'images telles que la détection de contours, le changement d'espace de couleur... Le traitement d'images est un thème de recherche situé entre l'informatique et le traitement du signal.

I.1. Rappels sur la notion d'image:

Une image réelle est obtenue à partir d'un signal continu bidimensionnel comme par exemple un appareil photo ou une caméra... Sur un ordinateur, on ne peut pas représenter de signaux continus, on travaille donc sur des valeurs discrètes.

Définition: Une image numérique est définie comme un signal fini bidimensionnel échantillonné à valeurs quantifiées dans un certain espace de couleurs. Elle est constituée de points (pixels).

Signal fini : une image possède des dimensions finies, exemple : 640x480, 800x600 points…

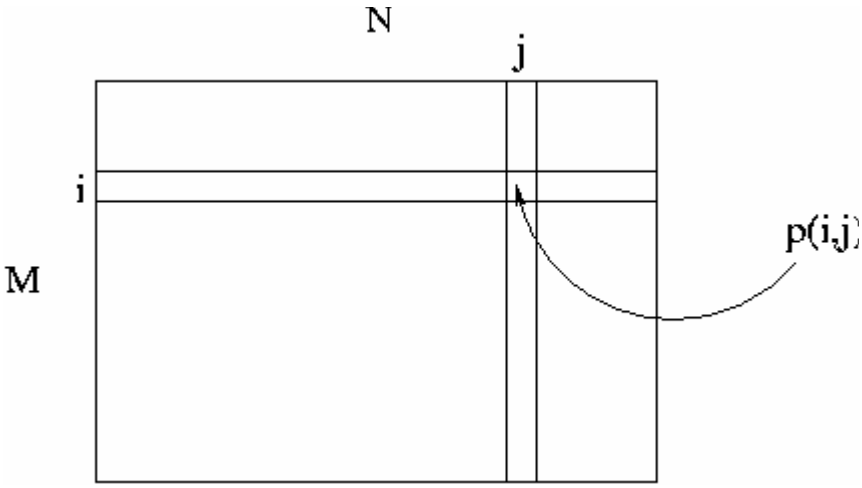
Signal bidimensionnel : une image possède deux dimensions : largeur, hauteur.

Signal échantillonné : les pixels d'une image sont régulièrement espacés sur une grille carrée.

Valeurs quantifiées : les valeurs des pixels appartiennent à un intervalle borné connu.

Espace de couleur : il existe de nombreuses façon de percevoir les couleurs d'une image, l'espace de représentation le plus connu est l'espace rgb (rouge-vert-bleu).

Autrement dit, une image est une matrice MxN de valeurs entières prises sur un intervalle borné [0,Ng] où Ng est la valeur maximale du niveau de gris.



p(i,j) est le niveau de gris du pixel de coordonnées ligne i et colonne j dans l'image. $p(i,j) \in [0, Ng]$. Les valeurs des niveaux de gris sont des entiers.

I.1.1. Image binaire

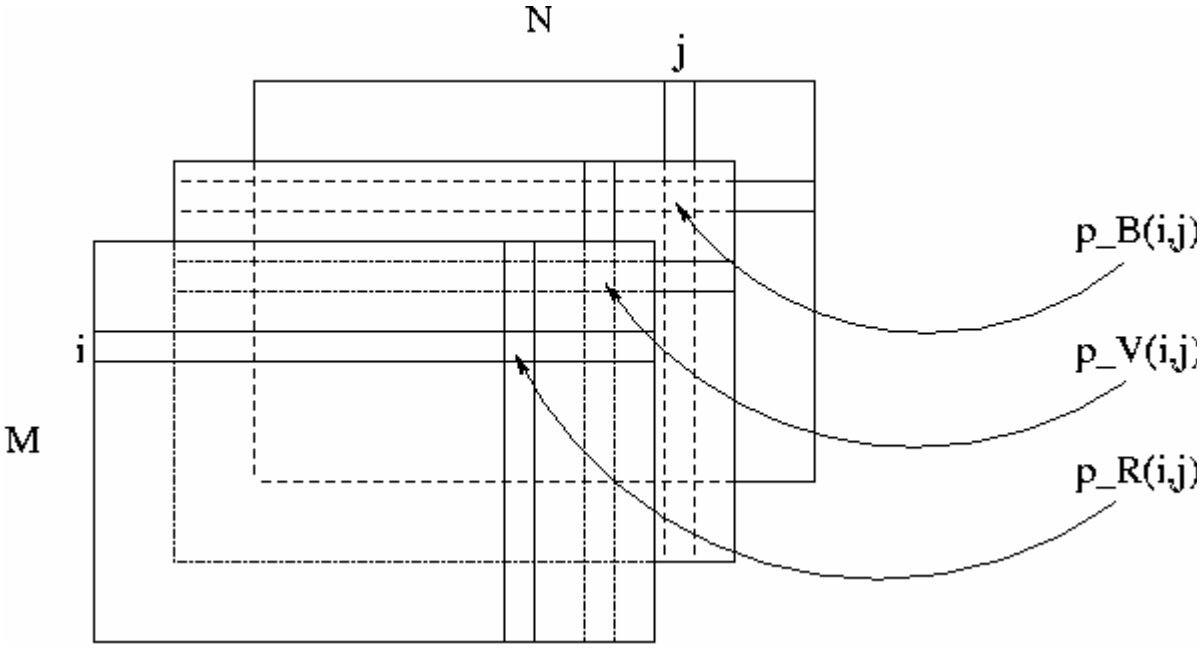
Une image binaire est une image MxN où chaque point peut prendre uniquement la valeur 0 ou 1. Les pixels sont noirs (0) ou blancs (1). Le niveau de gris est codé sur un bit (Binary digIT). Dans ce cas, on revient au cas donné en I.1. avec Ng = 2 et la relation sur les niveaux de gris devient: p(i,j) = 0 ou p(i,j) = 1.

I.1.2. Image en niveaux de gris

Une image ne niveaux de gris autorise un dégradé de gris entre le noir et le blanc. En général, on code le niveau de gris sur un octet (8 bits) soit 256 nuances de dégradé. L'expression de la valeur du niveau de gris avec Ng = 256 devient: $p(i,j) \in [0, 255]$.

I.1.3. Image couleur

Une image couleur est la composition de trois (ou plus) images en niveaux de gris sur trois (ou plus) composantes. On définit donc trois plans de niveaux de gris, un rouge, un vert et un bleu. La couleur finale est obtenue par synthèse additive des ces trois (ou plus) composantes.



On a les relations sur les niveaux de gris: p_R(i,j) $\in$[0, 255], p_V(i,j) $\in$[0, 255], p_B(i,j) $\in$[0, 255]. On voit bien sur la figure qu'une image couleur est en fait l'association de trois plans de niveau de gris, chacun d'eux étant une couleur de base.

I.1.4. Image à valeurs réelles

Pour certains calcul sur les images, le résultat peut ne pas être entier, il est donc préférable de définir l'image de départ et l'image résultat comme des images à valeurs réelles. En général, une image à valeurs réelle est telle que le niveau de gris est un réel compris entre 0.0 et 1.0. On a dans ce cas pour une image à niveaux de gris: p(i,j)$\in$[0.0, 1.0]. Pour une image couleur, la relation devient p_R(i,j)$\in$[0.0, 1.0], p_V(i,j)$\in$[0.0, 1.0], p_B(i,j)$\in$[0.0, 1.0].

I.2. Rappels sous Matlab :

Une image Matlab est une matrice bidimensionnelle de valeurs entières ou réelles. Les principales fonctions de traitement d'images sous Matlab se trouvent dans la boîte à outils (toolbox) image processing (traitement d'images). L'aide sur cette boîte à outils est obtenue en tapant ***help images*** en ligne de commande de Matlab. Ensuite, l'aide sur une commande particulière est obtenue en tapant help suivi du nom de la commande, par exemple ***help edge***.

» help images

Image Processing Toolbox.
Version 2.2 (R11) 05-Aug-1998

Release information.

Readme - Display information about versions 2.0, 2.1, and 2.2.

Image display.
colorbar - Display colorbar (MATLAB Toolbox).
getimage - Get image data from axes.
image - Create and display image object (MATLAB Toolbox).
imagesc - Scale data and display as image (MATLAB Toolbox).
immovie - Make movie from multiframe indexed image.
imshow - Display image.
montage - Display multiple image frames as rectangular montage.
subimage - Display multiple images in single figure.
truesize - Adjust display size of image.
warp - Display image as texture-mapped surface.
zoom - Zoom in and out of image or 2-D plot (MATLAB Toolbox).

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

Geometric operations.
imcrop - Crop image.
imresize - Resize image.
imrotate - Rotate image.
interp2 - 2-D data interpolation (MATLAB Toolbox).

Pixel values and statistics.
corr2 - Compute 2-D correlation coefficient.
imcontour - Create contour plot of image data.
imfeature - Compute feature measurements for image regions.
imhist - Display histogram of image data.
impixel - Determine pixel color values.
improfile - Compute pixel-value cross-sections along line segments.
mean2 - Compute mean of matrix elements.
pixval - Display information about image pixels.
std2 - Compute standard deviation of matrix elements.

Image analysis.
edge - Find edges in intensity image.
qtdecomp - Perform quadtree decomposition.
qtgetblk - Get block values in quadtree decomposition.
qtsetblk - Set block values in quadtree decomposition.

Image enhancement.
histeq - Enhance contrast using histogram equalization.
imadjust - Adjust image intensity values or colormap.
imnoise - Add noise to an image.
medfilt2 - Perform 2-D median filtering.
ordfilt2 - Perform 2-D order-statistic filtering.
wiener2 - Perform 2-D adaptive noise-removal filtering.

Linear filtering.
conv2 - Perform 2-D convolution (MATLAB Toolbox).
convmtx2 - Compute 2-D convolution matrix.
convn - Perform N-D convolution (MATLAB Toolbox).
filter2 - Perform 2-D linear filtering (MATLAB Toolbox).
fspecial - Create predefined filters.

Linear 2-D filter design.
freqspace - Determine 2-D frequency response spacing (MATLAB Toolbox).
freqz2 - Compute 2-D frequency response.

fsamp2 - Design 2-D FIR filter using frequency sampling.
ftrans2 - Design 2-D FIR filter using frequency transformation.
fwind1 - Design 2-D FIR filter using 1-D window method.
fwind2 - Design 2-D FIR filter using 2-D window method.

Image transforms.
dct2 - Compute 2-D discrete cosine transform.
dctmtx - Compute discrete cosine transform matrix.
fft2 - Compute 2-D fast Fourier transform (MATLAB Toolbox).
fftn - Compute N-D fast Fourier transform (MATLAB Toolbox).
fftshift - Reverse quadrants of output of FFT (MATLAB Toolbox).
idct2 - Compute 2-D inverse discrete cosine transform.
ifft2 - Compute 2-D inverse fast Fourier transform (MATLAB Toolbox).
ifftn - Compute N-D inverse fast Fourier transform (MATLAB Toolbox).
iradon - Compute inverse Radon transform.
phantom - Generate a head phantom image.
radon - Compute Radon transform.

Neighborhood and block processing.
bestblk - Choose block size for block processing.
blkproc - Implement distinct block processing for image.
col2im - Rearrange matrix columns into blocks.
colfilt - Perform neighborhood operations using columnwise functions.
im2col - Rearrange image blocks into columns.
nlfilter - Perform general sliding-neighborhood operations.

Binary image operations.
applylut - Perform neighborhood operations using lookup tables.
bwarea - Compute area of objects in binary image.
bweuler - Compute Euler number of binary image.
bwfill - Fill background regions in binary image.
bwlabel - Label connected components in binary image.
bwmorph - Perform morphological operations on binary image.
bwperim - Determine perimeter of objects in binary image.
bwselect - Select objects in binary image.
dilate - Perform dilation on binary image.
erode - Perform erosion on binary image.
makelut - Construct lookup table for use with applylut.

Region-based processing.
roicolor - Select region of interest, based on color.
roifill - Smoothly interpolate within arbitrary region.
roifilt2 - Filter a region of interest.
roipoly - Select polygonal region of interest.

Colormap manipulation.
brighten - Brighten or darken colormap (MATLAB Toolbox).
cmpermute - Rearrange colors in colormap.
cmunique - Find unique colormap colors and corresponding image.
colormap - Set or get color lookup table (MATLAB Toolbox).
imapprox - Approximate indexed image by one with fewer colors.
rgbplot - Plot RGB colormap components (MATLAB Toolbox).

Color space conversions.
hsv2rgb - Convert HSV values to RGB color space (MATLAB Toolbox).
ntsc2rgb - Convert NTSC values to RGB color space.
rgb2hsv - Convert RGB values to HSV color space (MATLAB Toolbox).
rgb2ntsc - Convert RGB values to NTSC color space.
rgb2ycbcr - Convert RGB values to YCBCR color space.
ycbcr2rgb - Convert YCBCR values to RGB color space.

```
Image types and type conversions.
  dither      - Convert image using dithering.
  gray2ind    - Convert intensity image to indexed image.
  grayslice   - Create indexed image from intensity image by thresholding.
  im2bw       - Convert image to binary image by thresholding.
  im2double   - Convert image array to double precision.
  im2uint8    - Convert image array to 8-bit unsigned integers.
  im2uint16   - Convert image array to 16-bit unsigned integers.
  ind2gray    - Convert indexed image to intensity image.
  ind2rgb     - Convert indexed image to RGB image (MATLAB Toolbox).
  isbw        - Return true for binary image.
  isgray      - Return true for intensity image.
  isind       - Return true for indexed image.
  isrgb       - Return true for RGB image.
  mat2gray    - Convert matrix to intensity image.
  rgb2gray    - Convert RGB image or colormap to grayscale.
  rgb2ind     - Convert RGB image to indexed image.

Toolbox preferences.
  iptgetpref - Get value of Image Processing Toolbox preference.
  iptsetpref - Set value of Image Processing Toolbox preference.

Demos.
  dctdemo     - 2-D DCT image compression demo.
  edgedemo    - Edge detection demo.
  firdemo     - 2-D FIR filtering and filter design demo.
  imadjdemo   - Intensity adjustment and histogram equalization demo.
  nrfiltdemo - Noise reduction filtering demo.
  qtdemo      - Quadtree decomposition demo.
  roidemo     - Region-of-interest processing demo.

Slide shows.
  ipss001     - Region labeling of steel grains.
  ipss002     - Feature-based logic.
  ipss003     - Correction of nonuniform illumination.

------------------------------
```

Comme on peut le constater, la boîte à outils images de Matlab contient de nombreuses fonctions qui permettent le développement facile et rapide d'algorithmes en fonction du problème à traiter. C'est un très bon outil pour la validation de méthodes de traitement d'images appliquées à un problème particulier.

II. Lire et écrire des images sous Matlab

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. L'exemple ci-dessous permet de lire une image au format TIFF, de la décoder dans la variable img et de l'afficher à l'écran dans une figure. La commande ***axis('image')*** rend l'image affichée carrée pour garder les proportions. L'appel à ***axis on*** permet l'affichage des graduations des axes. Enfin, ***colorbar*** affiche la barre des couleurs de l'image.

```
» img=imread('saturn.tif');
» figure;imshow(img);
» axis('image');
» axis on
» colorbar
```

L'accès à une élément particulier d'une image est indexé par le nom et la position de cet élément. Par exemple, si on conserve l'image img ci-dessus, on peut récupérer les valeurs ou les modifier aisément.

```
» img(3,2)
» img(1:10,30:40)
» img(1:3,31:39) = 0;
» figure;imshow(img);
```

Matlab offre une possibilité intéressante, en effet, il est possible d'afficher plusieurs images dans la même figure. Pour ce faire, il faut utiliser la commande ***subplot***. Elle s'utilise avec comme arguments le nombre de ligne, le nombre de colonnes et le numéro de l'image dans la figure. Dans l'exemple ci-dessous on souhaite afficher deux images sur la même ligne dans une seule figure.

```
» img=imread('blood1.tif');
» img2=imread('alumgrns.tif');
» figure;subplot(1,2,1);imshow(img);
» subplot(1,2,2);imshow(img2);
```

Afin de fermer une figure sous Matlab, on tape ***close*** s'il s'agit de la dernière figure ouverte ou bien ***close*** avec en paramètre le numéro de figure pour fermer la figure donnée en paramètres. Pour fermer toutes les figures, on demande la fermeture avec le mot-clé ***all***.

```
» img=imread('rice.tif');
» img2=imread('alumgrns.tif');
» figure;imshow(img);
» figure;imshow(img2);
» close(1)
» close all
```

Pour sélectionner une figure, on demande l'affichage de celle-ci à l'aide de la commande ***figure*** qui prend en paramètre le numéro de la figure. Un exemple illustre ce cas ci-dessous.

```
» figure(1)
```

Matlab autorise l'exportation d'images sous divers formats: BMP, TIFF, EPS, PS... La commande qui permet da sauvegarder une figure est ***print -dFORMAT fichier***. Un exemple est donné ci-dessous. Dans cet exemple, on affiche une image dans une figure et grâce à la commande ***print***, on exporte le résultat dans le format JPEG avec pour nom de fichier result.jpg.

```
» img=imread('rice.tif');
» figure;imshow(img);
» print -djpeg result.jpg
```

Les valeurs des images lues sous Matlab sont entières, mais dans certaines circonstances, on a besoin de travailler sur des valeurs réelles. La transformation pour passer d'entier à réel utilise la fonction ***im2double***.

```
» img=imread('rice.tif');
» figure;imshow(img);
» imgdbl=im2double(img);
» figure;imshow(imgdbl);
» imgint=im2uint8(imgdbl);
» figure;imshow(imgint);
» imwrite(imgint,'test.jpg','jpeg');
» whos
```

III. Opérations géométriques

Les opérations géométriques classiques sont permises avec la boîte à outils de traitement d'images: rotation, changement de taille, découpage...

L'exemple ci-dessous illustre la rotation d'une image avec Matlab. Dans le premier cas, imgrot1 est plus grande que img. Dans le second cas, le paramètre

'crop' impose un découpage de l'image et la taille de l'image imgrot2 est la même que celle de l'image img.

```
» img=imread('bacteria.tif');
» figure;imshow(img);
» imgrot1=imrotate(img,3,'bilinear');
» figure;imshow(imgrot1);
» imgrot2=imrotate(img,3,'bilinear','crop');
» figure;imshow(imgrot2);
```

Le zoom permet d'agrandir une partie de l'image à l'aide de la souris, pour cela, il faut entrer la commande dans l'éditeur de Matlab puis sélectionner une zone de l'image à agrandir. Le bouton droit de la souris permet de revenir à la taille normale.

```
» zoom
```

Un autre outil géométrique très utile pour détecter les niveaux de gris dans une image est la fonction de profil. On exécute cette fonction, on choisit à l'aide de la souris une ligne de l'image et on obtient le profil du niveau de gris le long de cette ligne.

```
» img=imread('bacteria.tif');
» figure;imshow(img);
» improfile
```

IV. Détection de contours

La détection de contours permet de repérer les différents objets qui constituent la scène de l'image. Il existe de nombreuses méthodes pour trouver les contours des objets, la plupart sont basées sur les dérivées premières et secondes de l'image.

```
» img = imread('rice.tif');
» cont1 = edge(img,'prewitt');
» cont2 = edge(img,'canny');
» figure; imshow(img);
» figure; imshow(cont1);
```

```
» figure; imshow(cont2);
```

La détection de contours permet de repérer dans les images les objets qui s'y trouvent avant d'appliquer le traitement uniquement sur ces objets. Pour mieux comprendre la notion de contour, il est possible de visualiser une image en 3D.

```
» img = imread('rice.tif');
» img=im2double(img);
» figure;mesh(img);
```

V. Histogramme - seuillage

L'histogramme d'une image donne la répartition de ses niveaux de gris. Ainsi pour une image qui possède 256 niveaux de gris, l'histogramme représente le niveau de gris en fonction du nombre de pixels à ce niveau de gris dans l'image.

```
» img = imread('rice.tif');
» histo = imhist(img,256);
» figure;plot(histo);
```

On sait que les niveaux de gris à zéro correspondent au noir et que les niveaux de gris à 1 indiquent le blanc. L'histogramme donne donc une excellente idée de la séparation entre quelque chose qui est clair et quelque chose qui est foncé dans l'image. Typiquement, une utilisation de ce fait est le seuillage d'une image, ce terme désigne la définition d'un seuil au-dessus ou en-dessous duquel on va garder certaines valeurs de niveaux de gris.

```
» img=imread('saturn.tif');
» figure;imshow(img);
» img=im2double(img);
» figure;subplot(1,2,1);imshow(img);
» result=(img2>0.5).*img;
» subplot(1,2,2);imshow(result);
```

VI. Espace de couleurs

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.

```
» img=imread('blood1.tif');
» figure;imshow(img)
» colorbar
» colormap(hot)
» colormap(hsv)
» colormap(gray)
» colormap(bone)
» colormap(copper)
» colormap(pink)
» colormap(white)
...
» help graph3d
```

La boîte à outils images de Matlab gère les espaces de couleur RGB, HSV, YCbCr, NTSC. Un exemple ci-dessous permet de se rendre compte de l'utilisation d'un changement d'espace de couleur. On lit une image colorée, on la passe dans l'espace HSV et on va changer la saturation avant de repasser dans l'espace RGB pour visualiser le résultat.

```
» img=imread('flowers.tif');
» figure;imshow(img);
» imghsv=rgb2hsv(img);
» figure;imshow(imghsv);
» imghsv(100:200,300:400,1)=0.7;
» imgrgb=hsv2rgb(imghsv);
» figure;imshow(imgrgb);
» imghsv=rgb2hsv(img);
» figure;imshow(imghsv);
» imghsv(100:200,300:400,2)=0;
» imgrgb=hsv2rgb(imghsv);
» figure;imshow(imgrgb);
» imghsv=rgb2hsv(img);
» figure;imshow(imghsv);
» imghsv(100:200,300:400,3)=0.3;
» imgrgb=hsv2rgb(imghsv);
```

```
» figure;imshow(imgrgb);
```

Une fonction utile à connaître est la fonction de passage d'une image couleur à une image en niveaux de gris. Cette fonction ***rgb2gray*** permet d'obtenir une image en niveaux de gris en partant d'une image couleur. Dans certains cas, cette transformation est très utile.

```
» img=imread('blood1.tif');
» figure;imshow(img)
» nivgris=rgb2gray(img);figure;imshow(nivgris);
```

VII. Transformée de Fourier

La transformée de Fourier est un outil mathématique de traitement du signal qui permet de passer d'une représentation temporelle à une représentation fréquentielle du signal. Cette théorie est basée sur le fait que toute fonction périodique est décomposable sur une base de sinus et de cosinus. Ainsi, on peut passer d'une représentation temporelle du signal (dans le repère temporel classique) à une représentation en fréquence sur une base de sinus et de cosinus (dans le repère fréquentiel). La puissance de cet outil réside dans le fait que cette transformée est réversible et qu'elle peut être étendue aux signaux non périodiques (qu'on considère alors comme de période infinie).

L'exemple ci-dessous transforme une image dans l'espace de Fourier, fait la transformation inverse pour voir si tout s'est bien passé (la transformée de Fourier est réversible). Ensuite, le programme modifie les valeurs dans l'espace de Fourier et effectue la reconstruction avec cette fois une nette différence.

```
» close all
» img=imread('blood1.tif');
» img=im2double(img);
» figure;subplot(1,3,1);imshow(img)
» fourier=fft2(img);
» subplot(1,3,2);imshow(real(fourier));
» subplot(1,3,3);imshow(imag(fourier));
» retour=ifft2(fourier);
» figure;imshow(real(retour));
» fourier(1:200,1:250) = 0;
» retour=ifft2(fourier);
» figure;imshow(real(retour));
» imgresult=abs(retour-img);
» figure;mesh(imgresult);
```

VIII. Conclusion

Le traitement d'images permet de modifier le contenu des images afin de tirer l'information utile pour une application particulière. Matlab offre de nombreuses possibilités de traitement avec une palette très fournie d'outils prêts à l'emploi. L'inconvénient majeur de Matlab réside dans sa relative lenteur pour effectuer certaines opérations de calculs (par exemple la transformée de Fourier). Toutefois Matlab permet de déployer rapidement des tests pour vérifier la validité d'une méthode de traitement d'images. La manipulation d'images revient à la manipulation de matrices qui est très facile grâce au langage de haut niveau de Matlab.