

6 Traitement d'image

Ces exercices nécessitent la "Image Progressing Toolbox"

Une bonne documentation se trouve dans le `images_tb.pdf` édité par la société "The Matworks Inc."

6.1 Ouverture, affichage, types d'images

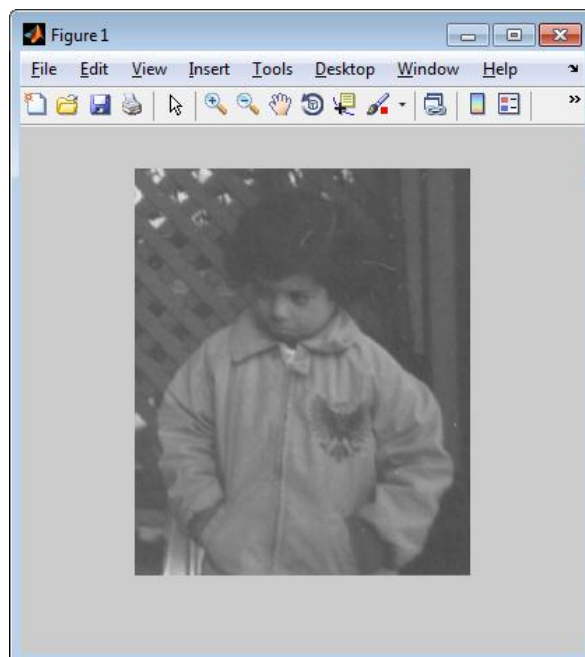
1. Ouvrir l'image 'pout.tif' au moyen de l'instruction : `>>I=imread('pout.tif')`
Il n'est pas nécessaire de fournir cette image car elle se trouve dans le répertoire `~/toolbox/images/indemos` qui se trouve dans le PATH de Matlab.
L'instruction `>>whos` donnera le nom, la taille et le type de l'image.
On constate que la valeur de l'image est `291x240 uint8`. Ceci donne la taille de l'image en pixels et indique que celle-ci possède 256 niveaux de gris.

```
>> I = imread('pout.tif');  
>> whos I
```

Name	Size	Bytes	Class	Attributes
I	291x240	69840	uint8	

Uint8 => pixel codé sur 8 bits

2. Visualiser cette image par l'instruction :
`>>imshow(I)`

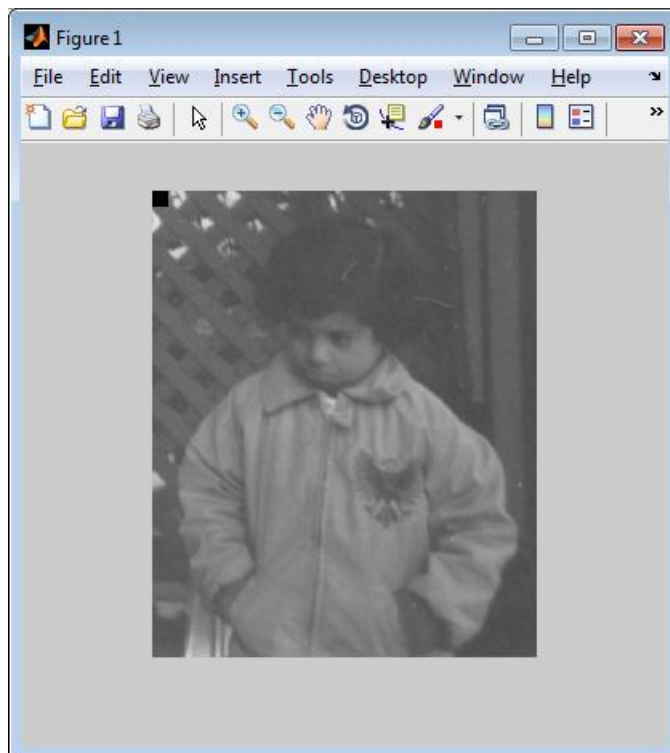


3. Placer un carré noir de dimension 10x10 dans le coin supérieur gauche de l'image

```
clc;  
close all;  
clear all;  
  
I = imread('pout.tif');  
  
I(1:10,1:10,1)=0;  
  
imshow(I);
```

4. Sauver cette nouvelle image par l'instruction `imwrite` et visualiser celle-ci

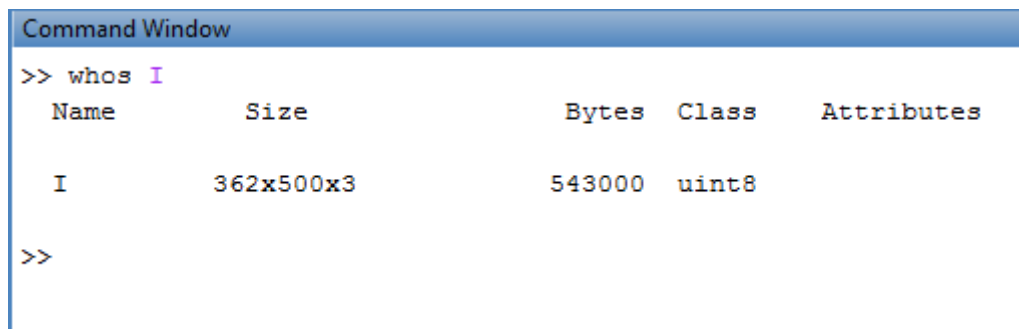
```
clc;  
close all;  
clear all;  
  
I = imread('pout.tif');  
  
I(1:10,1:10,1)=0;  
  
imshow(I);  
  
imwrite(I, 'carreNoir.jpg', 'jpg', 'Quality', 100);
```



5. Ouvrir l'image 'flowers.tif'

(a) Quelle est-le type et la taille de l'image.

```
clc;  
close all;  
clear all;  
  
I = imread('docs_EPHEC\Image\flowers.tif');  
  
imshow(I);
```

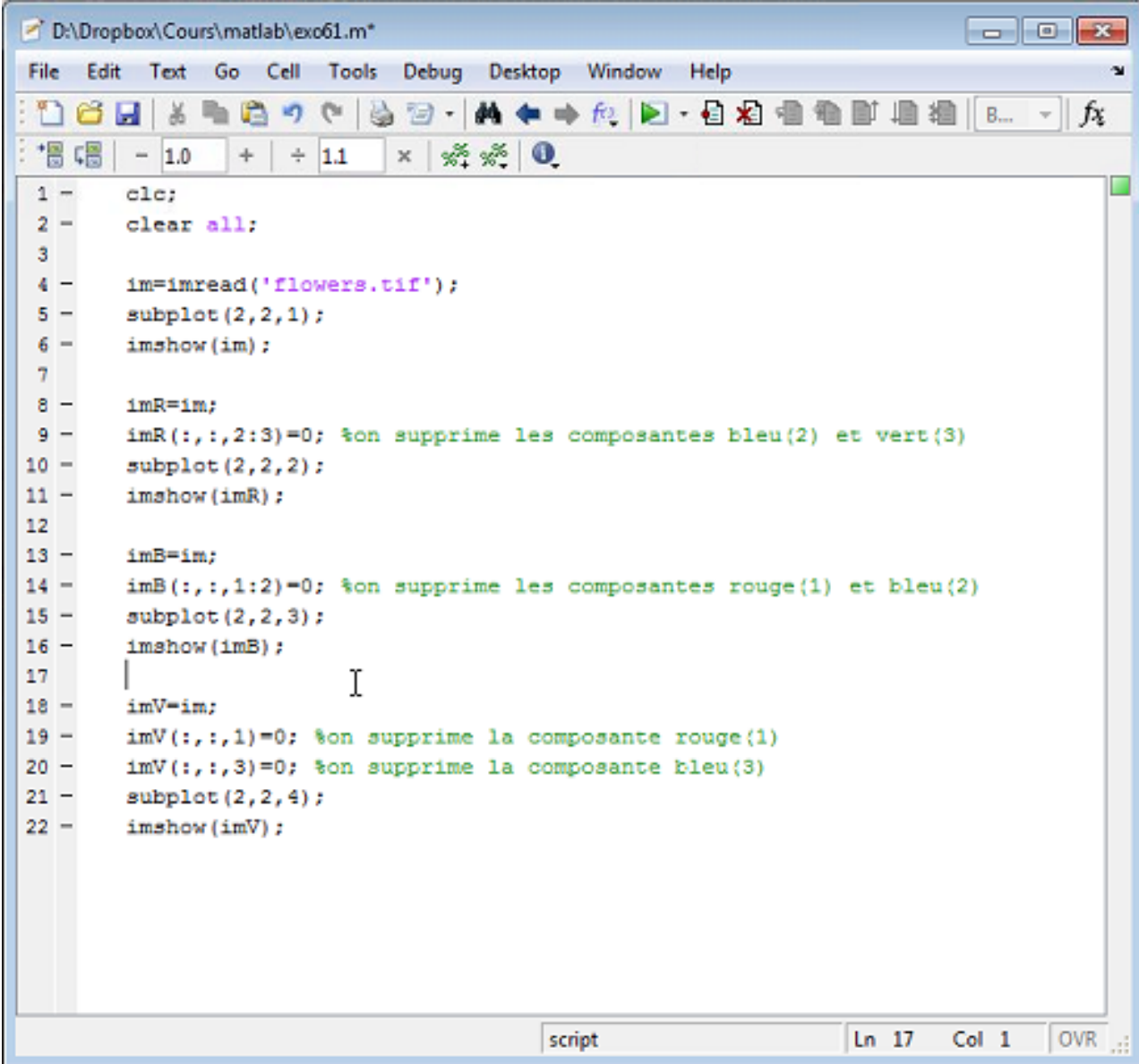


```
Command Window  
  
>> whos I  
      Name      Size      Bytes  Class  Attributes  
      I         362x500x3    543000  uint8  
  
>>
```

Image de 362px de haut, 500 de large et que c'est une matrice à trois dimensions donc une image couleur (RGB). Sa classe, c'est uint8 donc une image où les pixels sont codés sur 8bits.

Son type est donc RGB ==> **362x500x3** (sa dimension)

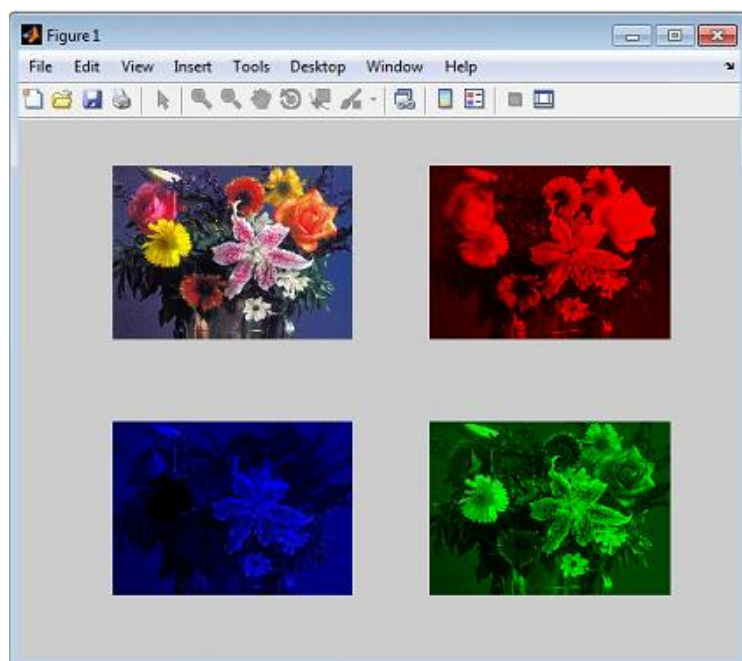
- (b) Visualiser sur 4 figures différentes l'image, sa composante rouge, sa composante verte et sa composante bleue. Est-ce que ceci vous semble cohérent ?



```
1 - clc;
2 - clear all;
3
4 - im=imread('flowers.tif');
5 - subplot(2,2,1);
6 - imshow(im);
7
8 - imR=im;
9 - imR(:,:,2:3)=0; %on supprime les composantes bleu(2) et vert(3)
10 - subplot(2,2,2);
11 - imshow(imR);
12
13 - imB=im;
14 - imB(:,:,1:2)=0; %on supprime les composantes rouge(1) et bleu(2)
15 - subplot(2,2,3);
16 - imshow(imB);
17
18 - imV=im;
19 - imV(:,:,1)=0; %on supprime la composante rouge(1)
20 - imV(:,:,3)=0; %on supprime la composante bleu(3)
21 - subplot(2,2,4);
22 - imshow(imV);
```

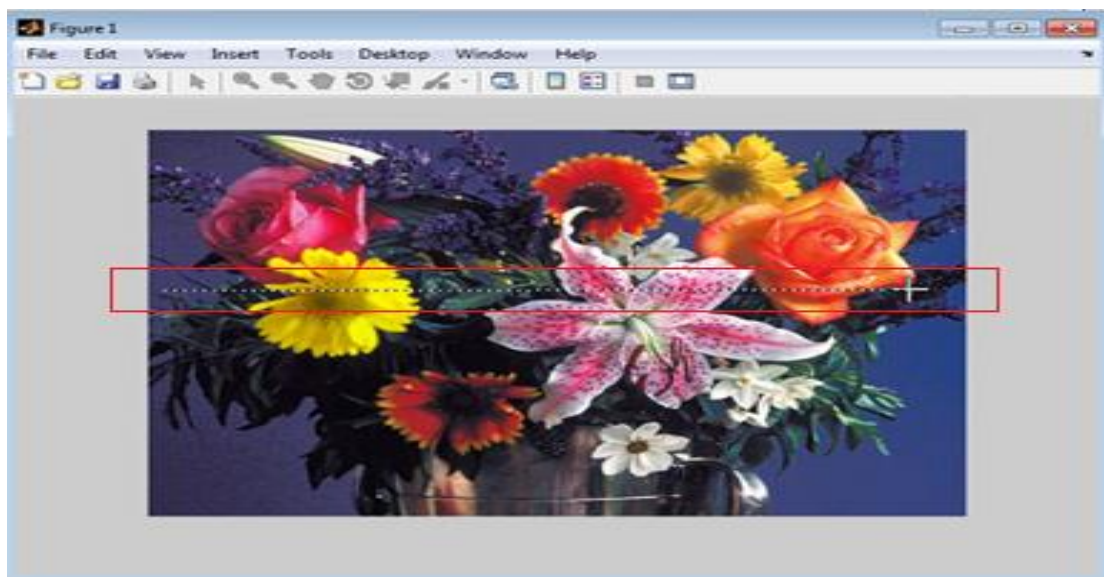
Dans le cas du vert on ne peut pas indiquer **1:3** pour la suppression (sinon on supprimerait les 3 couleurs). **Donc on les supprime séparément :**

```
imV(:,:,1)=0; %on supprime la composante rouge(1)
imV(:,:,3)=0; %on supprime la composante bleu(3)
```



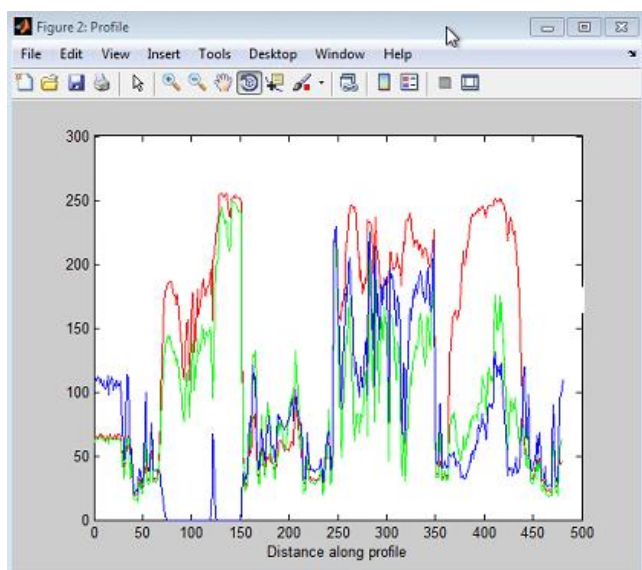
- (c) Utiliser l'instruction 'improfile' pour visualiser les composantes R, G et B d'une ligne sur l'image.

```
1 -   clc;
2 -   clear all;
3
4 -   im=imread('flowers.tif');
5 -   imshow(im)
6 -   improfile
```



Lorsque l'image s'affiche **improfile** permet de sélectionner une ligne avec la souris. Et d'ensuite afficher un graphique avec le niveau de chacune des 3 couleurs sur 256 bits.

On peut constater que le **rouge** est intense à la fin du graphe car la **fleur est orange** (+- RGB : 255, 122, 122). On constate également que la fleur jaune est un mélange de **vert** et **rouge**.



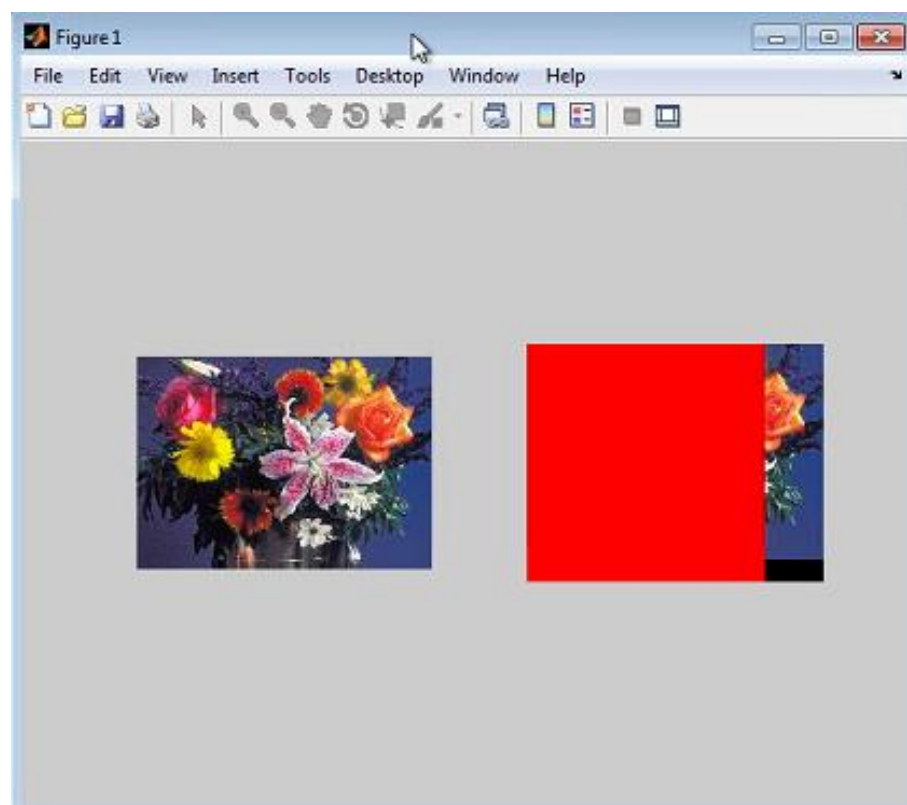
(d) i. Créer un carré 400x400 rouge d'intensité maximale

```
clc;
clear all;

im=imread('flowers.tif');
subplot(1,2,1); % une ligne, 2 colonnes, position 1
imshow(im);

im(1:400,1:400,2:3)=0; % supprime le bleu et le vert
im(1:400,1:400,1)=255; % met du rouge d'intensité max

subplot(1,2,2);
imshow(im);
improfile
```



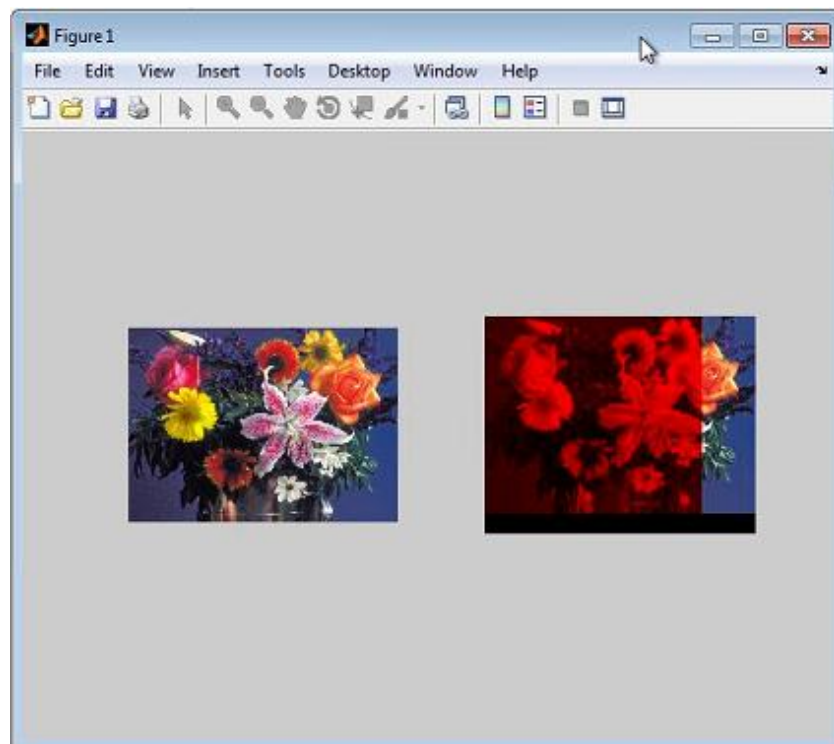
Maintenant même chose mais en gardant seulement l'intensité de rouge qu'il y avait à la base sur l'image. Les zones les plus noires sont celles où le rouge est le moins présent.

```
clc;
clear all;

im=imread('flowers.tif');
subplot(1,2,1); % une ligne, 2 colones, position 1
imshow(im);

im(1:400,1:400,2:3)=0; % supprime le bleu et le vert
%im(1:400,1:400,1)=255; % met du rouge d'intensité max

subplot(1,2,2);
imshow(im);
improfile
```



ii. Créer un carré 400x400 vert d'intensité maximale

```
clc;
clear all;

im=imread('flowers.tif');
subplot(1,2,1);
imshow(im);

im(1:400,1:400,1)=0;
im(1:400,1:400,3)=0;
im(1:400,1:400,2)=255;

subplot(1,2,2);
imshow(im);
improfile
```



iii. Créer un carré 400x400 bleu d'intensité maximale

```
clc;
clear all;

im=imread('flowers.tif');
subplot(1,2,1);
imshow(im);

im(1:400,1:400,1)=0;
im(1:400,1:400,2)=0;
im(1:400,1:400,3)=255;

subplot(1,2,2);
imshow(im);
improfile
```



iv. Créer un carré 400x400 rouge+vert d'intensité maximale

```
clc;
clear all;

im=imread('flowers.tif');
subplot(1,2,1);
imshow(im);

im(1:400,1:400,1)=255;
im(1:400,1:400,2)=255;
im(1:400,1:400,3)=0;

subplot(1,2,2);
imshow(im);
improfile
```



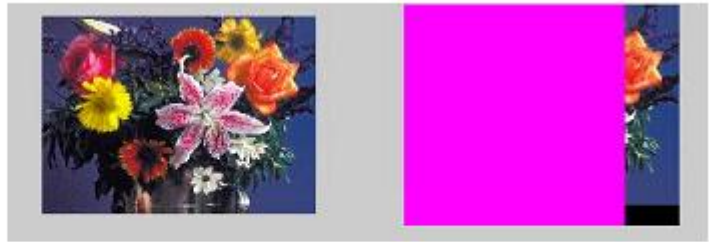
v. Créer un carré 400x400 rouge+bleu d'intensité maximale

```
clc;
clear all;

im=imread('flowers.tif');
subplot(1,2,1);
imshow(im);

im(1:400,1:400,1)=255;
im(1:400,1:400,2)=0;
im(1:400,1:400,3)=255;

subplot(1,2,2);
imshow(im);
improfile
```



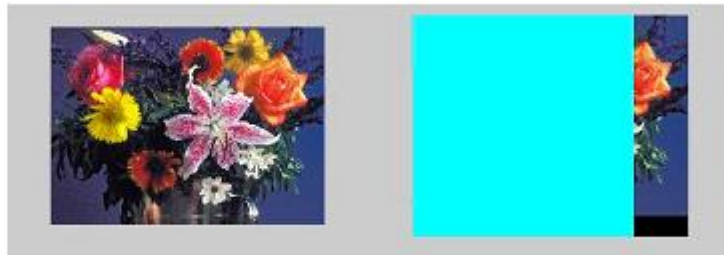
vi. Créer un carré 400x400 vert+bleu d'intensité maximale

```
clc;
clear all;

im=imread('flowers.tif');
subplot(1,2,1);
imshow(im);

im(1:400,1:400,1)=0;
im(1:400,1:400,2)=255;
im(1:400,1:400,3)=255;

subplot(1,2,2);
imshow(im);
improfile
```



vii. Créer un carré 400x400 rouge+vert+bleu d'intensité maximale

```
clc;
clear all;

im=imread('flowers.tif');
subplot(1,2,1);
imshow(im);

im(1:400,1:400,1)=255;
im(1:400,1:400,2)=255;
im(1:400,1:400,3)=255;

subplot(1,2,2);
imshow(im);
improfile
```



viii. Créer un carré 400x400 dont la couleur est égale au fond bleu de trouvant dans le coin inférieur droit de l'image 'flowers.tif'.

```
clc;
clear all;

im=imread('flowers.tif');
subplot(1,2,1);
imshow(im);

im2=im;
im2(1:400,1:400,1)=im(end,end,1)
im2(1:400,1:400,2)=im(end,end,2)
im2(1:400,1:400,3)=im(end,end,3)

subplot(1,2,2);
imshow(im2);
```



Ici on prend le dernier pixel en bas à droite de l'image, on extrait sa composante rouge et on l'insère dans le carré à son niveau de rouge, on en fait de même pour sa composante verte et bleu.

(e) Classe d'une image

i. Constater que l'image 'flowers.tif' est de classe uint8

```
>> I = imread('docs_EPHEC\Image\flowers.tif');
>> whos I
```

Name	Size	Bytes	Class	Attributes
I	362x500x3	543000	uint8	

ii. Transformer cette image en classe double (instruction >>double)

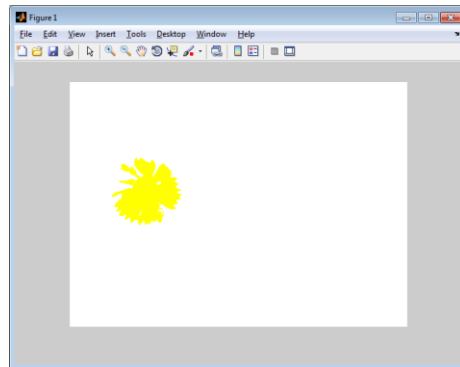
```
>> I = double(I);
>> whos I
```

Name	Size	Bytes	Class	Attributes
I	362x500x3	4344000	double	

- iii. Visualiser la nouvelle image et constater qu'elle est toute blanche.
En effet, en classe double, les niveaux de couleurs sont compris entre 0 et 1.

Comment afficher alors la nouvelle image en classe double ?

Donc un `imshow(I)` ;



Ca donne rien.

```
clc;
clear all;

im=imread('flowers.tif');
subplot(1,2,1);
imshow(im);

im2=double(im);
im2=im2/255

subplot(1,2,2);
imshow(im2);
%improfile
```



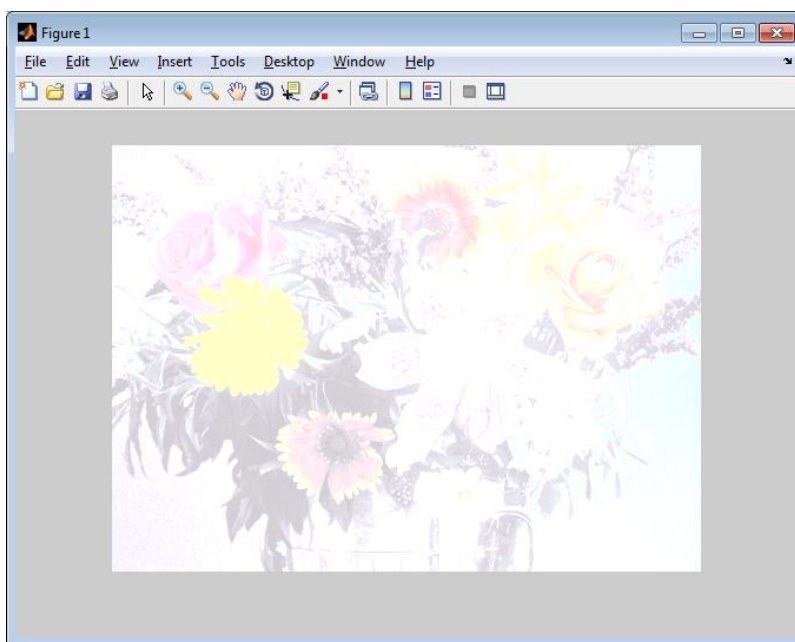
Chaque composante couleur d'un pixel d'une image en double à une valeur entre 0 et 1. Vu que l'image de base est codée sur 8 bits, on va diviser la valeur de chaque pixel par 255. Donc si le pixel était blanc en uint8 sa valeur était de 255, en le divisant par 255 on obtient 1 ==> la couleur du blanc en double...

6.2 Opérations arithmétiques sur une image - coordonnées

Les opérations arithmétiques ne peuvent être effectuées que sur des images de type "double"

1. Ajouter la valeur 200 à l'image 'flowers.tif'. Constater la saturation de l'image produite. Constater l'erreur obtenue.

```
>> I = imread('flowers.tif');  
>> I = I +200;  
>> imshow(I)  
>>
```



En ajoutant 200 à l'image on sature chaque une des 3 teintes de couleur de chaque pixel et étant donné que la valeur max d'une teinte est 255 qui correspond au blanc, la majeure partie de l'image est blanche.

2. Ouvrir l'image "pout.tif", la transformer en type double et diviser chaque valeur par 255.

Transformer cette image en utilisant l'instruction `im2double`.

Vérifier si les deux images sont égales (suggestion, chercher le min et le max de la différence ou chercher toutes les valeurs de cette différence au moyen de l'instruction `unique`).

Donc comme exercice précédent on transforme `flowers.tif` en double et on divise par 255 car double est entre 0 et 1 et que `flowers.tif` était un `uint8` à la base.

```
1 - clear all; close all; clc;
2 - I=imread('pout.tif');
3 - I2=double(I)/255;
4 - I3=im2double(I);
5 - subplot(1,3,1);imshow(I);
6 - subplot(1,3,2);imshow(I2);
7 - subplot(1,3,3);imshow(I3);
```



`Im2double` divise automatiquement l'image par le nombre correspondant à sa classe. Ici il voit que c'est une image de classe `uint8` et donc sait qu'il faut le diviser par 255.

3. Afficher l'image 'flowers.tif' en niveau de gris.

(suggestion : la luminance vaut $0.299 R + 0.587 G + 0.114 B$)

Comparer votre résultat avec celui obtenu au moyen de l'instruction `>>rgb2gray`

```
1 - clear all; close all; clc;
2 - I=imread('flowers.tif');
3 - I=im2double(I);
4 - I2(:, :)=0.299*I(:, :,1)+0.587*I(:, :,2)+0.114*I(:, :,3);
5 - I3=rgb2gray(I);
6
7 - subplot(1,3,1);imshow(I);
8 - subplot(1,3,2);imshow(I2);
9 - subplot(1,3,3);imshow(I3);
```



Les luminances données dans l'énoncé sont donc les conventions pour transformer un plan d'image en son niveau de gris. On remarque donc que le vert (0.587) est plus clair que le rouge (0.299) qui lui est plus clair que le bleu (0.114).

`Rgb2gray()` revient au même que de multiplier chaque plan d'image par sa luminance.

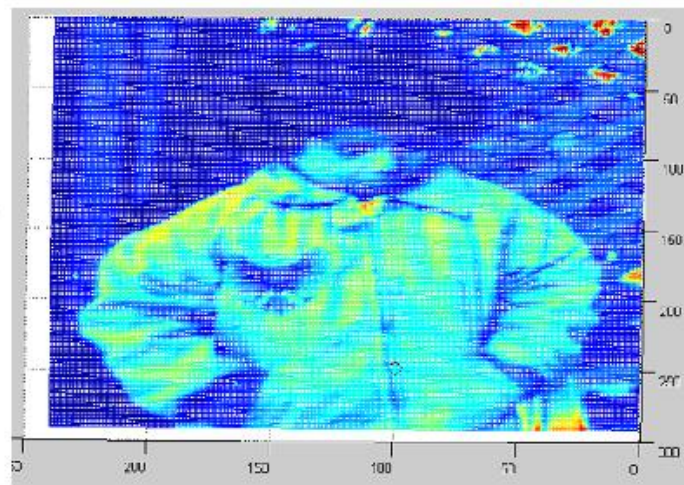
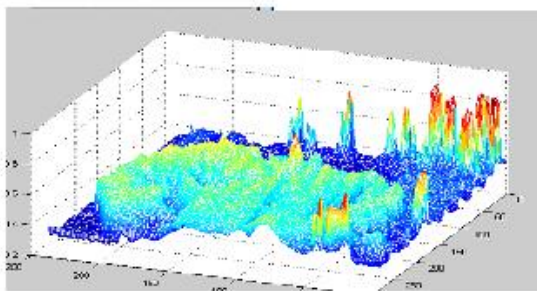
4. Visualiser l'image 'pout.tif' en 3D (pixels x,y et niveau de gris) des fonctions 'mesh(x,y,z)' et 'meshgrid'.

L'image possède 291 lignes et 240 colonnes.

Utiliser les instructions :

```
im=imread('pout.tif');  
im=double(im)/255;  
[X,Y]=meshgrid(1:240,1:291);  
mesh(im)
```

```
clear all; close all; clc;  
I=imread('pout.tif');  
I=im2double(I);  
[x,y]=meshgrid(1:240,1:291);  
mesh(I);
```



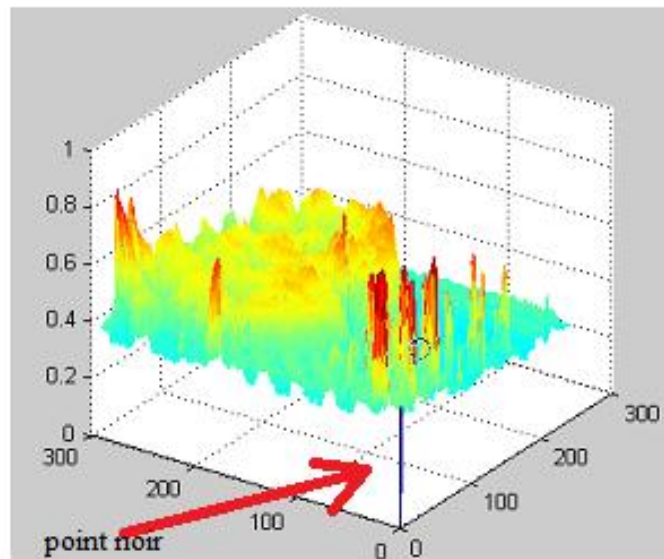
On peut donc visualiser les niveaux de gris de l'image sous plusieurs vu (celle de droite du dessus, celle à gauche de côté).

Les couleurs représentent l'intensité du niveau de gris PAR RAPPORT A L'IMAGE (voir exemple suivant) :

- bleu : très faible
- vert : faible
- jaune : moyen
- rouge : haut

5. Dessiner un point noir dans le coin supérieur gauche de l'image

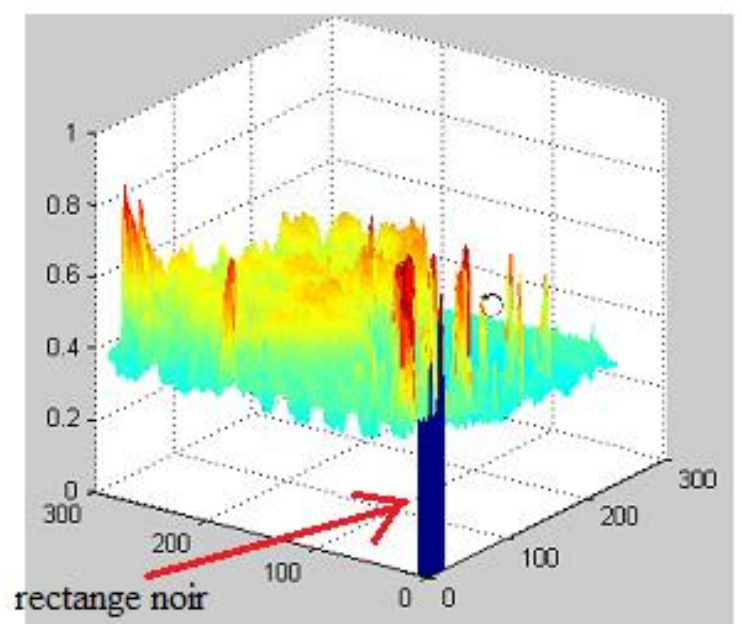
```
clear all; close all; clc;  
I=imread('pout.tif');  
I=im2double(I);  
I(1,1)=0;  
imshow(I);  
mesh(I);  
colormap(jet)
```



Colormap est parfois là par défaut pour indiquer les couleurs des différents niveaux de gris de l'image, il vaut mieux l'indiquer...

6. Dessiner un rectangle noir de dimension 10x20 dans le coin supérieur gauche de l'image

```
clear all; close all; clc;  
I=imread('pout.tif');  
I=im2double(I);  
I(1:10,1:20)=0;  
imshow(I);  
mesh(I);  
colormap(jet)
```



7. Dessiner en noir chaque pixel de l'image précédente au moyen des 3 programmes suivants. Les instructions tic toc permettent de visualiser le temps d'exécution d'un programme.

```
clear all; close all; clc;

im=imread('pout.tif');
im=im2double(im);
tic
im(:,:)=0;
toc

im=imread('pout.tif');
im=im2double(im);
tic
for ii=1:291
    im(ii,:)=0;
end
toc

im=imread('pout.tif');
im=im2double(im);
tic
for ii=1:291
    for jj=1:240
        im(ii,jj)=0;
    end
end
toc
```

Command Window

```
Elapsed time is 0.000614 seconds.
Elapsed time is 0.000399 seconds.
Elapsed time is 0.001175 seconds.
fx >>
```

Workspace

Name	Value	Min	Max
ii	291	291	291
im	<291x240 double>	0	0
jj	240	240	240

Les vecteurs sont donc plus performant que les boucles d'autant plus lorsqu'on précise sa valeur de départ et de fin (ex : 1:10) au lieu de laisser matlab retrouver la valeur de départ et de fin tout seul (:). Ce n'est en fait pas un très bon exemple dans le cahier, il aurait été plus intéressant de montrer l'interet comme ca :

```

clc;
close all;
clear all;

%moins rapide
im = imread('pout.tif');
im=im2double(im);

tic
for ii=1:291
    for jj=1:240
        im(ii,jj)=0;
    end;
end;
toc

%plus rapide
im = imread('pout.tif');
im=im2double(im);

tic
    im(:,:)=0;
toc

%encore plus rapide
im = imread('pout.tif');
im=im2double(im);

tic
    im(1:291,1:240)=0;
toc

```

```

Command Window
Elapsed time is 0.012377 seconds.
Elapsed time is 0.001001 seconds.
Elapsed time is 0.000250 seconds.
>>

```

6.3 Traitement d'images-filtrage

6.3.1 Traitement d'une image

L'image diapol.jpg provient d'une ancienne diapositive. Elle est trop rouge. Corrigez-la.

```

clear all; close all; clc;
im=imread('diapol.jpg');
subplot(1,2,1)
imshow(im);
im(:,:1)=im(:,:1)*0.9
im(:,:2)=im(:,:2)*1.5
im(:,:3)=im(:,:3)*1.2
subplot(1,2,2)
imshow(im);

```



On diminue donc le rouge de 10% en gardant seulement 90%, on ajoute 50% de vert et 20% de bleu car l'image est trop sombre en retirant seulement du rouge.

Faire de même avec diapo2.jpg

```
clear all; close all; clc;  
im=imread('diapo2.jpg');  
subplot(1,2,1)  
imshow(im);  
im(:, :, 1)=im(:, :, 1)*1.3  
im(:, :, 2)=im(:, :, 2)*1.8  
im(:, :, 3)=im(:, :, 3)*1.6  
subplot(1,2,2)  
imshow(im);
```



On ajoute de tout car l'image est trop sombre mais moins de rouge que le reste car l'image est déjà un peu rougeatre.

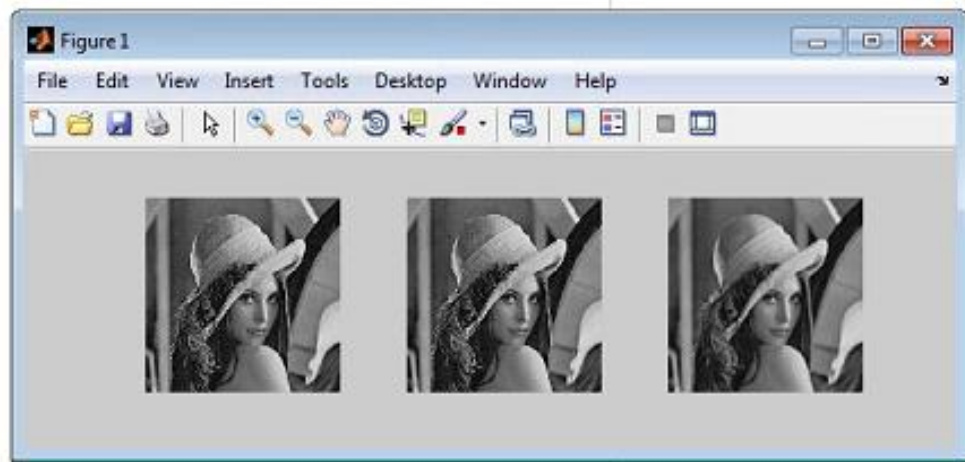
6.3.2 Filtre moyeneur

1. Ouvrir l'image lenna.bmp
2. Placer des pixels noirs bien visibles et non adjacents sur l'image
3. Ecrire un programme qui remplace les points de l'image par la moyenne des 9 points voisins (ne pas traiter les points manquants sur les bords). Visualiser le résultat.

```
clear all; close all; clc;
im=imread('lenna.bmp');
im=im2double(im);
subplot(1,3,1);
imshow(im);

im(23,56)=0;
im(23,150)=0;
im(53,56)=0;
im(168,83)=0;
im(200,156)=0;

subplot(1,3,2);
imshow(im);
[x y]=size(im);
im2=im;
for i=2:x-1
    for j=2:y-1
        im2(i,j)=(im(i+1,j+1)+im(i-1,j-1) ...
            +im(i+1,j-1)+im(i-1,j+1)+im(i+1,j) ...
            +im(i,j+1)+im(i-1,j)+im(i,j-1)+im(i,j))/9;
    end;
end;
subplot(1,3,3);
imshow(im2);
```



On a donc ajouté 5 pixels noirs au sein de l'image à des coordonnées bien précise.

Ensuite on effectue la moyenne des 8 pixels entourant un pixel et lui-même sur toute l'image. Donc chaque pixels est modifié et correspond à la moyenne de 9 pixels, l'image est légèrement floutée.

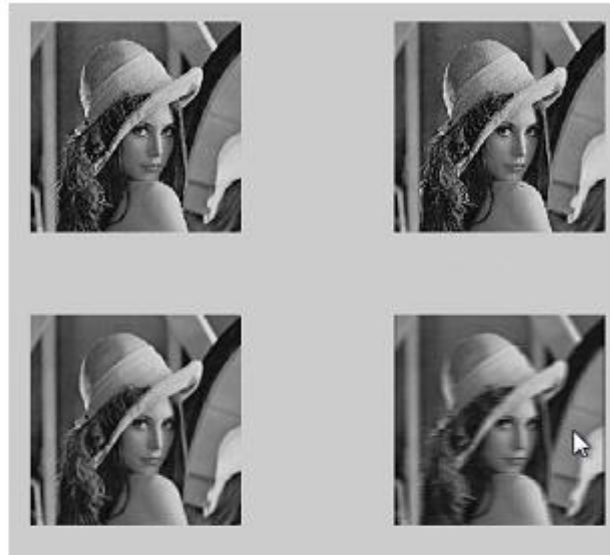
Evidement on ne voit rien, mais les points noirs ont bien été remplacés par la moyenne des pixels qui l'entoure et lui-même, il est donc possible de retrouver l'emplacement de ce point mais il est moins visible.

4. Effectuer la même opération au moyen de l'instruction `imfilter(I,h)` ou `h` est une matrice 3x3 composée de $1/9$.

```
clear all; close all; clc;
im=imread('lenna.bmp');
im=im2double(im);
subplot(2,2,1);
imshow(im);

im(23,56)=0;
im(23,150)=0;
im(53,56)=0;
im(168,83)=0;
im(200,156)=0;

subplot(2,2,2);
imshow(im);
[x y]=size(im);
im2=im;
for i=2:x-1
    for j=2:y-1
        im2(i,j)=(im(i+1,j+1)+im(i-1,j-1) ...
            +im(i+1,j-1)+im(i-1,j+1)+im(i+1,j) ...
            +im(i-1,j)+im(i,j+1)+im(i,j-1)+im(i,j))/9;
    end;
end;
subplot(2,2,3);
imshow(im2);
im3=imfilter(im,[[1/9 1/9 1/9] [1/9 1/9 1/9] [1/9 1/9 1/9]]);
subplot(2,2,4);
imshow(im3);
```



Même principe mais cette fois ci en utilisant une fonction toute faite. On ne voit plus du tout les points noirs mais le résultat est plus flou que la technique précédente.

5. Recommencer l'opération avec une matrice h 5×5 . Que deviennent les points noirs et l'image ?

```
clear all; close all; clc;
im=imread('lenna.bmp');
im=im2double(im);
subplot(2,2,1);
imshow(im);

im(23,56)=0;
im(23,150)=0;
im(53,56)=0;
im(168,83)=0;
im(200,156)=0;

subplot(2,2,2);
imshow(im);
[x y]=size(im);
im2=im;
for i=2:x-1
    for j=2:y-1
        im2(i,j)=(im(i+1,j+1)+im(i-1,j-1) ...
            +im(i+1,j-1)+im(i-1,j+1)+im(i+1,j) ...
            +im(i,j+1)+im(i-1,j)+im(i,j-1)+im(i,j))/9;
    end;
end;
subplot(2,2,3);
imshow(im2);
im3=imfilter(im,[1 1 1 1 1] [1 1 1 1 1] [1 1 1 1 1] [1 1 1 1 1] [1 1 1 1 1])/25;
subplot(2,2,4);
imshow(im3);
```



L'image est d'autant plus floue car on se base sur 25 points entourant chaque pixel et on fait leur moyenne.

6. Ecrire un programme sous Matlab qui atténue les points noirs en effectuant une moyenne sur les 8 points voisins (en ne prenant pas la valeur centrale). Visualiser le résultat. Est-il meilleur? A quelle matrice h correspond cette opération?

```
clear all; close all; clc;
im=imread('lenna.bmp');
im=im2double(im);
subplot(2,2,1);
imshow(im);

im(23,56)=0;
im(23,150)=0;
im(53,56)=0;
im(168,83)=0;
im(200,156)=0;

subplot(2,2,2);
imshow(im);
[x y]=size(im);
im2=im;
for i=2:x-1
    for j=2:y-1
        im2(i,j)=(im(i+1,j+1)+im(i-1,j-1) ...
            +im(i+1,j-1)+im(i-1,j+1)+im(i+1,j) ...
            +im(i,j+1)+im(i-1,j)+im(i,j-1))/8;
    end;
end;
subplot(2,2,3);
imshow(im2);
```



Résultat légèrement meilleur et toujours légèrement flou par rapport à l'image de base.

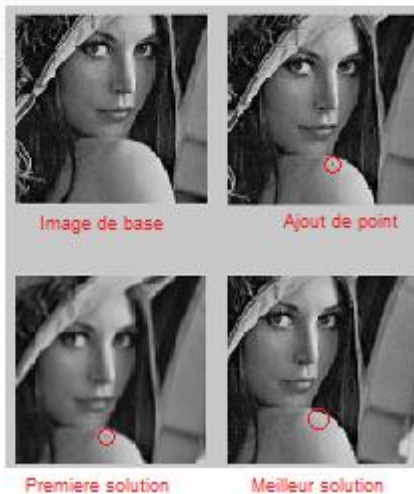
7. Essayer d'améliorer ce filtre par exemple, en ne l'appliquant qu'aux points noirs "isolés" c-à-d lorsque la différence entre un point et ses voisins dépasse un certain seuil.

```

im(200,156)=0;

subplot(2,2,2);
imshow(im);
[x y]=size(im);
im2=im;
for i=2:x-1
    for j=2:y-1
        im2(i,j)=(im(i+1,j+1)+im(i-1,j-1) ...
            +im(i+1,j-1)+im(i-1,j+1)+im(i+1,j) ...
            +im(i,j+1)+im(i-1,j)+im(i,j-1)+im(i,j))/9;
    end;
end;
subplot(2,2,3);
imshow(im2);
im3=im;
for i=2:x-1
    for j=2:y-1
        m=(im(i+1,j+1)+im(i-1,j-1) ...
            +im(i+1,j-1)+im(i-1,j+1)+im(i+1,j) ...
            +im(i,j+1)+im(i-1,j)+im(i,j-1))/8
        if (m-im(i,j))>0.2
            im3(i,j)=m;
        end
    end;
end;
subplot(2,2,4);
imshow(im3);

```



```

If(m-im(i,j))>0.2)
    Im3(i,j)=m ;
End

```

==> si la **différence** de niveau de gris entre la **moyenne** sur les pixels entourant chaque pixel **et** la valeur de ce **pixel** est **plus grande** que **0.2** on **remplace** le **pixel** par cette **moyenne**. (0.2 à été retrouvé après quelques test, 0.25 est en fait plus précis ! (mais flemme de redécouper l'image...)).

8. Appliquer un filtre median (medfilt2) à votre image initiale et comparer votre résultat avec votre image filtrée par le filtre moyenneur.

```
clear all; close all; clc;
im=imread('lenna.bmp');
im=im2double(im);
subplot(2,2,1);
imshow(im);
im(23,56)=0;
im(23,150)=0;
im(53,56)=0;
im(168,83)=0;
im(200,156)=0;
subplot(2,2,2);
imshow(im);
[x y]=size(im);

im3=im;
for i=2:x-1
    for j=2:y-1
        m=(im(i+1,j+1)+im(i-1,j-1) ...
            +im(i+1,j-1)+im(i-1,j+1)+im(i+1,j) ...
            +im(i,j+1)+im(i-1,j)+im(i,j-1))/8;
        if(m-im(i,j)>0.25)
            im3(i,j)=m;
        end
    end
end;
subplot(2,2,3);
imshow(im3);

im4=medfilt2(im,[3 3]);
subplot(2,2,4);
imshow(im4);
```



L'image est lissée, les points on disparus mais perd certain de ces détails.

Le filtre médian :

8	8	8	8	8	8
8	8	8	8	8	8
8	8	8	8	8	8
8	8	8	8	8	8
8	8	0	8	255	8
8	8	8	8	8	8

voisinage

bruit impulsif

On applique un filtre médian avec un voisinage 3x3. Celui-ci ordonne les valeurs des pixels du voisinage par ordre croissant, et attribue en sortie la valeur médiane sur ce voisinage aux pixels à filtrer.

```
Filtre médian  
If2=image_bruit;  
If2 = medfilt2(image_bruit,[3 3]);
```

Après l'effort le réconfort :



6.3.3 Objets, régions, mesures sur une image

La fonction `bwlabel` permet de réaliser un nombre très important d'opérations sur les images.

Elle réalise tout d'abord, la "labelisation" de l'image.

Cette opération consiste à repérer les objets non connexes de l'image puis à numéroter tous les pixels de chaque objet par un même nombre. Chacun des points d'un objet sera donc identifié par un même chiffre

En comptant le nombre de pixels par objet, on trouvera sa surface. On pourra également déterminer un grand nombre de propriétés de ceux-ci : centre de gravité, excentricité, ...

La fonction `|regionprops|` permet d'obtenir ces propriétés.

Par ex :

`S=regionprops (L, Excentricity')`, `(L, 'Area')`... Cette fonction s'applique uniquement à des images binaires.

La méthode est donc la suivante :

1. convertir l'image albireo.bmp en image binaire (nommée L par ex.) au moyen des fonctions `im2bw` et `graythresh`

```
close all;
clear all;
clc;
image=imread('TEST_5.jpg');

if size(image,3)==3
    img=im2bw(graythresh(img));
end
imshow(image);
```

Il est intéressant d'utiliser une image logique et donc de coder chaque pixel sur un bit : soit sa valeur est de 1 soit de 0. La fonction `im2bw` permet de binariser l'image.

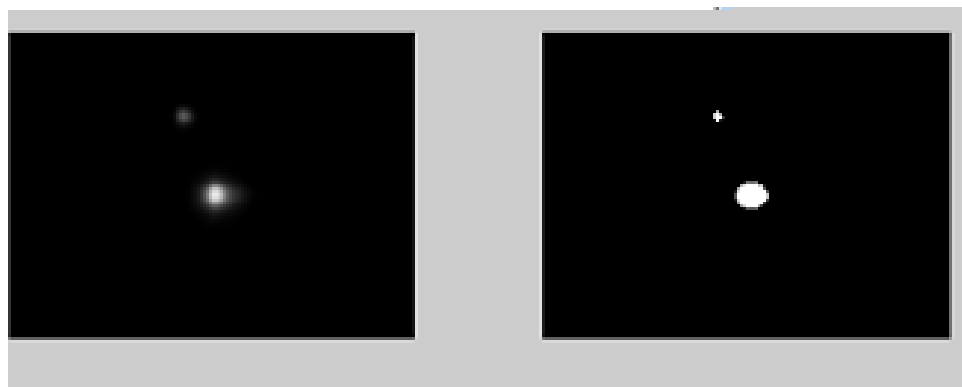
La fonction `graythresh` utilise la méthode d'Otsu's qui consiste à effectuer un seuillage automatique à partir de la forme de l'histogramme de l'image ou la réduction d'une image à niveaux de gris en une image binaire. L'algorithme suppose alors que l'image à binariser ne contient que deux classes de pixels, (le premier plan et l'arrière-plan) puis calcule le seuil optimal qui sépare ces deux classes afin que leur variance intra-classe soit minimale.

2. l'instruction `L=bwlabel(L,4)` par ex. va labeliser l'image binaire en objets dont 4 pixels maximum sont en commun

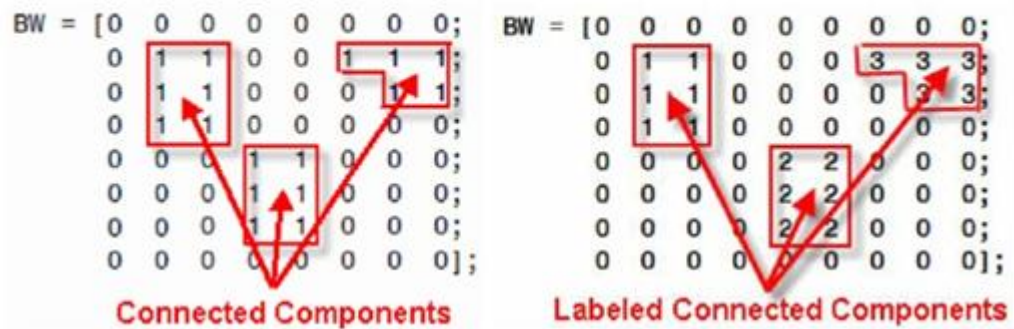
```
clear all; close all; clc;
im=imread('albireo.bmp');
im=im2double(im);
subplot(2,2,1);
imshow(im);

im2=im2bw(im,graythresh(im));
subplot(2,2,2);
imshow(im2);

bwlabel(im2);
regionprops(im2)
```



La fonction **bwlabel()** va compter le nombre d'objet dans l'image et les identifier par un numéro logique.



(e) Utiliser l'instruction **regionprops** pour mesurer les propriétés des objets. **Regionprops** renverra un « array of structure » [S Area] renverra un vecteur représentant la surface de chaque objets.

(f) Repérer la taille des deux éléments ayant la plus grande surface.

```
clear all; close all; clc;
im=imread('albireo.bmp');
im=im2double(im);
subplot(2,2,1);
%imshow(im);

im2=im2bw(im,graythresh(im));
subplot(2,2,2);
%imshow(im2);

bwlabel(im2);
r=regionprops(im2)
r(1)
r(2)
```

```
r =

2x1 struct array with fields:
    Area
    Centroid
    BoundingBox

ans =

    Area: 13
    Centroid: [83.3077 40.0769]
    BoundingBox: [1x4 double]

ans =

    Area: 152
    Centroid: [98.8487 77.1053]
    BoundingBox: [1x4 double]
```

Area est la surface de l'image, centroid est le centre de gravité de l'image (pas le centre même !!) et boundingbox est une boite encadrant l'image en fonction de sa surface.

(g) Une fonction `find` renverra les indices des éléments dont la taille est supérieure à cette valeur (par exemple l'objet contenant des 2 et l'objet contenant des 5).



(l'imageaudessus)

Trouver l'objet contenant l'étiquette 2 :

```
clc;
close all;
clear all;

im = imread('l'imageaudessus.jpg');
im2 = im2bw(im, graythresh(im));
[X num] = bwlabel(im2);
[rows, cols, vals] = find(X == 2)
```

puis celui étiqueté à 5 :

```
clc;
close all;
clear all;

im = imread('l'imageaudessus.jpg');
im2 = im2bw(im, graythresh(im));
[X num] = bwlabel(im2);
[rows, cols, vals] = find(X == 5)
```

On obtient donc les coordonnées de chaque pixel de l'objet recherché et sa valeur (donc 1 vu que c'est du blanc).

2. Enlever les petites tâches d'une image :

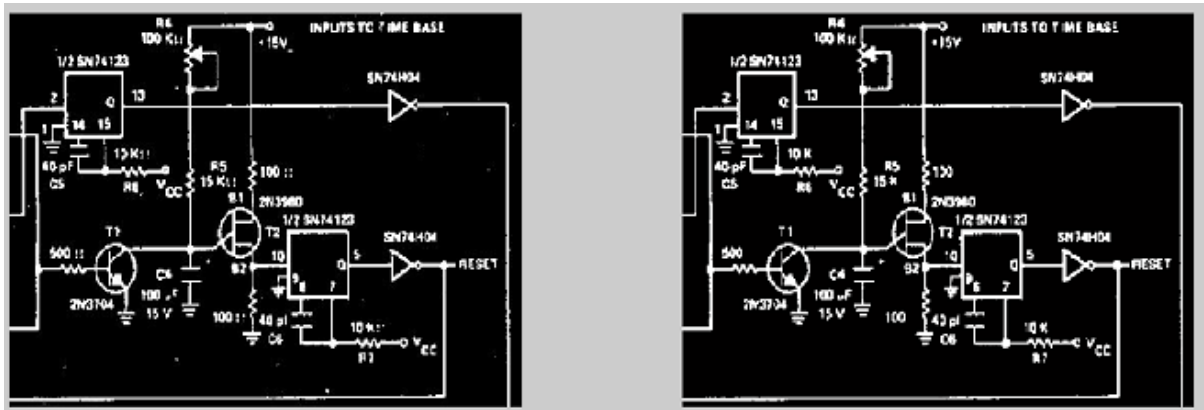
(a) Ouvrir l'image 'circuit.tif'

(b) Constaté que l'image comporte de petites tâches.

Utiliser la fonction `bwlabel` pour les éliminer

```
clear all; close all; clc;

I=imread('Image/circuit1.tif');
subplot(1,2,1)
imshow('Image/circuit1.tif');
I = im2bw(I);
I = bwareaopen(I,10);
subplot(1,2,2);
imshow(I);
```



On supprime les objets blanc de moins et égal à 10 pixels.

3. Selectionner les parties allongées d'une image.

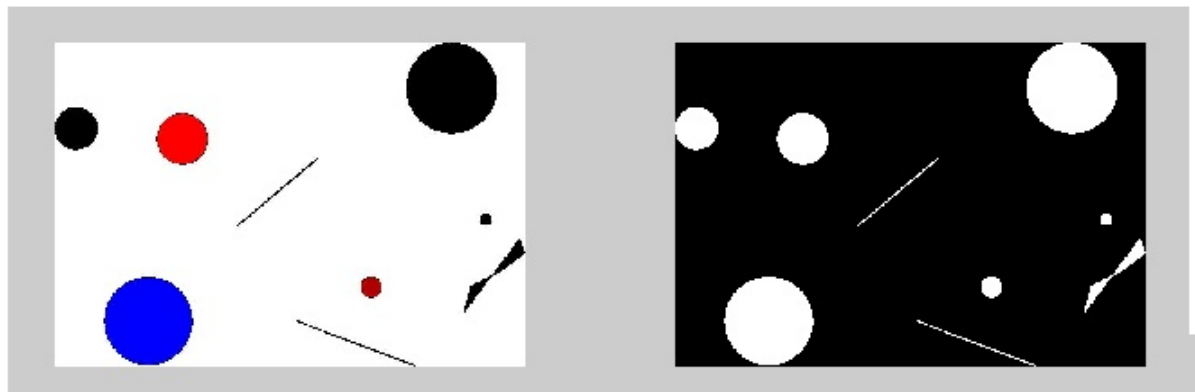
Soit l'image **bwlabel_exc.jpg**

On procèdera de la même manière que dans l'exemple précédent mais on utilisera la propriété d'excentricité (S.Eccentricity)

Il faudra inverser l'image binaire avant de la labeliser car la fonction bwlabel met le background à 0.

Ne pas le faire impliquerait de n'avoir qu'un seul objet : le background

```
clear all;close all; clc;
I=imread('Image/bwlabel_exc.jpg');
subplot(1,2,1)
imshow('Image/bwlabel_exc.jpg');
I = im2bw(I);
I = ~I;
I = bwlabel(I);
R = regionprops(I,'Eccentricity')
for i=1:size(R)
    R(i)
end
subplot(1,2,2);
imshow(I);
```



<pre> R = 9x1 struct array with fields: Eccentricity ans = Eccentricity: 0.0060 ans = Eccentricity: 0.0331 ans = Eccentricity: 0.0307 ans = Eccentricity: 0.9998 </pre>	<pre> ans = Eccentricity: 0.9999 ans = Eccentricity: 0.0646 ans = Eccentricity: 0.0475 ans = Eccentricity: 0.9933 ans = Eccentricity: 0.1336 </pre>
--	--

Ces données représentent les coefficients d'excentricité. Plus celui-ci est petit, plus la forme se rapproche d'un cercle.

6.3.4 Détection de bords

1. Ouvrir l'image 'pout.tif' dénommée I.
2. Ecrire un programme qui remplace chaque pixel de l'image par la valeur du pixel de gauche moins la valeur du pixel du dessous. Afficher le résultat. Donner la valeur de la matrice h auquel ce filtre correspond.

```

%exo 6.3.4
I=imread('pout.tif'); % 1.
[x y]=size(I);
subplot(2,3,1);
imshow(I);

% 2.
im=I;
for xx=2:x
    for yy=1:y-1
        im(xx,yy)=I(xx-1,yy)-I(xx,yy+1);
    end
end
subplot(2,3,2);
imshow(im);

```

3. Un bord (edge en anglais) est un chemin le long duquel l'intensité d'une image varie beaucoup. Un bord correspondra souvent à la frontière d'un objet.

Afficher les bords des objets par l'instruction :

`>>edge(I,'sobel')` où I est l'image à traiter

On peut aussi essayer :

`>>edge(I,'canny')` où I est l'image à traiter

Constaté la différence entre ces deux résultats.

```
I=imread('pout.tif');
[x y]=size(I);
% 3.
subplot(2,3,3);
edge(I,'sobel');
subplot(2,3,4);
edge(I,'canny');
```

4. Si on exécute :

`h=fspecial('sobel')` on obtient :

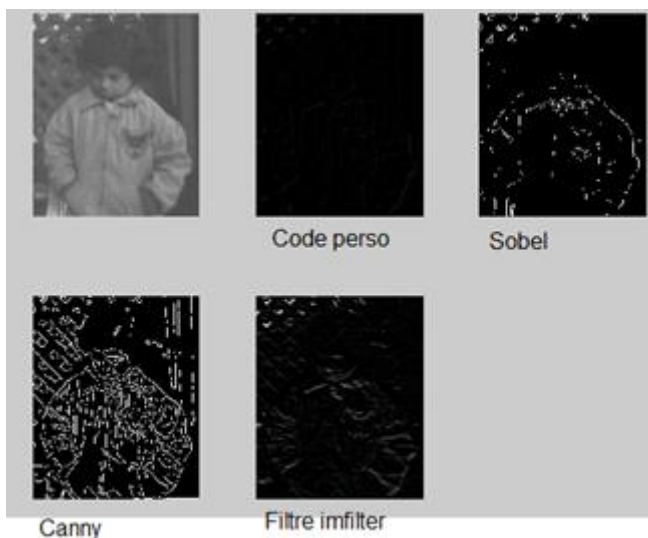
$$h = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

On constate que, pour le calcul d'un nouveau pixel, les pixels de la ligne horizontale ne sont pas pris en compte et les valeurs des points sur la verticale sont soustraites. Si on applique l'instruction avec l'option 'canny', on n'obtient pas le même résultat que précédemment.

Essayer également l'instruction `imfilter(I,h)`.

Interpréter les résultats obtenus.

```
I=imread('pout.tif');
[x y]=size(I);
% 4.
subplot(2,3,5);
h=fspecial('sobel')
imshow(imfilter(I,h));
```



6.3.5 Ajustement d'une image

1. Que signifient :

- (a) contraste
- (b) histogramme
- (c) brillance (brightness) .
- (d) correction gamma

(a) Contraste

Le contraste est une propriété intrinsèque à une image qui permet de quantifier, la capacité de distinguer deux régions distinctes. Il s'agit dans ce cas de distinguer deux régions suffisamment grandes d'après l'intensité des points représentés par des niveaux de gris en image numérique.

Le contraste est une quantité qui doit varier de 0 pour des zones de même intensité à 1 pour les zones d'intensités les plus différentes. Si on note I_{\max} l'intensité maximale et I_{\min} l'intensité minimale, le contraste C peut être défini par :

$$C = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}$$

On voit que si l'intensité minimale est nulle (point image noire), le contraste est maximal.

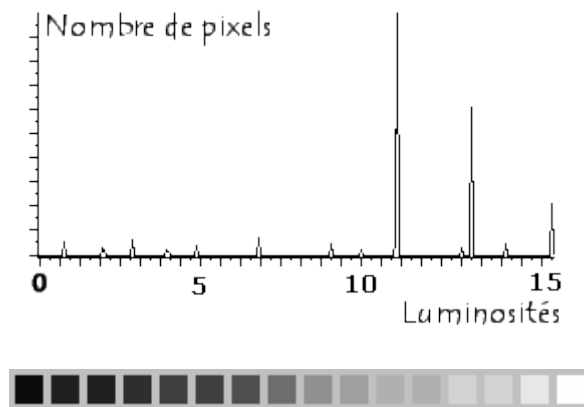
(b) Histogramme

Un histogramme est un graphique statistique permettant de représenter la distribution des intensités des pixels d'une image, c'est-à-dire le nombre de pixels pour chaque intensité lumineuse. Par convention un histogramme représente le niveau d'intensité en abscisse en allant du plus foncé (à gauche) au plus clair (à droite).

Ainsi, l'histogramme d'une image en 256 niveaux de gris sera représenté par un graphique possédant 256 valeurs en abscisses, et le nombre de pixels de l'image en ordonnées. Prenons par exemple l'image suivante composée de niveaux de gris :



L'histogramme et la palette associés à cette image sont respectivement les suivants :



L'histogramme fait apparaître que les tons gris clairs sont beaucoup plus présents dans l'image que les tons foncés.

Le ton de gris le plus utilisé est le 11^{ème} en partant de la gauche.

Pour les images en couleur plusieurs histogrammes sont nécessaires. Par exemple pour une image codée en RGB :

- un histogramme représentant la distribution de la luminance,
- trois histogrammes représentant respectivement la distribution des valeurs respectives des composantes rouges, bleues et vertes.

(c) Brillance (brightness).

La luminosité est un attribut de la perception visuelle dans laquelle une source semble être rayonnant ou réfléchissant la lumière. En d'autres termes, la luminosité est la perception induite par la luminance d'une cible visuelle. C'est un attribut subjectif / propriété d'un objet observé.

(d) Correction gamma

Les appareils d'acquisition vidéo, photo et autres scanners ne savent pas directement enregistrer des images linéarisées pour les écrans ou pour l'impression. Ils sont donc équipés d'un traitement numérique pour adapter les images à notre vision. C'est la correction gamma.

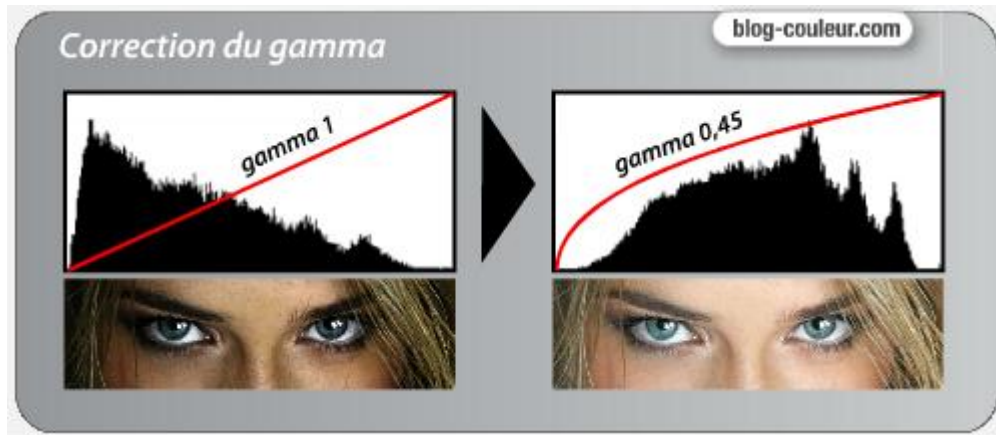


Fig.1. On appelle image linéarisée, une image ayant subi la correction gamma pour être conforme à notre vision.

Le principe de la correction gamma est un principe fondamental dans le flux de reproduction des images, car il modifie la distribution tonale pour la faire entrer dans un standard.

2. Comment corriger une image par ajustement d'intensité ou par égalisation de l'histogramme ?

```
I=imread('pout.tif');
subplot(2,2,1);
imshow(I);

% 2.
im=im*1.2; % on augmente l'intensité
subplot(2,2,2);
imshow(im);

im=histeq(I); % égalisation histogramme
subplot(2,2,3);
imshow(im);
```


3. Lire l'aide de la fonction `imadjust`

Utiliser la fonction `imadjust` pour améliorer l'image `pout.tif` par correction gamma.

```
% 3.  
I=imread('pout.tif');  
im=imadjust(I); % correction gamma  
subplot(2,2,4);  
imshow(im);
```

Résultat:

```
I=imread('pout.tif');  
subplot(2,2,1);  
imshow(I);  
  
% 2.  
im=im*1.2; % on augmente l'intensité  
subplot(2,2,2);  
imshow(im);  
  
im=histeq(I); % égalisation histogramme  
subplot(2,2,3);  
imshow(im);  
  
% 3.  
im=imadjust(I); % correction gamma  
subplot(2,2,4);  
imshow(im);
```

