Notes Matlab - Images

```
-) Morgan Valentin
```

Table des matières

- Images
  - Ouverture, affichage et types d'images
    - 1) Placer un carré noir (10x10) dans le coin supérieur gauche, visualiser et sauver dans un fichier
    - 2) Visualiser sur 4 figures différentes, l'image, sa composante verte, rouge et bleu
    - 3) Créer un carré 400x400 rouge d'intensité maximale
    - 4) Créer un carré 400x400 dont la couleur est égale au fond bleu se trouvant dans le coin inférieur droit de l'image flowers
    - 5) Note sur les classes d'images
  - Opérations arithmétiques sur une image
    - 1) Ajout et division de valeurs
    - 2) Afficher une image en niveau de gris manuellement et via rgb2gray
      - a) Manuellement
      - b) Via rgb2gray
    - 3) Dessiner en noir chaque pixel de l'image poutif
  - Traitement d'images-filtrage
    - 1) Corriger l'image diapo1 qui est trop rouge
    - 2) Filtre moyenneur
      - a) Appliquer un filtre moyenneur à l'image lenna
    - 3) Objets, régions, mesures sur une image
      - 1) Manipulation d'objets d'une image
      - 2) Enlever les petites taches d'une image
      - 3) Afficher uniquement les formes qui ressemblent à des lignes
      - 4) Afficher uniquement les lettres a de l'image text
      - 5) Afficher les différentes pièces de l'image pieces et afficher uniquement la plus grande pièce
    - 4) Détection de bords
      - a) Détection de bords d'une image en couleur
      - b) 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 et donner la valeur de la matrice h auquel ce filtre correspond.
    - 5) Ajustement d'une image

Images

Ouverture, affichage et types d'images

1) Placer un carré noir (10x10) dans le coin supérieur gauche, visualiser et sauver dans un fichier

```
fh = figure;    % on créée une figure
imshow(I);      % on affiche l'image dans la figure
hold on;        % retains plots in the current axes so that new plots added to
the axes do not delete existing plots

rectangle('Position', [0 0 10 10], 'FaceColor', 'k');   % on dessine le
rectangle
% position = [coinSuppGauche X, coinSuppDroiteY, tailleCarre X, tailleCarre Y]
frm = getframe(fh);

imwrite(frm.cdata, 'pout_carre.png'); % écrit les données de frm (crée juste
avant) dans le fichier

img2 = imread('pout_carre.png');
imshow(img2); % On affiche l'image
```

2) Visualiser sur 4 figures différentes, l'image, sa composante verte, rouge et bleu

Type d'image ?

```
img = imread('flowers.tif');

whos

    Name        Size                Bytes  Class    Attributes

    img       362x500x3            543000  uint8
```

Important :

- "`Size`" => Représente la taille de l'image
  - hauteur x largeur x dimansion
- "`Class`" => "`uint8`" signifie que l'image est en int (Integer quoi)

```
% image
img = imread('flowers.tif');
subplot(2,2,1);
imshow(img);
title('image');

% composante rouge
subplot(2,2,2);
R = img(:,:,1); % R c'est toutes les colonnes et toutes les lignes de img mais
seulement la première composantes RGB (donc R)
imshow(R);
title('R');

% composante verte
subplot(2,2,3);
G = img(:,:,2); % G c'est toutes les colonnes et toutes les lignes de img mais
seulement la seconde composantes RGB (donc G)
imshow(G);
title('G');

% composante bleue
subplot(2,2,4);
B = img(:,:,3); % B c'est toutes les colonnes et toutes les lignes de img mais
seulement la troisième composantes RGB (donc B)
imshow(B);
title('B');
```

Retenir qu'au niveau des couleurs....

RGB = Red Green Blue :

- Les valeurs vont de 0 jusqu'à 255.
- En niveau de noir / blanc,
  - Une valeur à 0 => Très noir
  - Une valeur à 255 => Très blanc

> Si on regarde le jaune (255,255,0) en niveau de bleu, on verra noir, et au niveau du R et G, on verra blanc.

3) Créer un carré 400x400 rouge d'intensité maximale

```
carre = rectangle('Position', [0 0 400 400], 'FaceColor', [1 0 0]);
```

> Entre autre, le "[1 0 0]" => RGB => Modifie le vert intensité max ? => [0 1 0]

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

```
img = imread('flowers.tif');

pixel = impixel(img, 500, 362); % on prends les valeurs du pixel en bas à droite de img
pixel = pixel./1000             % on divise toutes les valeurs rgb pour qu'elles soient entre 0 et 1

rectangle('Position', [0 0 400 400], 'FaceColor', pixel);
```

> **Le "whos" a renvoyé comme taille d'image "362x500x3" (hauteur x largeur x dimension). impixel(image, largeur, hauteur).**

5) Note sur les classes d'images

Notre image est en unint8...

```
img = imread('flowers.tif');
imgDouble = double(img);
imshow(imgDouble); % On remarque que l'image est blanche...
imgDouble = im2double(img);
```

> L'image est blanche car la classe Double n'affiche que les niveaux de couleurs entre 0 et 1.

Opérations arithmétiques sur une image

1) Ajout et division de valeurs

```
J = imadd(img, 200) % où img est notre image et 200 est la valeur à ajouter.
```

```
Z = imdivide(img, 255) % où img est notre image et 255 est la valeur de
diviseur.
```

2) Afficher une image en niveau de gris manuellement et via rgb2gray

a) Manuellement

La luminance vaut : 0,299R + 0,587G + 0,114B

```
img = imread('flowers.tif');
subplot(1, 1, 1);
manual_gray = 0.299 * img(:,:,1) + 0.587 * img(:,:,2) + 0.114 * img(:,:,3);
imshow(manual_gray);
title('Manual Gray');
```

b) Via rgb2gray

```
img = imread('flowers.tif');
subplot(1, 1, 1);
gray = rgb2gray(img);
imshow(gray);
title('rgb2gray');
```

3) Dessiner en noir chaque pixel de l'image poutif

```
im=imread('pout.tif');

% (a)
tic
    im(1:291,1:240)=0;
    imshow(im);
toc
Elapsed time is 0.15 seconds.
```

> Les intructions `tic` et `toc` permettent de visualiser le temps d'exécution d'un programme.
> Retenir que la vectorisation est **beaucoup plus rapide** que les boucles.

Traitement d'images-filtrage

1) Corriger l'image diapo1 qui est trop rouge

```
I = imread('diapo1.jpg');
improfile % tirer une ligne sur l'image et appuyer sur "Enter". Diagonale haut gauche à Bas droite.
I(:,:,1) = I(:,:,1).*0.625; % Pour chaque ligne, chaque colonne, multiplier les valeurs de Rouge (actuelles) par 0,625.
imshow(I);
```

On voit que dans l'intervalle 400-450 :

- R = 200
- G et B = 125
- Donc 125/200 = 0,625

---

2) Filtre moyenneur

Les filtres moyenneurs, comme leur noms l'indique, calculent la moyenne, éventuellement pondérée, des pixels situés dans le voisinage de chaque pixel. Cette famille de filtres permet de réduire le bruit dans l'image, ce qui rend les zones homogènes plus lisses. Par contre, les contours sont fortement dégradés, et les structures trop fines peuvent devenir moins visibles.

```
M = ones(3,3)/9; % matrice filtre moyenneur
```

> *C'est une matrice 3x3 avec que des 1/9.*

a) Appliquer un filtre moyenneur à l'image lenna

```
img = imread('LENNA.BMP');
subplot(1,2,1);
imshow(img);
title('Original');

h = ones(3,3)/9; % Filtre moyenneur
x = imfilter(img, h); % Application du filtre à l'image
subplot(1,2,2);
imshow(x);
title('Image filtré via filtre moyenneur');
```

3) Objets, régions, mesures sur une image

"`bwlabel`" => Permet de "labéliser" une image, c'est-à-dire repérer des objets dans une image puis à numéroter tous les pixels de chaque objet par un même nombre.

Chaque point d'un objet sera donc identifier par un même chiffre.

Grâce à cela, on pourra donc déterminer :

- `La surface` (la taille de l'objet quoi)
- `Le centre de gravité`
- `L'excentricité` :
  - Si la valeur est ***proche de 0*** => **Cercle**.
  - Si la valeur est ***proche de 1*** => **Droite**.

---

1) Manipulation d'objets d'une image

```
subplot(1,3,1);
img = imread('albireo.bmp');
imshow(img);
title('Original');

% Tranforme l'image en binaire
subplot(1,3,2);
level = graythresh(img);
img_bin = im2bw(img, level);
imshow(img_bin);
title('Binaire');

 % Labiliser l'image.
[img_label, nb_elements] = bwlabel(img_bin);

% Structure + tableau contenant les valeurs des surfaces des objets labilisés
stats = regionprops(img_label); % Structure contenant des propriétés : "Area",
"Centroid", "BoundingBox"
%stats.Area; Renvoie la taille des zones labelisées (chaque objet de l'image)
%stats(1).Area; Renvoie la taille du 1ère élement labilisé.
area_values = [stats.Area];

%find(area_values); % Renvoie les indices de chaque objets labilisé (dans notre
cas, il y a 2 objets)
find(area_values<15); % Renvoie l'indice 1, vu que le 1ère élément à comme
taille 13 [stat(1).Area]

subplot(1,3,3);
img_objet2 = bwareaopen(img_bin,15); % Va afficher les objets de l'image binaire
"img_bin" dont la taille de ces objets sont supérieur à 15 (donc uniquement la
grosse boule)
imshow(img_objet2);
title('Grosse boule - bwareaopen');
```

2) Enlever les petites taches d'une image

```
subplot(1, 3, 1);
img = imread('circuit1.tif');
imshow(img);
title('Original');

% Tranforme l'image en binaire
subplot(1,3,2);
level = graythresh(img);
img_bin = im2bw(img, level);
imshow(img_bin);
title('Binaire');

% Labiliser l'image.
[img_label, nb_elements] = bwlabel(img_bin);

% Structure + tableau contenant les valeurs des surfaces des objets labilisés
stats = regionprops(img_label);
area_values = [stats.Area];

% PS : On remarque que chaque point à retirer à l'air de faire max 5 pixels.
petit_points = find(area_values > 5);

% Image nettoyé
subplot(1, 3, 3);
img_nettoye = ismember(img_label, petit_points);
imshow(img_nettoye);
title('Image corrigee');
```

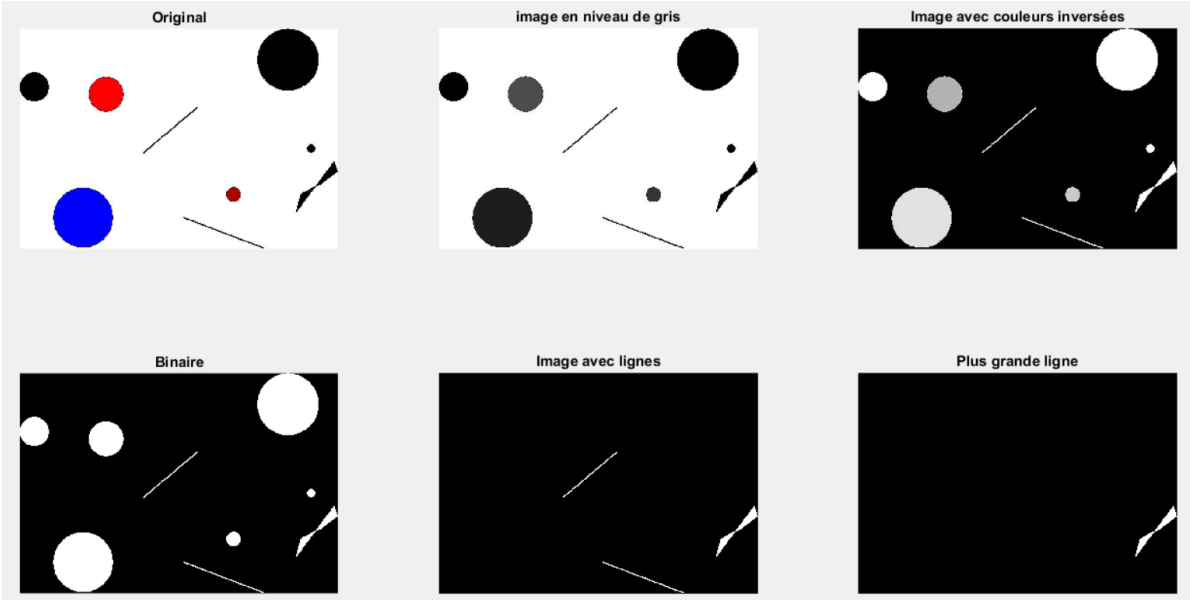
On peut remplacer les lignes :

- `petit_points = find(area_values > 5);`
- `img_nettoye = ismember(img_label, petit_points);`

Par la ligne `img_nettoye = bwareaopen(img_label, 5);` UNIQUEMENT car on utilise les "Area".

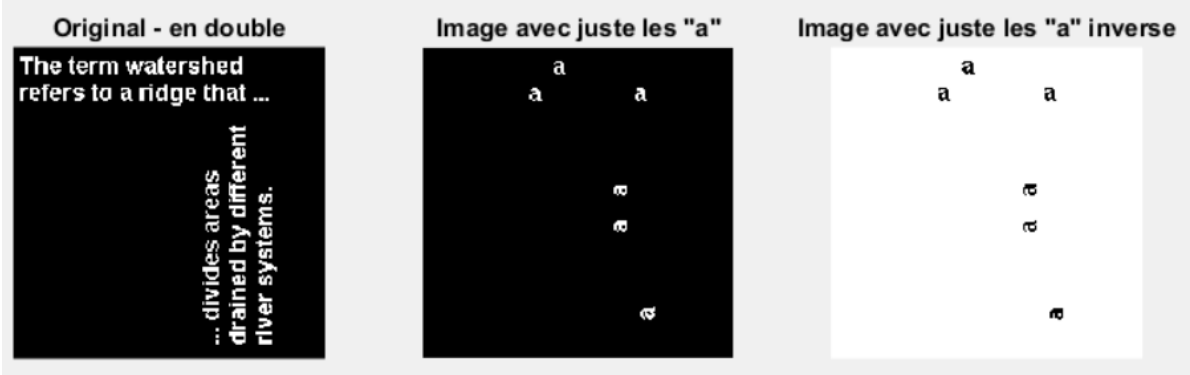
3) Afficher uniquement les formes qui ressemblent à des lignes

```
subplot(2,3,1);
img = imread('bwlabel_exc.jpg');
imshow(img);
title('Original');

% Transforme l'image en niveau de gris
subplot(2,3,2);
img_gray = rgb2gray(img);
imshow(img_gray);
title('image en niveau de gris');

% Inverse les couleurs blanches et noir
subplot(2,3,3);
img_inverse = imcomplement(img_gray);
imshow(img_inverse);
title('Image avec couleurs inversées');

% Tranforme l'image en binaire
subplot(2,3,4);
level = graythresh(img_inverse);
img_bin = im2bw(img_inverse, level);
imshow(img_bin);
title('Binaire');

% Labiliser l'image.
[img_label, nb_elements] = bwlabel(img_bin);

% Structure + tableau contenant les valeurs des surfaces des objets labilisés
stats = regionprops(img_label, 'Area', 'Eccentricity'); % Area et Eccentricity
permettent de spécifier les propriétés qui nous intéresse au lieu de nous
retrouver avec celles de base
eccentricity_values = [stats.Eccentricity];

lines = find(eccentricity_values > 0.8);

% Image nettoyé
subplot(2,3,5);
img_nettoye = ismember(img_label, lines);
imshow(img_nettoye);
title('Image avec lignes');

% Bonus : Trouve la ligne la plus grande (en pixel quoi)
subplot(2,3,6);
grande_ligne=0;
grande_ligne_index=0;
for i = 1 : length(lines)
   if grande_ligne < stats(lines(i)).Area
      grande_ligne = stats(lines(i)).Area;
      grande_ligne_index = lines(i);
   end
end
img_ligne_grande = ismember(img_label, grande_ligne_index);
imshow(img_ligne_grande);
title('Plus grande ligne');
```


Original
image en niveau de gris
Image avec couleurs inversées
Binaire
Image avec lignes
Plus grande ligne

4) Afficher uniquement les lettres a de l'image text

```
subplot(1,3,1);
I = imread('text.png');

img = im2double(I); % Conversion en double car cette image est en type
"logical", on peut rien faire avec ça
imshow(img);
title('Original - en double');

level = graythresh(img); % Conversion en binaire
img_bin = im2bw(img, level);

[img_label, nb_elements] = bwlabel(img_bin); % Labiliser l'image quoi

stats = regionprops(img_label);
area_values = [stats.Area];

lettre_a = find(area_values == 68);

subplot(1,3,2);
img_lettre_a = ismember(img_label, lettre_a);
imshow(img_lettre_a);
title('Image avec juste les "a"');

% Bonus: Inverser les couleurs pour un meilleur visuel
subplot(1,3,3);
img_inverse = imcomplement(img_lettre_a);
imshow(img_inverse);
title('Image avec juste les "a" inverse');
```



5) Afficher les différentes pièces de l'image pieces et afficher uniquement la plus grande pièce

Lien de l'image

```
subplot(2,2,1);
img = imread('pieces.png');
imshow(img);
title('Original');

% PAS besoin d'inverse les couleurs vu que le fond est noir et les pieces sont
blanches

% Tranforme l'image en binaire
subplot(2,2,2);
level = graythresh(img);
img_bin = im2bw(img, level);
imshow(img_bin);
title('Binaire');

% Rempli en blanc la pièce noir - sinon elle disparait
img_bin_corrected = imfill(img_bin,'holes');
img_bin_corrected = bwareaopen(img_bin_corrected,30);

% Labiliser l'image.
[img_label, nb_elements] = bwlabel(img_bin_corrected);

% Structure + tableau contenant les valeurs des surfaces des objets labilisés
stats = regionprops(img_label, 'Area', 'Eccentricity');
eccentricity_values = [stats.Eccentricity];

cercles = find(eccentricity_values < 0.5);

% Image avec que les cercles - on ne prends pas les petits points
subplot(2,2,3);
img_cercles = ismember(img_label, cercles);
imshow(img_cercles);
title('Image avec cercles');

% Trouve le cercle le plus grand
subplot(2,2,4);
grand_cercle=0;
grand_cercle_index=0;
for i = 1 : length(cercles)
   if grand_cercle < stats(cercles(i)).Area
      grand_cercle = stats(cercles(i)).Area;
      grand_cercle_index = cercles(i);
   end
end
img_grand_cercle = ismember(img_label, grand_cercle_index);
imshow(img_grand_cercle);
title('Plus grand cercle');
```

Nombre de pièces ? => Voir la variable "`nb_elements`".

Original

Binaire

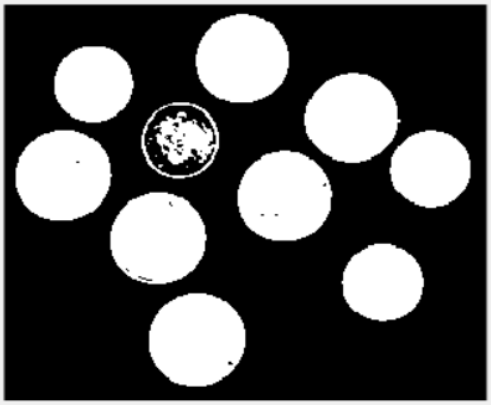
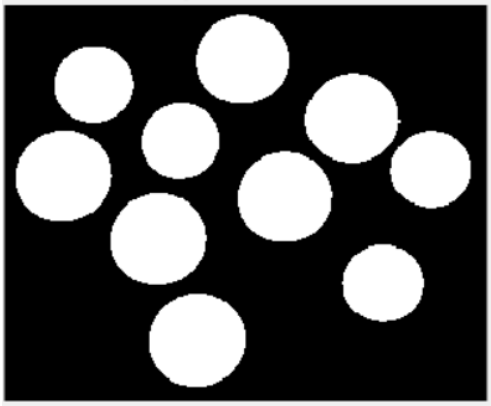
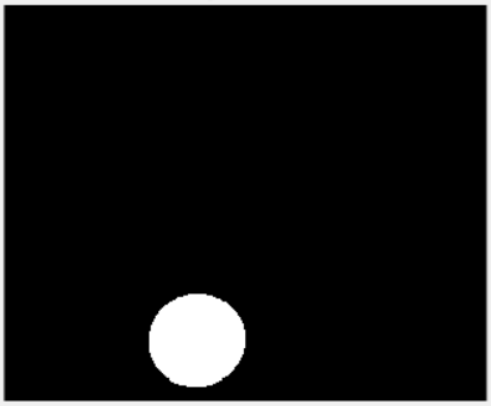


Image avec cercles



Plus grand cercle



4) Détection de bords

Pour la détection de bords, on utilise 3 méthodes :

1. Via la matrice (s'applique avec `imfilter`)
2. Via Sobel (s'applique avec `edge`)
3. Via Canny (s'applique avec `edge`)

> ***PS : Il faut une image en binaire pour pouvoir détecter les bords d'une image !***

> ***PPS : Du coup, on transforme on gris si l'image est en couleur, et une fois en gris, on la transforme en binaire.***

a) Détection de bords d'une image en couleur

```
subplot(2,3,1);
img = imread('flowers.tif');
imshow(img);
title('Original');

subplot(2,3,2);
img_gray = rgb2gray(img); % conversion de l'image en gris
imshow(img_gray);
title('Gray');

subplot(2,3,3);
level = graythresh(img_gray);
img_bin = im2bw(img_gray,level); % Conversion de l'image en binaire
imshow(img_bin);
title('Binaire');

% detection des bords - Sobel
subplot(2,3,4);
img_edged_sobel = edge(img_bin,'sobel');
imshow(img_edged_sobel);
title('Sobel');

% detection des bords - Canny
subplot(2,3,5);
img_edged_canny = edge(img_bin,'canny');
imshow(img_edged_canny);
title('Canny');

% detection des bords - Matrice
subplot(2,3,6);
matrice_detection_bords = [0 0 0; -1 1 0; 0 0 0]; % Matrice pour detecter les
bords
img_edged_matrice = imfilter(img_bin,matrice_detection_bords);
imshow(img_edged_matrice);
title('Matrice');
```

b) 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 et donner la valeur de la matrice h auquel ce filtre correspond.

```
img = imread('pout.tif');
subplot(1,3,1);
imshow(img);
title('Original');

subplot(1,3,2);
level = graythresh(img);
img_bin = im2bw(img,level); % Conversion de l'image en binaire
imshow(img_bin);
title('Binaire');


[w, h] = size(img_bin);
% remplacement chaque pixel par la valeur du pixel de gauche moins celui du
% dessous
for x=2 : w - 1
    for y = 2 : h - 1
        img_replace(x,y) = img_bin(x-1,y) - img_bin(x,y+1);
    end
end

subplot(1,3,3);
imshow(img_replace);
title('Pixel remplacé');
```

5) Ajustement d'une image

1. `Contraste` = Variation de l'ombre et la lumière dans une image
2. `Histograme` = Distribution des intensités de l'image
3. `Luminance` = Puissance de la lumière émise par une image
4. `Correction gamma` => Permet de corriger la luminosité et le contraste d'une image par rapport à son support de diffusion

Contraste :

- `imadjust` (img, height, width, intensity) => où l'intensité on passe à 0,5 par exemple.
- Appliquer la matrice (filtre passe haut) :
  - `(0 -1 0; -1 5 -1; 0 -1 0)`

Repoussage :

- Appliquer la matrice :
  - `(-2 -1 0; -1 1 1; 0 1 2)`

```
img = imread('LENNA.BMP');
subplot(3,3,1);
imshow(img);
title('Original');

% amelioration de l'image - imadjust par défaut
subplot(3,3,2);
img_imadjust = imadjust(img);
imshow(img_imadjust);
title('imadjust image');

% Correction par égalisation de l'histogramme
subplot(3,3,3);
histeq(img);
title('Egalisation histogramme');

% amelioration du contraste - imadjust
subplot(3,3,4);
img_contrast_imadjust = imadjust(img,[],[],0.5);
imshow(img_contrast_imadjust);
title('imadjust contrast');

% amelioration du contraste via matrice
subplot(3,3,5);
matrice_contraste = [0 -1 0; -1 5 -1; 0 -1 0];
img_contrast_matrice = imfilter(img,matrice_contraste);
imshow(img_contrast_matrice);
title('matrice contraste');

% Repoussage via matrice
subplot(3,3,6);
matrice_repoussage = [-2 -1 0; -1 1 1; 0 1 2];
img_repoussage_matrice = imfilter(img,matrice_repoussage);
imshow(img_repoussage_matrice);
title('matrice repoussage');

% Contour - fspecial Sobel
subplot(3,3,7);
S = fspecial('sobel');
img_contour_sobel = imfilter(img,S);
imshow(img_contour_sobel);
title('fspecial Sobel - contour');

% Image flou via imgaussfilt
subplot(3,3,8);
img_flou = imgaussfilt(img,5);
imshow(img_flou);
title('Flou gaussien - imgaussfilt');
```

Original

imadjust image

Egalisation histogramme

imadjust contrast

matrice contraste

matrice repoussage

fspecial Sobel - contour

Flou gaussien - imgaussfilt