*Maxence PETITPIERRE Groupe B*

*Clément EYRAUD 6/12/16*

**TP Morphologie Mathématiques**

**des**

**Images Binaires**



**Lenna** (ou plus justement **Lena**) est un morceau de photo d'une [*playmate*](https://fr.wikipedia.org/wiki/Playmate) prise dans le numéro de novembre (miss novembre) [1972](https://fr.wikipedia.org/wiki/1972) du magazine [*Playboy*](https://fr.wikipedia.org/wiki/Playboy). Elle sert d'image de test pour les algorithmes de [traitement d'image](https://fr.wikipedia.org/wiki/Traitement_d%27image) et est devenue *de facto* un standard industriel et scientifique.

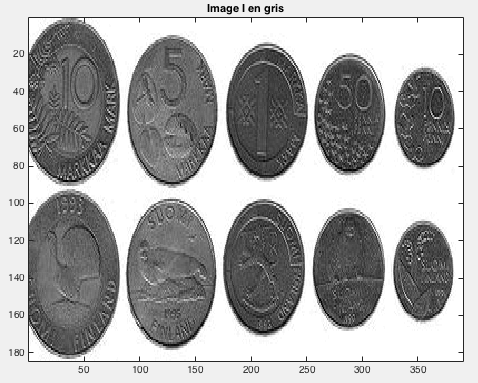
Pour ce TP de TSI, nous allons utiliser deux applications (granulométrie et recherche d’un motif) pour mettre en place les opérateurs de base de la morphologique mathématique (érosion, ouverture, dilatation etc ..).

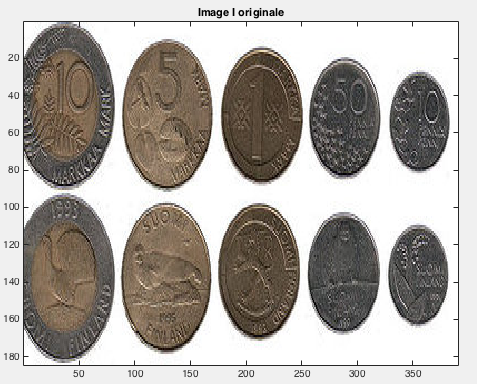
# I. Opérations de base sur une Image

## I.1 Lecture, transformation en niveau de gris et affichage

Dans cette partie, on lit l’image piece.png, on l’affiche puis on la transforme en niveau de gris. Le code nous est fournit et on le rappel ci-dessous pour conserver les même notations pendant tout le TP.

|  |
| --- |
| I = imread('piece.jpg');  figure(1), imagesc(I), title('Image I originale') ;  I=rgb2gray(I);  figure(2), imagesc(I), colormap ('gray'), title('Image I en gris'); |

On obtient les 2 figures suivantes :



*Figure 1 : Image I à traiter Figure 2 : Image I en niveau de gris*

## 

## I.2 Histogramme et seuillage

On réaliste l’histogramme des niveaux de gris :

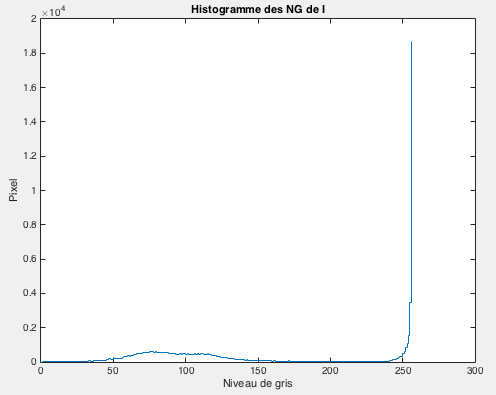
|  |
| --- |
| H=imhist(I);  figure(3), stairs(H), title('Histogramme des NG de I');  xlabel('Niveau de gris'), ylabel('Pixel'); |

L'histogramme d'une image donne sa répartition en niveau de gris. Pour une image qui possède 256 niveaux de gris, l'histogramme représente le nombre de pixel en fonction du niveau de gris dans l'image.

Le niveau de gris 0 correspond au noir et le niveau 255 au blanc.

A partir de l'histogramme obtenu on va pouvoir fixer un seuil au dessus duquel on va garder certains niveaux de gris. D'après la figure 2, on observe des pièces de différentes teintes grises (qui correspond à la zone de 50 à 125 sur l'histogramme) représentées sur un fond blanc (fort pic à 255 sur l'histogramme).

Voici l’histogramme obtenu :

*Figure 3 : Histogramme en niveau de gris de l’image I*

Ensuite on calcul la valeur du seuil pour ensuite seuiller l’image avec. On calcul le complémentaire de cette image pour l’afficher et comparer à l’originale. On finira par calculer la taille de l’image.

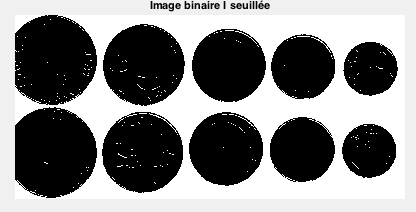
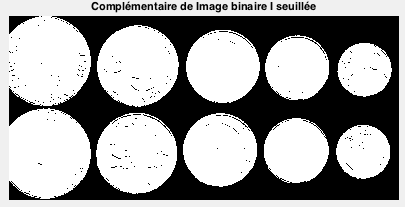
|  |
| --- |
| %Calcul du seuil  s=graythresh(I);  fprintf('\nValeur de seuil : %f',s);  %Situer le seuil sur l'histogramme  Seuil=s\*255  Ib=im2bw(I,s);  figure(4); imshow(Ib), title('Image binaire I seuillée');  %Complémentaire de l'image seuillÈe  Ib = imcomplement(Ib);  figure(5); imshow(Ib), title('Complémentaire de Image binaire I seuillée');  %Taille de l'image  [G, W]=size(Ib);  fprintf('\nTaille image : %d par %d\n',G,W); |

La fonction « graythresh » retourne le seuil d’une image binaire. Ce seuil calculé est donné comme un rapport du niveau de gris seuil sur le nombre total de niveau de gris.

Si l’on multiplie le seuil s par le nombre de niveaux, c’est à dire 255, on obtient la valeur du seuil en niveau de gris que l’on peut situer sur l’histogramme. On obtient :

|  |
| --- |
| Valeur de seuil : 0.666667  Seuil =  170  Taille image : 184 par 390 |

Si l’on situe le niveau de gris 170 sur la figure 3, on observe bien qu’il permet de séparer la zone de 50 à environ 150 du pic à 255. On obtient les deux figures à suivante :



*Figure 4 : Image seuillée Matlab  Figure 5 : Complémentaire de l’image seuillée*

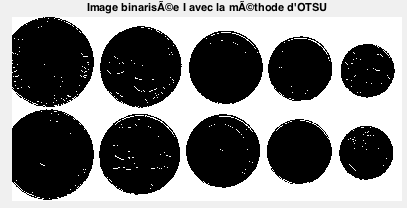
On nous demande par la suite de d’écrire une méthode de seuillage automatique par maximisation de la variance interclasse en ne travaillant que sur l’histogramme. Cette méthode correspond à ce que nous avons vu au TP1. Il existe aussi une autre méthode connue, la méthode d’OTSU. Cet algorithme suppose alors que l'image à binariser ne contient que deux classes, les objets et l'arrière-plan. L'algorithme itératif calcule alors le seuil optimal T qui sépare ces deux classes afin que la variance intra-classe soit minimale et que la variance inter-classe soit maximale.

(source : <https://sites.google.com/site/lizantchristopher/services/binarisation-1> ).

On obtient donc le code suivant :

|  |
| --- |
| %Methode de seuillage automatique : OTSU  NB\_total=G\*W;  Proba=H/NB\_total;  for i=1:255    %Calcul des proba des classes  Prob1=Proba(1:i);  Prob2=Proba(i+1:255);  P1=sum(Prob1);  P2=sum(Prob2);    % Calcul de la moyenne des classes %  n1 = 0:i-1;  n2 = i:254;  n1=n1';  n2=n2';  M1 = sum( n1.\*Prob1)/P1;  M2 = sum( n2.\*Prob2)/P2;  % Calcul de la variance des classes %  Var1 = sum(((n1 - M1).^2).\*Prob1);  Var2 = sum(((n2 - M2).^2).\*Prob2);    VarianceIntraClasse(i) = Var1 + Var2;  end    % Détermination du seuil %  [Val,Indice] = min(VarianceIntraClasse(1:255));  seuil = (Indice-1) ;  seuil/255  % BINARISATION %  BW1 = im2bw(I,seuil/255);  BW1 = BW1\*255;  % Affichage de l'image binarisée %  figure(6);  imshow(BW1);  title('Image binarisée I avec la méthode d''OTSU'); |

On peut donc comparer la figure obtenue avec la méthode de Matlab et voire que ces deux images sont identiques :



Seuil obtenu :

*Figure 6 : Image seuillée avec OTSU*

seuil =

166

ans =

0.6510

On obtient un seuil de 166, soit relativement proche de celui retourné par « graythresh ».

# II Morphologie mathématique

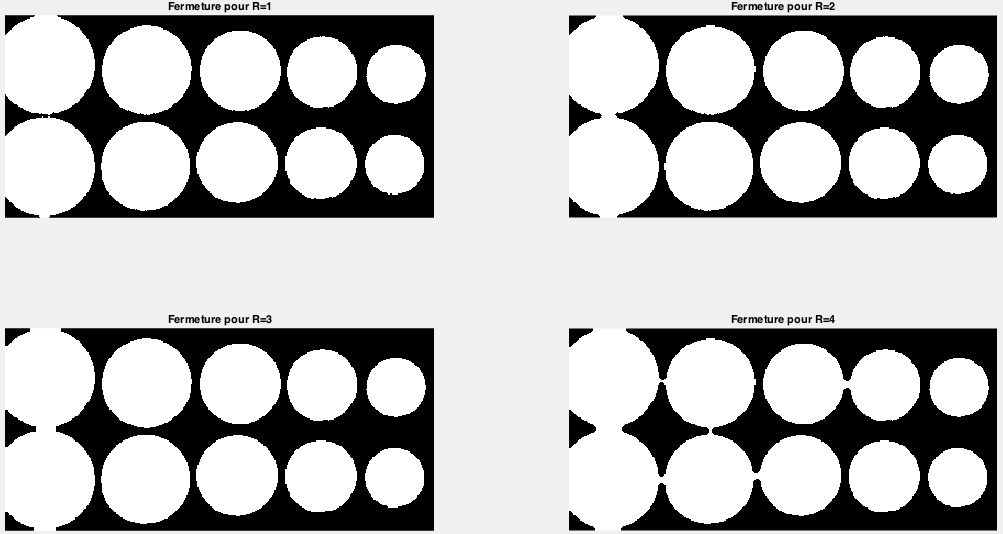
## II.1 Granulométrie

## Dans cette partie, on cherche à obtenir la courbe de granulométrie des pièces, c'est-à-dire le nombre de pièces en fonction de leur taille. Avant cela, pour ne pas fausser la statistique, on procède à quelques prétraitements afin de « nettoyer » l’image.

## Tout d’abord nous travaillerons uniquement avec le complémentaire de l’image seuillée car pour ce traitement d’image, les objets (les pièces) doivent être en blancs. Le premier prétraitement à faire est de « boucher » les trous des objets par une opération de fermeture. Pour cette opération, nous avons besoin d’un élément structurant. Ce dernier doit prendre la forme de l’élément principal de l’image. Ici nous avons des pièces rondes, nous prenons donc comme élément structurant un disque. Pour définir cet élément nous devons spécifier un rayon en pixel. Pour déterminer le meilleur rayon de correction nous réalisons plusieurs fermetures avec différents rayons de disque.

|  |
| --- |
| R=[1 2 3 4];  figure(7);  for m=1:4  SE=strel('disk',R(m));  %Doit prendre la forme de l'element principal de l'image, meme motif  I\_sansTrou=imclose(Ib,SE);  subplot(2,2,m);  imshow(I\_sansTrou);  title(['Fermeture pour R=',num2str(R(m))]);  end |

On obtient les différents résultats suivants :



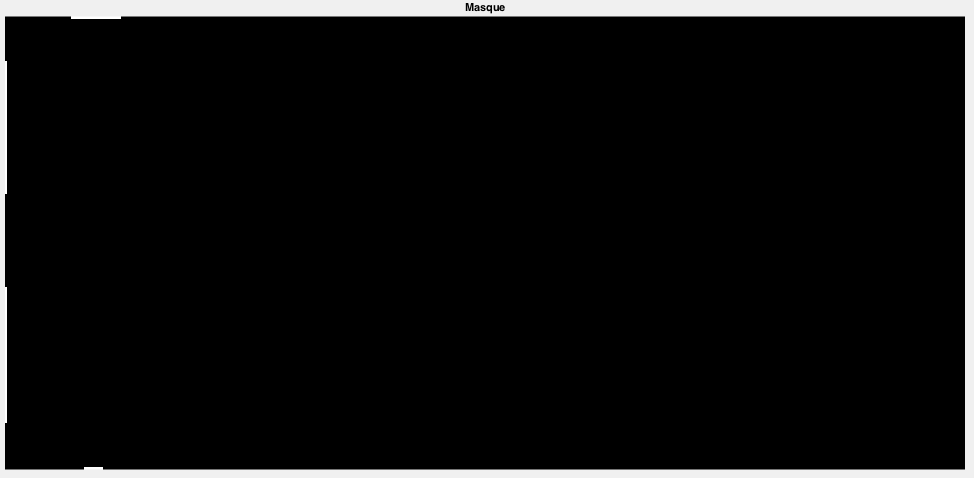
*Figure 7 : Fermeture de l’image pour différents rayons*

Finalement, le meilleur résultat est obtenu pour le rayon de 1 pixel. C’est donc cette valeur que nous conservons pour la suite du TP. Pour les autres rayons on observe un recouvrement d’objets.

Sur l’image obtenue par fermeture par un disque de rayon R=1, on observe que les deux grosses pièces sur la gauche de l’image touchent le bord. Nous allons donc les supprimer pour éviter de mal estimer leur taille sur la courbe granulométrique.

Premièrement nous définissons un marqueur. Ce marqueur sera définit de la taille de l’image et composé de pixels mis à 1 sur tous les bords de l’image et nuls ailleurs. Ensuite on réalise l’intersection de ce masque avec l’image résultante de la fermeture par un disque de rayon 1.

|  |
| --- |
| %Fermeture  figure(7);  SE=strel('disk',1);  I\_st=imclose(Ib,SE);  imshow(I\_st); title('Fermeture pour R=1');    masque=zeros(G,W);  masque(1,:)=1; masque(:,1)=1;  masque(G,:)=1; masque(:,W)=1;  Im = masque .\* I\_st; %marqueur  figure(8); imshow(Im); title('Masque'); |

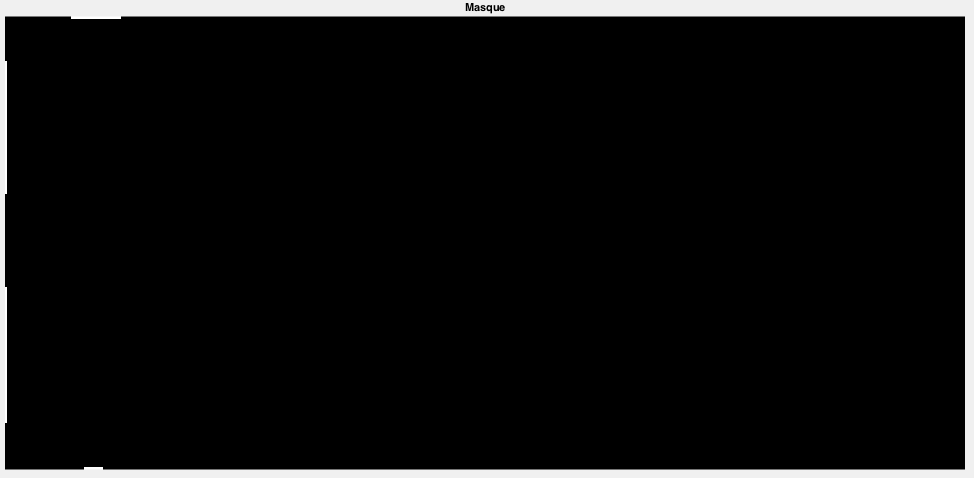


*Figure 8 : Masque des objets à reconstituer et supprimer*

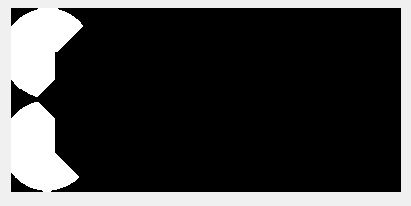
Sur la figure 8 ci-dessus on est en mesure de bien identifier à quels endroits les objets touchent les bords. Pour pouvoir les supprimer de l’image initiale, nous allons les reconstituer par une opération de dilatation du marqueur. Cette dilatation utilise le même élément structurant que la fermeture : un disque de rayon un pixel. A chaque dilatation, on réalise de nouveau l’intersection du marqueur avec l’image initiale de manière à ne conserver sur le marqueur que les objets à supprimer. On réitère cette opération dans un « while » et lorsque la différence entre deux marqueurs consécutifs est nulle, cela signifie que nous avons totalement reconstitué les objets à supprimer.

Pour achever cette opération, on soustrait le marqueur final, contenant les pièces à supprimer, à l’image initiale.

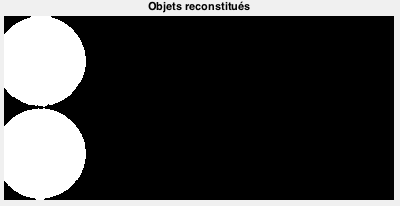
|  |
| --- |
| Im =imdilate(Im,SE).\*I\_st; %On dilate le marqueur  Im\_1=masque;  while(sum(sum(abs((Im - Im\_1)))) ~= 0)  %Im\_1 est le marqueur prÈcÈdent  Im\_1=Im;  Im =imdilate(Im,SE);  %Im est le marqueur courant  Im=Im .\* I\_st;  end  figure(9); imshow(Im); title('Objets reconstitués');  If=I\_st-Im;  figure (10); imshow(If); title('Image avec objets supprimÈes'); |

Reprenons la chronologie de la méthode :

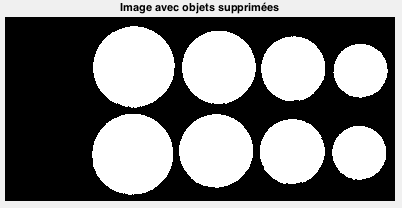
*Figure 9.1 : Masque initial pour reconstruction des objets*

**

*Figure 9.2 : Masque en cours de reconstruction des objets*

**

*Figure 9.3 : Masque final contenant les objets à supprimer*

**

*Figure 9.4 : Image dont on a supprimé les objets touchant les bords*

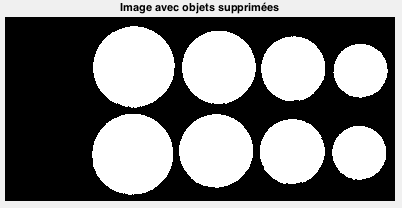
Arrivé à cette étape là, la dernière image obtenue va pouvoir être utilisée pour établir la courbe granulométrique.

Pour établir cette courbe, nous allons nous aider d’une fonction Matlab : «bweuler». Cette fonction retourne le nombre de composantes (d’objets) de l’image passée en paramètre. La méthode va être la suivante :

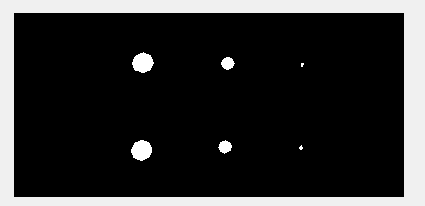
1. Calcul du nombre initial de composantes, d’après l’image : 8.
2. Tant que le nombre de composantes connexes de l’image n’est pas nul, nous allons éroder l’image par un élément structurant disque dont le rayon augmentera à chaque réitération de 1 pixel. A chaque incrémentation du rayon r, on recalcule le nombre de composantes que le soustrait au nombre initial et que l’on stocke dans un vecteur G.
3. Lorsque l’on a érodé tous les objets, la boucle se termine et on affiche la courbe granulométrique.

Voici le programme exécuté :

|  |
| --- |
| %Courbe granulom√©trique  % calcul du nbr de composante initales connexes dans une image binaire  N\_depart=bweuler(If);  r=0; %rayon du disque stucturant  N=N\_depart;  while(N~= 0)  r=r+1;  ES=strel('disk',r);  Ig=imerode(If,ES);  S=bweuler(Ig);  G(r)=N\_depart-S;  N=S;  %figure(11); imshow(Ig); pause(0.001);  end  figure(11);  plot(G);  ylabel('Nombre de piece');xlabel('Pixel de rayon');  title('Courbe granulomÈtrique'); |

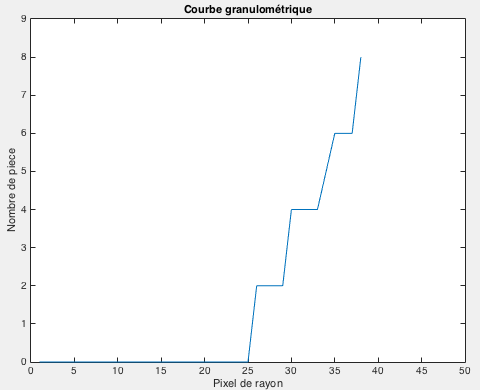
Et le résultat en image :

*Figure 10.1 : Image initiale*

**

*Figure 10.2 : Image en cours d’érosion*

Lorsque tous les objets ont été supprimés, l’érosion se termine et on affiche la courbe granulométrique. Nous n’avons pas besoin de spécifier d’axe des abscisses pour G car il est calculé à chaque itération. Or à chaque itération on augmente aussi le nombre de pixel. Donc lorsque l’on « plot(G) », celui-ci s’affiche en fonction des indices en commencant par 1 ce qui correspond exactement à l’échelle des pixels car la première valeur de G avait été calculée pour r=1.

**

*Figure 11 : Courbe granulométrique*

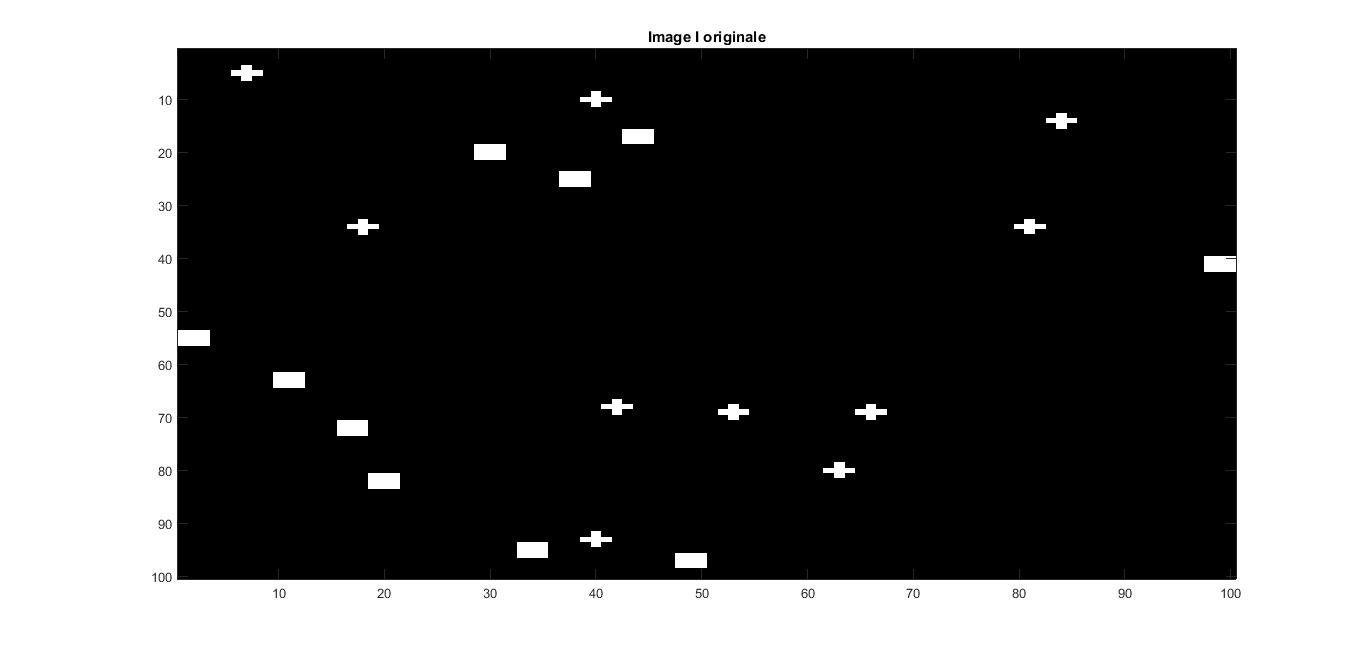
Sur cette courbe on observe qu’il n’y aucun objet dont le rayon est inférieur à 25 pixels. De plus, on voit bien qu’il y a au maximum huit objets sur la figure et qu’ils sont deux par deux de même taille. Ces résultats sont en concordance avec ce que l’on pourrait conjecturer en regardant l’image.

## 

## 

## II.2 Recherche d’un motif

Dans cette partie nous allons utiliser la transformée tout-ou-rien (Hit or miss). Nous allons tout d’abord afficher la première image (voir code 2.2.1) pour en faire l’étude.



|  |
| --- |
|  |

*Image « ICroix1.png »*

On obtient une image formée de croix et de carrés. Lorsqu’on s’intéresse à la taille de cette croix on peut voir qu’elle est représentée sous cette forme :

|  |
| --- |
|  |

On définit donc notre élément structurant, noté SE1 (qui inclut objet), comme une croix en créant la matrice 3x3 ci-dessous :

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

*Elément structurant SE1*

On définit ensuite le complémentaire de cette élément structurant, noté SE2 (qui inclut le fond), en créant la matrice 3x3 sous cette forme la :

|  |  |  |
| --- | --- | --- |
| 1 | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

*Elément structurant SE2*

Ensuite nous exécutons la transformée Tout-ou-rien grâce à la fonction « bwhitmiss. » Nous obtenons cette figure :

|  |
| --- |
| C:\Users\Clem\Documents\4A\TSI\TP2\figure2.jpg |

*Positionnement des croix figure ICroix1.png*

Une fois que nous avons définit nos marqueurs, nous pouvons utiliser la méthode vue dans la partie 2.1 qui consiste à les dilater pour ensuite faire une intersection avec notre image d’origine. En répétant ce processus plusieurs fois nous aurons l’image d’origine uniquement avec des croix.

Voici le résultat :

|  |
| --- |
| C:\Users\Clem\Documents\4A\TSI\TP2\figure3.jpg |

*Reconstitution de l’image d’origine uniquement avec des croix pour la figure ICroix1.png*

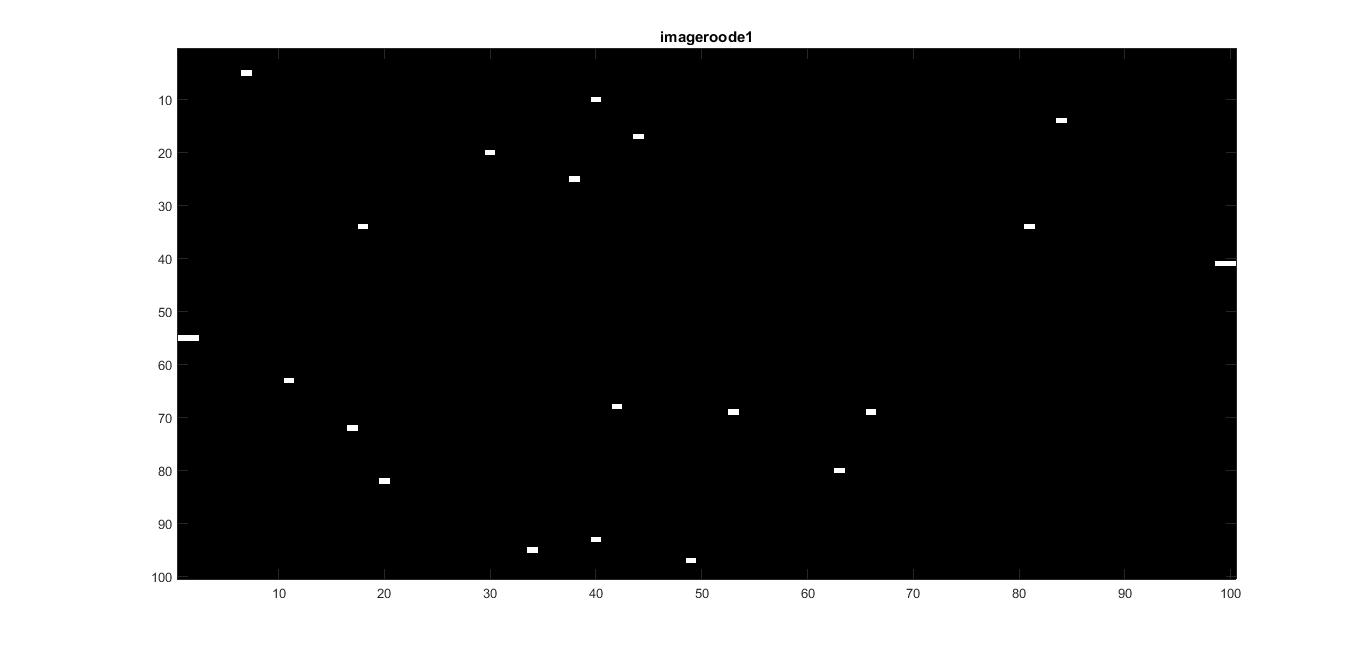
Voici le code utilisé dans la partie 2.2.1 :

|  |
| --- |
| clear all;close all;clc;  R=1 % rayon de longueur 1  BW1 = imread('Icroix1.png'); %on lit l'image  figure(1), imagesc(BW1),colormap('gray'), title('Image I originale') ; %on affiche celle-ci    SE1= [0,1,0;1,1,1;0,1,0]; % creation de l'élément structurant en forme de croix  SE2=~SE1; %element structurant complémentaire à SE1    % partie 1  BW2 = bwhitmiss(BW1,SE1,SE2); % realisation de la transformee tout-ou-rien  figure(2), imagesc(BW2),colormap('gray'), title('position des croix') ; % affichage de celle-ci  [G, W]=size(BW2); %dimensionnement de BW2  BW2\_1=zeros(G,W); % creation d'un matrice de meme taille que BW2  SE=strel('disk',R);% element structurant pour la reconstruction d'etoile    while(sum(sum(abs((BW2 - BW2\_1)))) ~= 0) %verification de si l'image est toujours différente de celle d'avant  BW2\_1=BW2;  BW2 =imdilate(BW2,SE); %dilatation de bw2  BW2=BW2 .\* BW1; %intersection avec l'image d'origine  end  figure(3), imagesc(BW2),colormap('gray'), title('image d origine seulement avec les croix'); % affichage |

*Code première parti 2.2.1*

Maintenant que nous avons vu le fonctionnement de la fonction « bwhitmiss », nous allons la recréer grâce à des érosions. En annexe 1 on rappel le fonctionnement de la transformée tout-ou-rien. Dans un premier temps on érode BW1, soit l’image d’origine, avec l’élément structurant SE1.

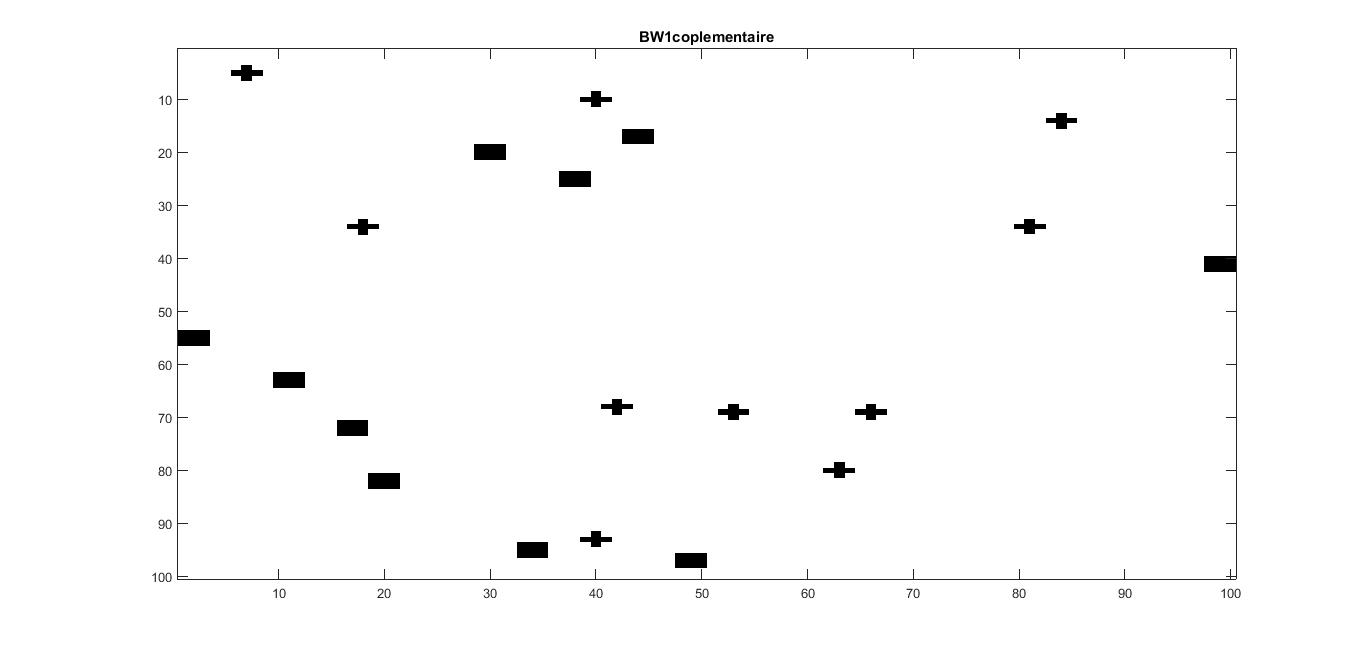
On obtient la figure suivante :



|  |
| --- |
|  |

*Erosion de BW1 avec comme élément structurant SE1 pour la figure ICroix1.png*

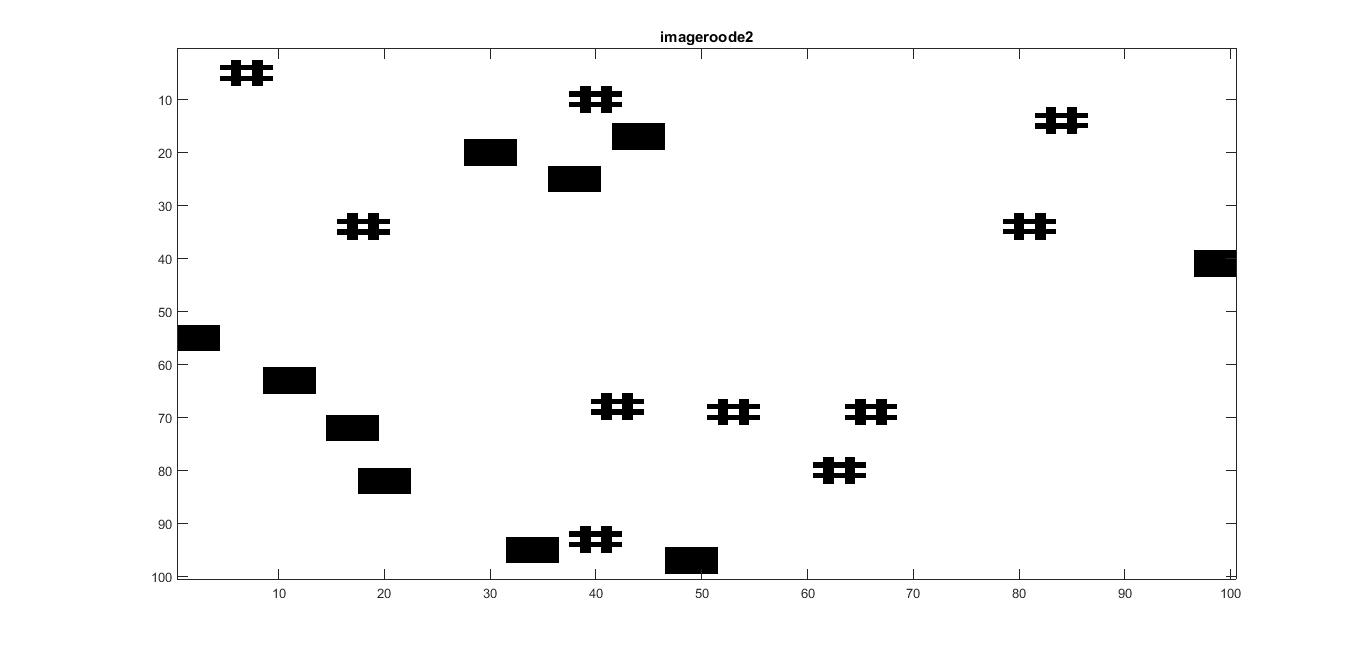
L’image résultante indique les positions des carrés et des croix comme nous l’avions conjecturé dans l’annexe 1. Ceci est cohérent car l’élément structurant est une croix et donc les carrés et les croix contiennent l’élément structurant.

Dans un second temps, nous allons faire une érosion sur le complémentaire de BW1 avec l’élément structurant SE2. Nous obtenons donc les graphiques suivants :

|  |
| --- |
|  |

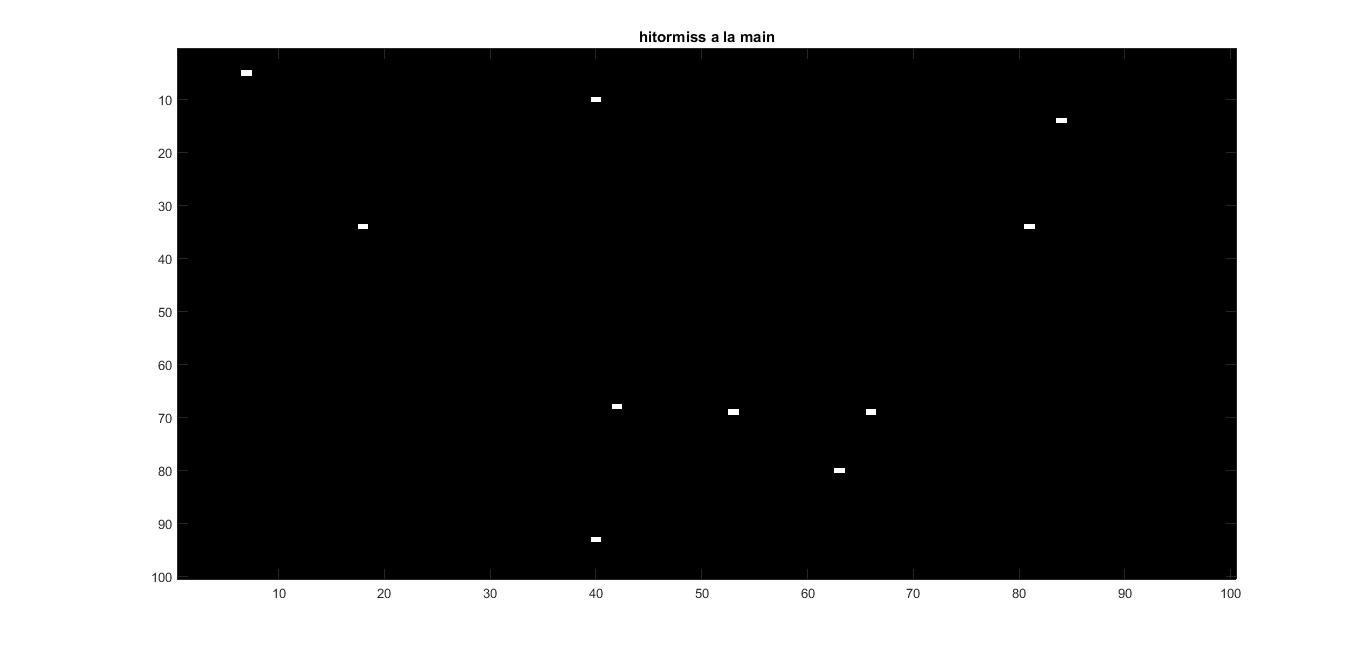
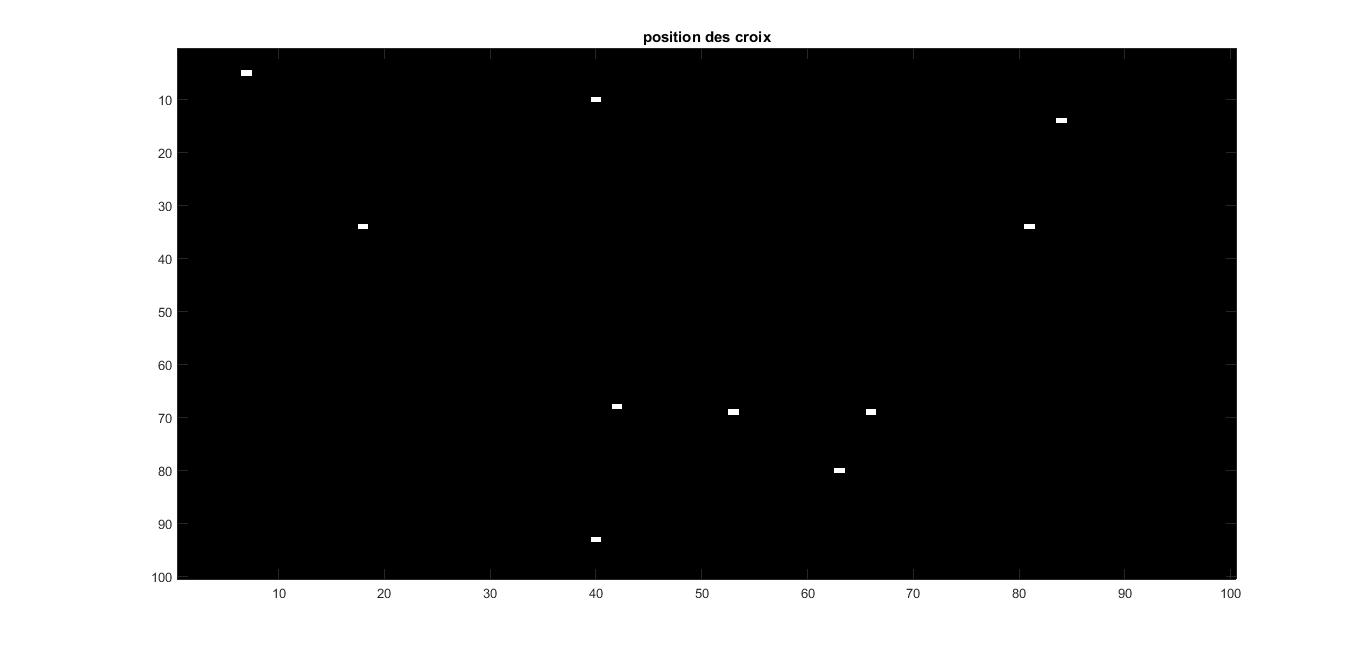
*Le complémentaire de BW1 pour la figure ICroix1.png*

|  |
| --- |
|  |

*Deuxième érosion de BW1 complémentaire avec l’élément structurant SE2 sur l’image ICroix1.png*

On remarque que tous les carrés sont à 0 et le centre de chaque croix est 1. Cela confirme les résultats de l’annexe1. Les carrés étant tous à 0, ils ne contiennent pas l’élément structurant SE2 contrairement aux croix qui quant à elles le contiennent toujours.

Pour finir, il ne reste plus qu’à faire l’intersection entre nos deux érosions ce qui, en binaire, revient à multiplier scalairement les deux images après érosion.



*Positionnement des croix figure ICroix1.png par la fonction bwhitmiss*

*Intersection des deux érosions pour la figure ICroix1.png*

On peut en conclure que le résultat obtenu est identique à celui retourné par la fonction « bwhitmiss » de Matlab.

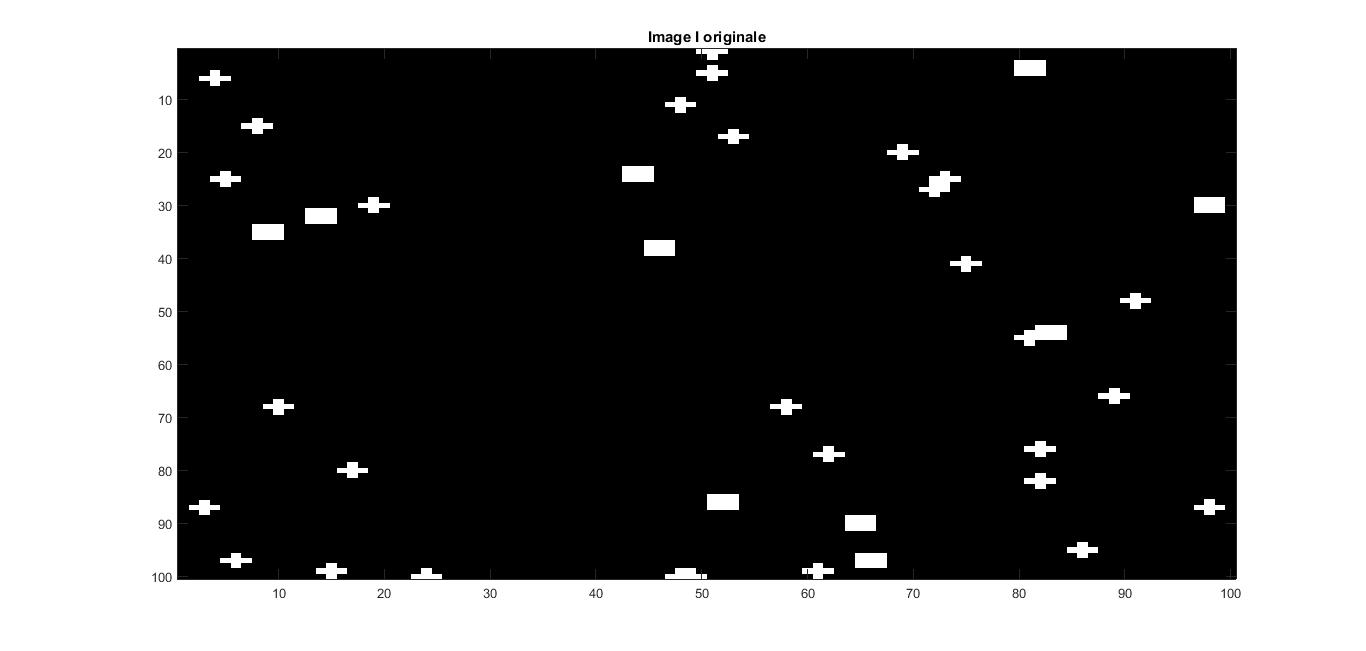
Remarque : Dans cet exemple, nous aurions pu directement tout faire avec une seule érosion car dans cette image, il n’existe pas d’autre motif que des croix ou des carrés. Mais dans une image où il existe que des moitiés de croix, cette méthode n’aurait pas fonctionné.

Voici le code pour Matlab pour cette partie :

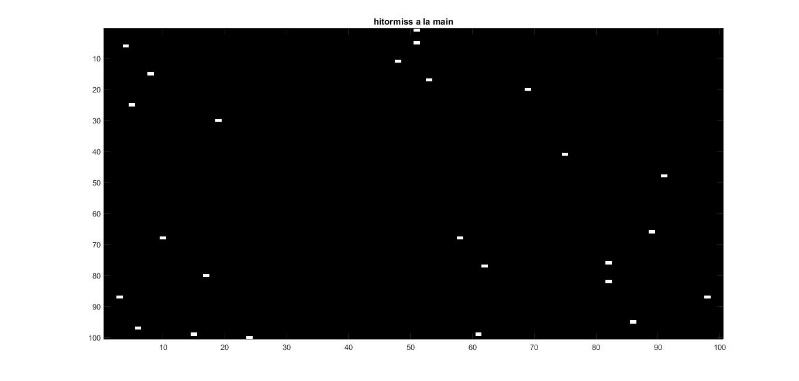
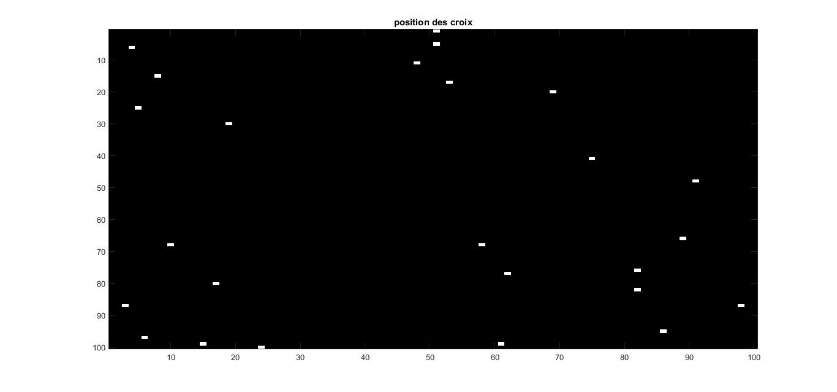
|  |
| --- |
| clear all;close all;clc;  BW1 = imread('Icroix1.png'); %on lit l'image  figure(1), imagesc(BW1),colormap('gray'), title('Image I originale') ; %on affiche celle-ci    SE1= [0,1,0;1,1,1;0,1,0]; % creation de l'élément structurant en forme de croix  SE2=~SE1; %element structurant complémentaire à SE1  % deuxieme partie code a la main  imageerode1=imerode(BW1,SE1);%on fait une erosion de BW1 avec l'element structurant SE1  figure(4), imagesc(imageerode1),colormap('gray'), title('imageroode1'); %on affiche celle-ci    BW1complementaire=imcomplement(BW1);% on prend l'image complementaire de BW1  figure(5), imagesc(BW1complementaire),colormap('gray'), title('BW1coplementaire');%on affiche celle-ci    imageerode2=imerode(BW1complementaire,SE2);%on fait une erosion de BW1complementaire avec l'element structurant SE2  figure(6), imagesc(imageerode2),colormap('gray'), title('imageroode2');%on affiche celle-ci    BW2=imageerode1.\*imageerode2;% On fait l'intersection des deux images apres erosions  figure(7), imagesc(BW2),colormap('gray'), title('hitormiss a la main');%on affiche celle-ci |

*Code Matlab de la partie 2.2.2*

Dans cette dernière partie nous tester ce même programme avec l’image ICroix2.png.

Tout d’abord voici l’image ICroix2.png :

*Image originale de ICroix2.png*

Nous avons obtenons les deux résultats suivant :

*Positionnement des croix grâce à la fonction bwhitmiss*

*Positionnement des croix grâce à notre fonction*

Nous pouvons conclure une nouvelle fois que le résultat est identique entre les deux méthodes.

Remarque : Nous observons quelques défauts dans ce Hit or miss, tels que les moitiés de croix en bordure. Ceux-ci pourraient être corrigés grâce au traitement spécifique des objets situés sur les bordures.

# Conclusion :

Dans ce TP nous avons pu mettre en pratique différents opérateurs de bases de la morphologie mathématique.

La granulométrie nous a permis de mettre en place le « nettoyage » d’une image par une opération de fermeture. S’en est suivit des opérations de dilatations sur un masque afin de détecter et de reconstruire des objets touchant les bords. Pour finir on soustrait ces objets à l’image nettoyée pour ensuite la traiter et ainsi obtenir sa courbe granulométrique.

La recherche de motif nous a permis de mettre en place et comprendre la fonction « bwhitmiss » de Matlab. Cette fonction réalise la transformée tout ou rien de l’image à l’aide d’un élément structurant et de son complémentaire pour notre cas. Ces éléments structurants sont les motifs représentés dans l’image. Par une méthode dilatation et d’intersection successives de la transformée tout-ou-rien avec l’image initiale, nous sommes en mesure de reconstituer une image contenant uniquement les motifs correspondant à l’élément structurant.

Nous avons par la suite voulu recréer la fonction « bwhitmiss » (Annexe 1) afin de bien comprendre son fonctionnement.

Certains opérateurs morphologiques n’ont pas été mis place dans ce TP. C’est le cas par exemple de l’ouverture. Nous aurions pu utiliser l’ouverture lors de la reconstitution des pièces à supprimer. Plutôt que de faire une fermeture sur le masque, une ouverture sur le complémentaire du masque aurait fournit le même résultat.

# Annexe1

Voici A notre image d’origine, on peut y voir un carré 3x3, une croix et 1 point.

Première étape :

L’image A

On exécute donc une érosion sur l’image A avec l’élément structurant B, une croix.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

L’élément structurant B

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Lorsqu’on fait l’érosion on obtient donc ceci :

Erosion 1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Seul le carré et la croix contiennent l’élément structurant donc le résultat est cohérent

Deuxième étape ;

On réalise l’érosion du complémentaire de A avec le complémentaire de B

Complémentaire A

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 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 | 1 |

L’élément structurant complémentaire de B

|  |  |  |
| --- | --- | --- |
| 1 | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Lorsqu’on réalise cette érosion on obtient :

Erosion 2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

On a gardé tous nos bords 1 pour l’exemple car c’est un cas particulier.

En réalité le résultat n’est pas réellement cette figure mais notre matrice n’étant pas assez grande, nous avons dû nous adapter. En réalité, les carrés en 3x3 donnent des carrés de zéro en 5x5. Les croix et les points donneront des schémas différents si les bords n’étaient fixés à 1. Ce qui est important c’est que la croix a, en son centre, 1 et que le point soit aussi représenté par un 1.

Troisième étape :

On réalise l’intersection entre nos deux matrices pour obtenir finalement tous les positionnements des croix dans cette matrice.

Erosion 1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Erosion 2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

L’intersection donne donc bien le centre de la croix.

Intersection

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |