Exercice Chapitre 4 – Traitement d'images binaires par Morphologie Mathématique

L'analyse par morphologie mathématique vise à modifier la structure et la forme des objets de l'image, par exemple, pour séparer les objets, les discriminer en fonction de leur taille, remplir les trous, …

- ♦ Rappels :

La transformation morphologique modifie la valeur d'un pixel de l'image en fonction de la valeur de ses voisins. Pour cela, on utilise un ***élément structurant***, qui est un masque binaire. Il permet de prendre en compte le voisinage du pixel.
Exemple d'un élément structurant à 4-connexité (le centre est marqué par un cercle) :

0	1	0
1	**1**	1
0	1	0

Les traitements morphologiques sont définis à partir de deux opérations de base qui sont l'***érosion*** et la ***dilatation***.

Définition :

Soient « S » le support de l'image, et « p » l'affixe d'un point P de cette image :

- L'érosion d'une forme X par un élément structurant B est notée « $\mathbf{X \ominus B}$ ». Elle est définie par : $\mathbf{X \ominus B = \{ p \in S,}$ **tel que** $\mathbf{B_p \subseteq X\}}$;
- La dilatation d'une forme X par un élément structurant B est notée « $\mathbf{X \oplus B}$ ». Elle est définie par : $\mathbf{X \oplus B = \{ p \in S,}$ **tel que** $\mathbf{\overline{B_p} \cap X \neq \varnothing\}}$.
- L'érosion et la dilatation sont duales par rapport à la complémentation :

$$X \oplus B = (X^C \ominus B)^C$$

Avec :
- X^C complémentaire de X ;
- $\overline{B}$ élément symétrique de B (i.e. $\overline{B}(i, j) = B(-i, -j)$) ;
- B_p élément transposé de B par le vecteur d'affixe p.

Vous allez utiliser Matlab pour réaliser des filtrages morphologiques. Avant de commencer, ouvrez le fenêtre d'aide de Matlab (commande ***helpwin***) et accédez à la ***Toolbox Image Processing***. Vous aurez ainsi une description détaillée pour chacune des fonctions utilisées dans cet exercice.

1. Erosion et dilatation d'une image binaire

Mettez à jour la liste des chemins dans le ***path browser***. Ouvrez un nouveau fichier « ***M-File*** » dans lequel vous allez saisir vos commandes et dont chaque ligne (se terminant par un « *;* ») sera ensuite interprétée. Chargez l'image *CIRCUIT.TIF* avec ***imread***.

1.1 – Binarisez l'image en utilisant la méthode de votre choix (cf. exercice binarisation du chapitre 2).

1.2 – Par convention le « fond » de l'image binaire doit être noir (à « 0 »), et la forme de l'objet doit être blanche (à « 1 »). Donc, vous pouvez inverser l'image binaire avec l'opérateur « ~ » (**I = ~ I**).

1.3 – Choisissez un élément structurant. Vous pouvez l'écrire directement sous la forme d'une matrice binaire (ex. SE = ***ones***(3), SE = [0 1 0;1 1 1;0 1 0], …). Erodez les formes de l'image binaire avec ***imerode***. Définissez, avec « l'aide de Matlab » (commande ***helpwin***), cette opération.

1.4 – De même, dilatez les formes de l'image binaire avec ***imdilate*** et définissez cette opération avec l'aide de Matlab.

1.5 – Erodez ou dilatez l'image binaire en jouant avec la forme de l'élément structurant. Observez notamment les résultats lorsque l'élément structurant n'est pas symétrique par rapport à son origine.

1.6 – On veut observer la relation de dualité entre l'érosion et la dilatation. Pour cela, créez une image binaire résultat de l'érosion du fond, et une seconde résultat de la dilatation des formes. Comparez ces deux images et concluez.

2. Ouverture (érosion puis dilatation) d'une image binaire bruitée

2.1 – Chargez l'image *CIRCUIT.TIF* et ajoutez lui du bruit avec la fonction ***randn***. Cette fonction permet de générer une matrice d'un bruit gaussien N(0, 1) que vous pouvez amplifier et à ajouter à votre image originale

Remarques :
- les dimensions de la matrice du bruit et de l'image originale doivent être les mêmes ;
- pensez à convertir vos données pour les opérations effectuées (fonctions ***double*** et ***uint8***).

2.2 – Binarisez l'image originale et l'image bruitée, puis comparez les.

2.3 – Choisissez un élément structurant symétrique. Faites l'érosion puis la dilatation de l'image binaire bruitée. La composée de ces deux fonctions s'appelle l'***ouverture*** de l'image. Quel est l'intérêt ?

2.4 – Que ce passe t-il si, cette fois, l'élément structurant n'est pas symétrique ? Proposez et testez une méthode qui garantisse alors, la reconstruction des objets à l'identique. Une piste : il faut penser à faire l'érosion et la dilatation avec un élément structurant différent.

3. Fermeture puis ouverture d'une image binaire

3.1 – Chargez l'image en niveaux de gris *PEARLITE.TIF*. Binarisez et inversez l'image.

3.2 – Faites la fermeture (***imclose***) de cette image binaire inverse. Définissez cette opération (cf. aide Matlab). Pour générer l'ES vous pouvez utiliser la fonction ***strel*** (cette fonction génère des « objets élément structurant »).

3.3 – Effectuez l'ouverture (***imopen***) de l'image binaire précédente. Définissez cette opération.

3.4 – Analysez la chaîne complète des opérations.

Correction de l'exercice : Morphologie Mathématique pour images binaires

1.1 – Voici un exemple de solution pour binariser l'image *CIRCUIT.TIF* :

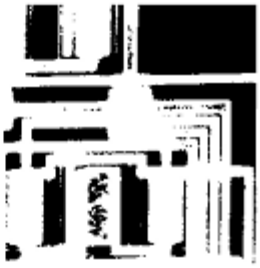
```
I = imread('CIRCUIT.TIF');
seuil = graythresh(I) % recherche du seuil avec la méthode d'Otsu
Ib = im2bw(I,seuil); % binarisation
```

1.2 – Sous Matlab, l'opérateur « ~ » correspond au « non logique ». Vous pouvez donc inverser une image binaire avec la commande : Ibi=~Ib. Pour observer l'opération effectuée, tapez :

```
figure(1)
imshow(Ib)
Ibi =~ Ib;
figure(2)
imshow(Ibi)
```

Ib

Ibi

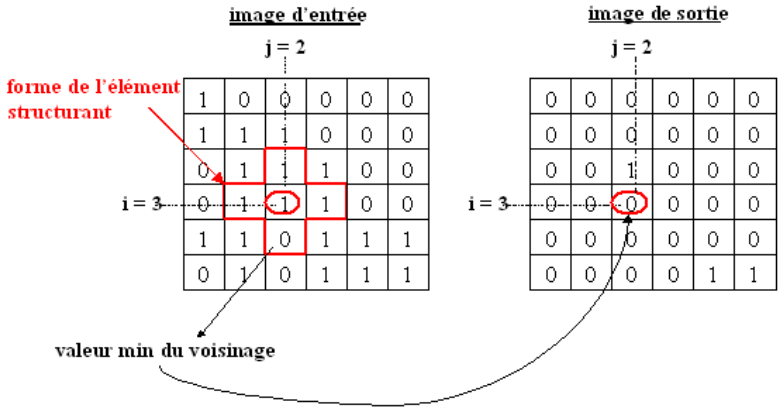


1.3 – on considère, par exemple, l'élément structurant suivant : $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

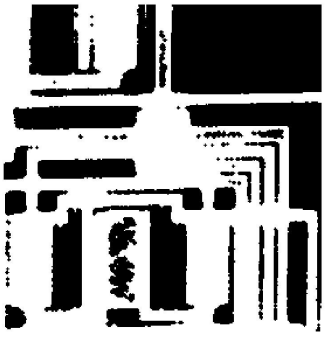
Pour éroder l'image binaire Ibi et afficher le résultat :

```
SE = [0 1 0;1 1 1;0 1 0] ; % élément structurant
Ier = imerode(Ibi,SE) ;
figure(3)
imshow(Ier)
```

L'élément structurant permet de définir un voisinage pour chacun des pixels de l'image d'origine : les pixels voisins sont ceux à 1 au sein de l'élément structurant. On balaie ensuite l'ensemble des pixels de l'image en leur appliquant l'élément structurant. La valeur d'un pixel après érosion est alors définie comme étant la valeur minimale de tous les pixels dans son voisinage. Pour une image binaire, si l'un des pixels du voisinage est à 0, la valeur de sortie du pixel est alors automatiquement 0.
La figure ci-dessous présente un exemple d'érosion, on détaille le cas du pixel d'affixe (3, 2), avec un élément structurant à 4-connexité.



Voici l'image obtenue après érosion :

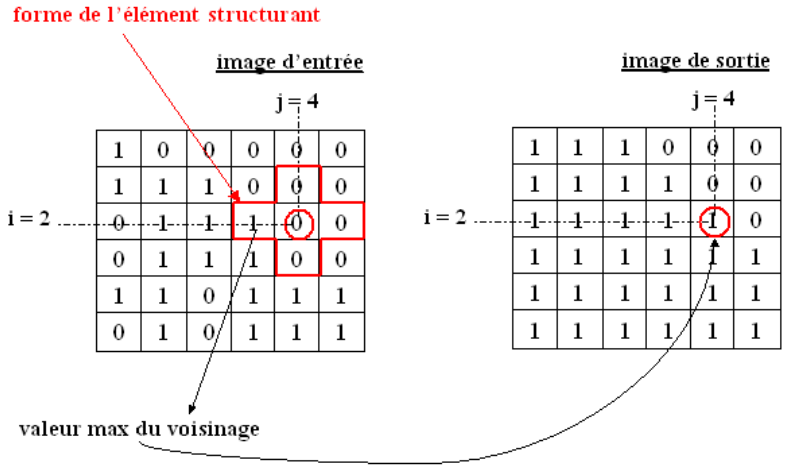


L'érosion a éliminé les pixels isolés sur le fond et a érodé le contour des objets de l'image *CIRCUIT*.

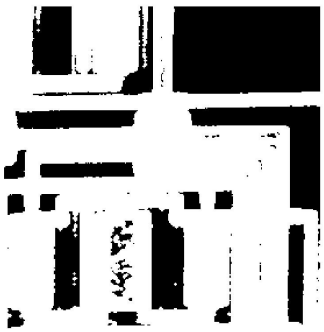
1.4 – Pour dilater l'image binaire Ibi et afficher le résultat :

```
Idi = imdilate(Ibi,SE) ;
figure(4)
imshow(Idi)
```

La valeur de sortie d'un pixel est la valeur maximale parmi tous les pixels compris dans le voisinage. Pour une image binaire, si l'un des pixel du voisinage est à 1, la valeur de sortie du pixel est 1. La figure ci-dessous présente un exemple de dilatation, et détaille le cas du pixel d'affixe (2,4), avec un élément structurant à 4-connexité.



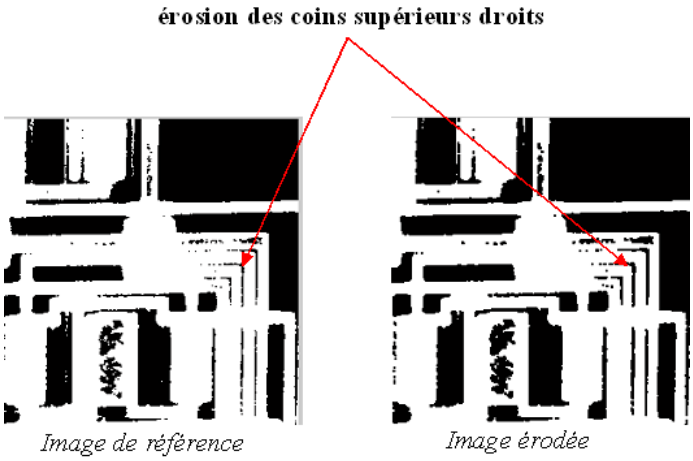
Voici l'image obtenue après dilatation :



La dilatation élimine les trous isolés dans les objets et dilate les bords des objets.

1.5 – On érode l'image avec l'élément structurant suivant : $\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

Dans ce cas, l'élément structurant n'est pas symétrique. Cette érosion tronque les coins supérieurs droits des objets.



Les formes érodées et dilatées dans les objets dépendent donc fortement de la forme de l'élément structurant.

1.6 – On veut montrer que : « dilater la forme c'est éroder le fond puis inverser », soit que : **$Ibi \oplus SE = (Ibi^c \ominus SE)^c$** :

- on créé Idi = $Ibi \oplus SE$:
  Idi = imdilate(Ibi,SE) ;

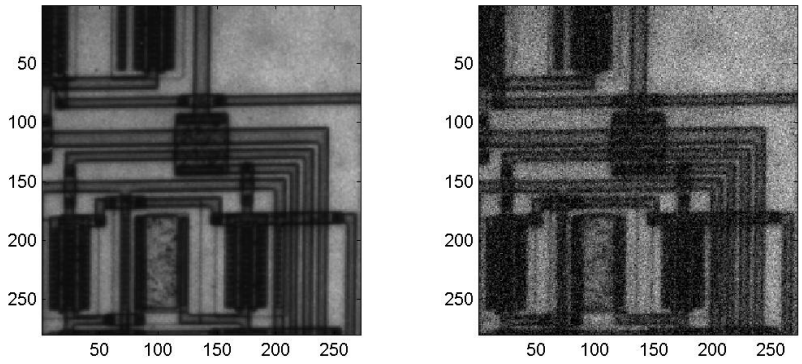
- on créé Idi2= $(Ibi^c \ominus SE)^c$:
  Idi2 = ~imerode(~Ibi,SE) ;

Les deux images obtenues sont identiques. On peut vérifier ce résultat en entrant la commande : ***isequal***(Idi,Idi2). La fonction ***isequal*** retourne 1 si les deux matrices sont identiques, 0 sinon.

2.1 – Voici la liste de commandes pour chargez l'image *CIRCUIT.TIF* et lui ajouter un bruit gaussien N(0, 1) :

```
I = imread('circuit.tif');
% Ajout de bruit :
[nb_lig, nb_col] = size( I ); % taille de l'image
Bruit = 25 * randn(nb_lig, nb_col);
IB = double( I ) + Bruit;
% Conversion
IB = uint8( IB );
```

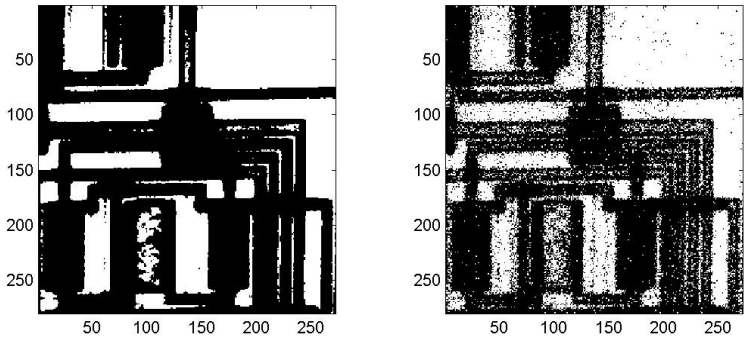
Voici l’image Circuit avant (à gauche) et après (à droite) ajout du bruit :



2.2 – Voici un exemple de script pour construire l’image binaire et l’image binaire bruitée :

```
seuil1 = graythresh(I) ;
Ib = im2bw(I,seuil1);
seuil2=graythresh(IB) ;
IBb = im2bw(IB,seuil2);
subplot(1,2,1)
subimage(Ib)
subplot(1,2,2)
subimage(IBb)
```

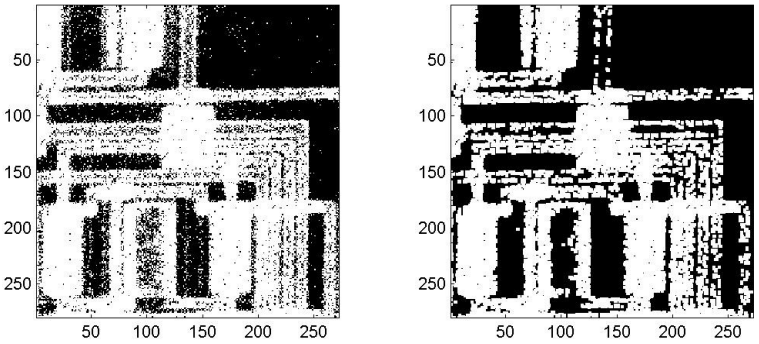
Voici les résultats obtenus:



2.3 – On considère, par exemple, un élément structurant à 8-connexité. Pour réaliser l’ouverture de l’image bruitée, on inverse l’image (afin d’avoir les formes en blanc et le fond en noir), puis on écrit :

```
SE = [1 1 1 ;1 1 1 ;1 1 1] ;
Iouv = imdilate(imerode(~IBb,SE),SE) ;
subplot(1,2,1)
subimage(~IBb)
subplot(1,2,2)
subimage(Iouv)
```

Voici les images obtenues:



Dans le cas présent, l'intérêt d'une telle transformation est d'éroder dans un premier temps les formes de l'image pour supprimer les pixels isolés qui correspondent au bruit, puis de dilater les formes de l'image afin de leur rendre une proportion proche de celle qu'elles avaient avant l'érosion. Le bruit est ainsi atténué.

2.4 – Si l'élément structurant n'est pas symétrique, l'érosion va bien atténuer le bruit de l'image (pixels isolés sur le fond). Pour reconstruire les formes initiales, on applique une dilatation avec le même élément structurant.

- ♦ Mise en garde :

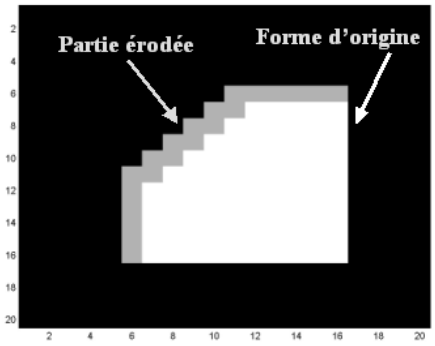
Il faut prendre garde à bien comprendre la définition d'une dilatation :

$$X \oplus B = \{ p \in S, \text{ tel que } \overline{B_p} \cap X \neq \emptyset \}$$

Le résultat de la dilatation est l'ensemble des points, tels que si on leur applique l'élément structurant **symétrique**, l'intersection avec la forme X n'est pas l'ensemble vide.

On considère, par exemple, un élément structurant non symétrique : $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$.

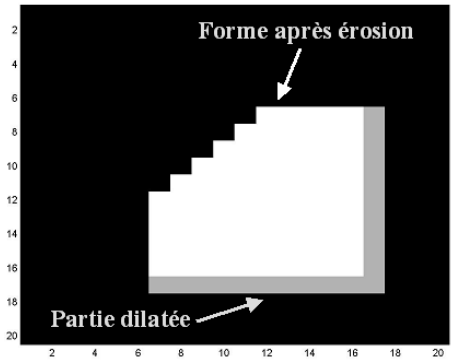
Le coin supérieur gauche est donc tronqué par dilatation:



Pour reconstruire la forme, on veut utiliser une dilatation.
Considérons que la définition de la dilation **ne tienne pas** compte de la symétrie : $X \oplus B = \{ p \in S, \text{ tel que } B_p \cap X \neq \emptyset \}$.

Attention cette définition est erronée et a pour unique but de vous présenter une erreur classique à ne pas commettre avec un élément structurant non symétrique.

Avec une mauvaise définition de la dilatation, on s'aperçoit que la reconstruction avec le même élément structurant n'aboutit pas :

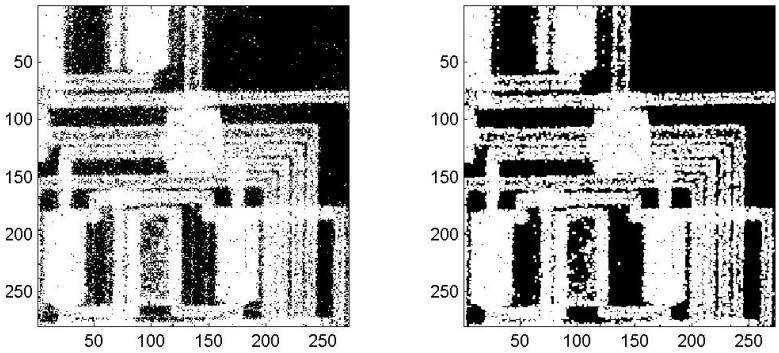


Pour reconstruire la partie grisée, il faut utiliser l'élément structurant suivant : $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ qui est le symétrique de l'élément structurant de départ.
Afin de créer des ouvertures et des fermetures qui garantissent cette relation de symétrie, et donc une reconstruction cohérente des formes, **la définition de la dilatation est caractérisée par l'élément structurant symétrique de celui utilisé lors de l'érosion**.
Dans le cas d'une ouverture avec un élément structurant symétrique, le problème ne se pose évidemment pas.

Voici un exemple de script pour réaliser l'ouverture de l'image bruitée binarisée *Circuit* avec un élément structurant non symétrique :

```
SE1 = [1 1 0 ;1 1 0 ;0 0 0]
SE2 = [0 0 0 ;0 1 1 ;0 1 1]
Iouv = imdilate(imerode(~IBb,SE1),SE2) ;
```

Voici les résultats obtenus :



Le bruit est atténué et les formes sur l'image de sortie gardent les mêmes proportions que sur l'image d'entrée.

3.1 – Voici la commande pour charger l'image *Pearlite.tif*, la binariser et l'inverser :

```
I = imread('pearlite.tif');
L = graythresh(I)
I = ~im2bw(I,L);
```

3.2 – Pour réaliser la fermeture de cette image on créé un élément structurant avec la fonction ***strel*** de Matlab :

```
SE = strel('disk', 6)
```

Dans la console on a alors :

```
SE =

Flat STREL object containing 109 neighbors.
Decomposition: 6 STREL objects containing a total of 22 neighbors

Neighborhood:
     0     0     1     1     1     1     1     1     1     0     0
     0     1     1     1     1     1     1     1     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     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     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
     0     1     1     1     1     1     1     1     1     1     0
     0     0     1     1     1     1     1     1     1     0     0
```

l'image de la fermeture est alors obtenue par la commande :

```
Ifer = imclose(I,SE);
```

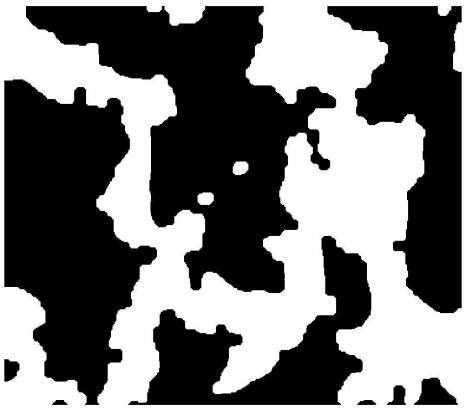
Voici l'image obtenue:

La fermeture adoucit les bords des formes, bouche les canaux étroits, fusionne les objets proches les uns des autres et bouche les trous de petite taille.

3.3 – Pour réaliser l’ouverture de l’image précédente, on utilise la fonction ***imopen*** de Matlab :

```
Im = imopen(Ifer,SE);
```

On obtient alors l’image suivante:



L’ouverture d’une image binaire adoucit les bords des formes en supprimant les détails.

3.5 – La succession de la fermeture puis de l’ouverture nous a permis d’extraire les objets larges de l’image binaire, et de réaliser une segmentation. Ces deux traitements sont typiquement utilisés pour adoucir les bords, compléter, ou pour enlever des objets dans une image binaire. Le choix de l’élément structurant dépend de la taille et de la forme des objets à modifier.