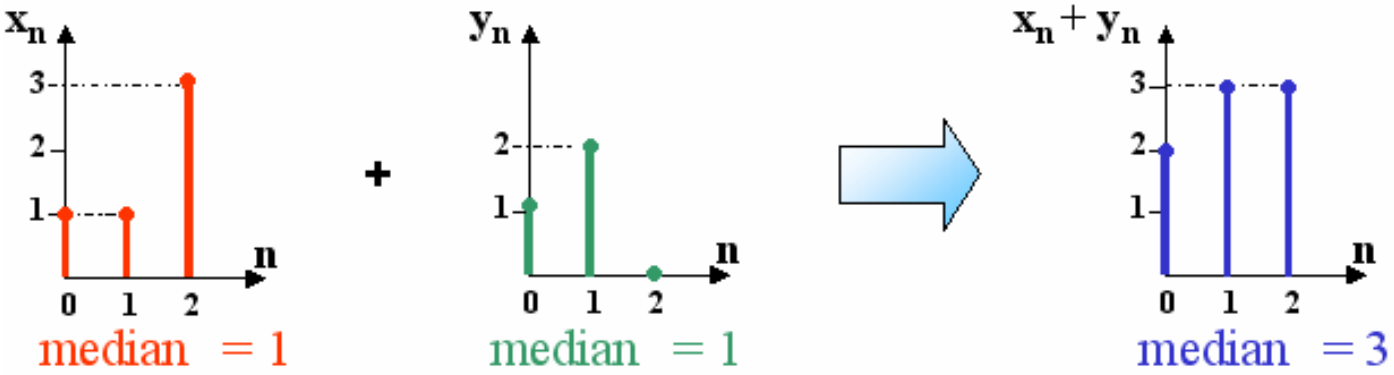
# Example of non-linear filtering: Median filtering

- Median filtering is a non-linear operation:

  Generally: median { $x_m + y_m$ } ≠ median { $x_m$ } + median { $y_m$ }

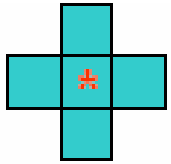
  - example with signal sequences (length = 3):




- Odd window sizes are commonly used in median filtering:

  $3 \times 3\ ;\ 5 \times 5\ ;\ 7 \times 7$

  - example: 5-pixel cross-shaped window



- For even-sized windows ( 2 K values ): the filter sorts the values then takes the average of the two middle values :

$$\text{Output value} = \frac{(\ K^{th} \text{ classified value} + (K+1)^{th} \text{ classified value}\ )}{2}$$

# Example of median filtering application

- 3×3 original image:

| | | |
|---|---|---|
| **91** | **55** | **90** |
| **77** | **68** | **95** |
| **115** | **151** | **210** |

➢ Median filtering using the full 3×3 neighborhood:

↳ we sort the original image values on the full 3×3 window:

55 , 68 , 77 , 90 , **91** , 95 , 115 , 151 , 210

median value = 91

➢ Median filtering using 5-pixel cross-shaped size:

| | | |
|---|---|---|
| | **55** | |
| **77** | **68** | **95** |
| | **151** | |

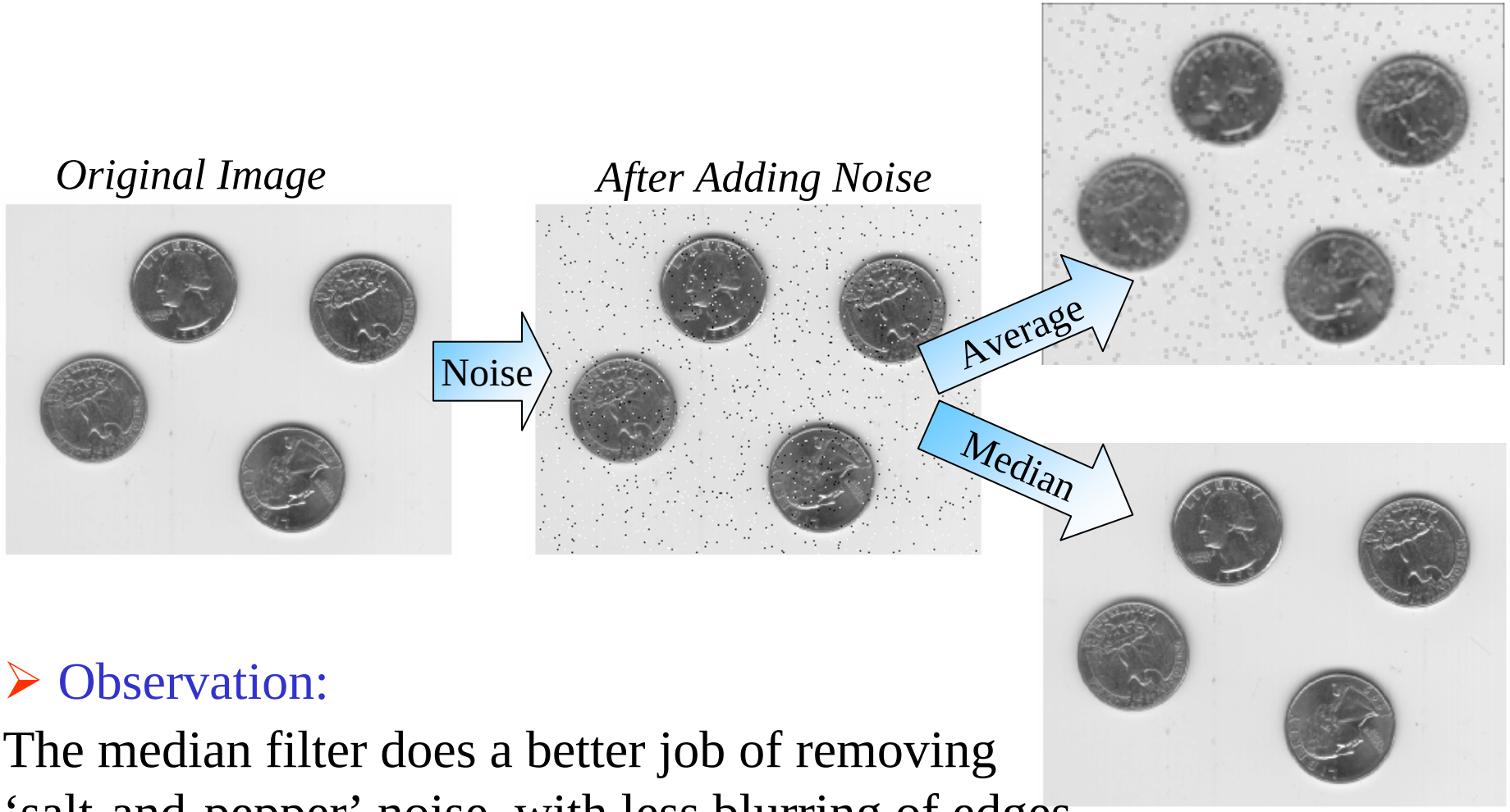
↳ We sort the original image values on the 3×3 cross window:

55 , 68 , **77** , 95 , 151

median value = 77

# Comparison: median filter and averaging filter

- Original image "*Eight.tif*" with added 'salt-and-pepper' noise then filtered with a (3-by-3) averaging filter and a (3-by-3) median filter.




➢ Observation:

The median filter does a better job of removing 'salt-and-pepper' noise, with less blurring of edges.

Exercise Chapter 4 – Comparison: non-linear filtering *vs.* linear filtering

Unlike filtering by convolution (linear filtering), non-linear filtering uses neighboring pixels according to a non-linear law. The median filter (specific case of rank filtering), which is used in this exercise, is a classical example of these filters. Just like the linear filters, a non-linear filter is performed by using a neighborhood.

## 1. To create a noisy image

Load the image *BOATS_LUMI.BMP*. Update the path browser.
The aim is to compare the effects of a linear and a non-linear filtering used to reduce the noise in an original image.
The Matlab function ***imnoise*** allows you to add different classical noises to an image. Use this function for computing the noisy image of *BOATS_LUMI* (use a "salt-and-pepper" noise).
Display the noisy image and explain how this kind of noise is added?

## 2. Application of a linear filtering

We want to reduce the noise in the image. Let us consider a $(3 \times 3)$ averaging filter for reducing the noise. Its convolution kernel is:

$$\frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Perform this filter (by using the Matlab function ***imfilter***), and visualize the noisy image. Interpret the result.

## 3. Application of a non-linear filtering

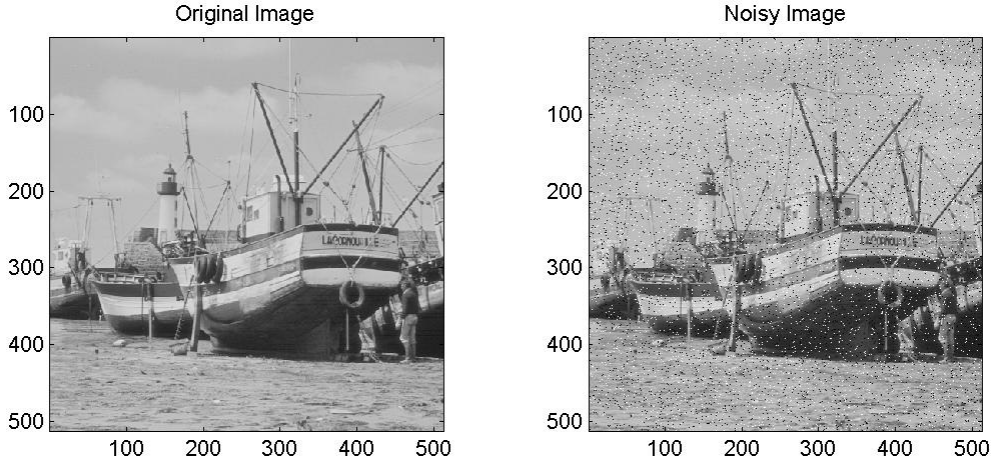
We want now to reduce the noise by using a $(3 \times 3)$ median filtering. You can implement this filtering with the Matlab function ***medfilt2***. Explain how this function works. Process the noisy image by performing this median filtering and visualize the results. Explain the differences between these results and the results of the linear filtering.

**Exercise solution: Linear filtering vs. non-linear filtering**

1 – Here are the commands to visualize the noisy image *BOATS_LUMI*:

```
I=imread('BOATS_LUMI.BMP') ; % to read the grayscale image
IB = imnoise(I,'salt & pepper'); % to create the noisy image
figure(1)
subplot(1,2,1)
subimage(I)
title('Original Image')
subplot(1,2,2)
subimage(IB)
title('Noisy Image')
```

Here are the images displayed:

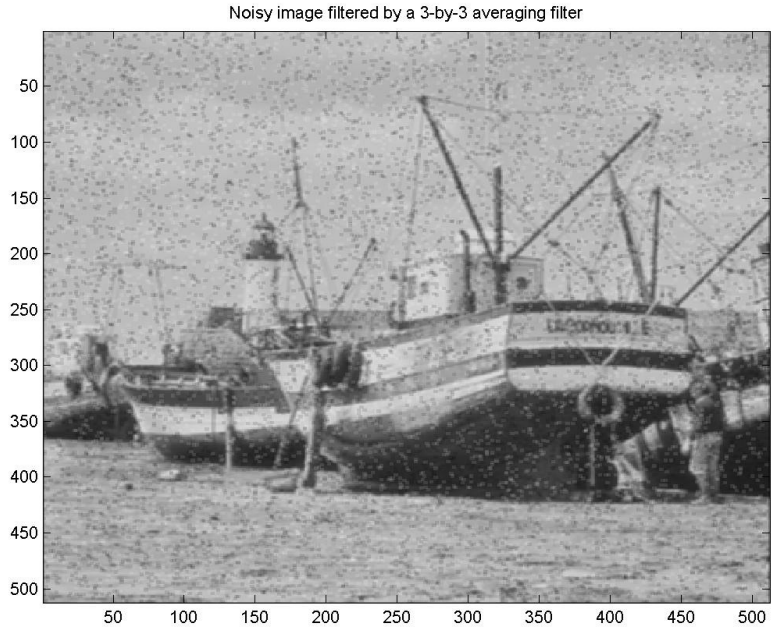



The 'salt-and-pepper' noise consists of random pixels being set to black or white (the extremes of the gray level range).
This kind of impulse noise can be generated by image digitalization or during image transmission.

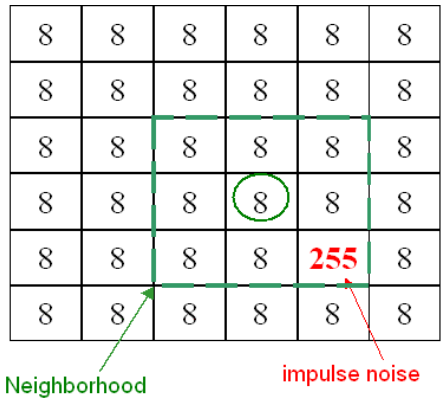
2 – Here are the commands to perform the 3-by-3 averaging filter:

```
% Averaging filter
N = ones(3)/9 ; % convolution kernel
If1 = imfilter(IB,N) ;
figure(2)
image(If1)
title('Noisy image filtered by a 3-by-3 averaging filter')
v=0:1/255:1; colormap([v' v' v']); % LUT for displaying in gray levels
```

Here is the image displayed:




The "salt-and-pepper" noise is not significantly reduced. We can still easily distinguish the noisy pixels. Each output pixel value is the mean value of all the values of its neighboring pixels, therefore when a noisy pixel is included in the neighborhood, its extreme value (0 and 255) is used to compute the mean value:



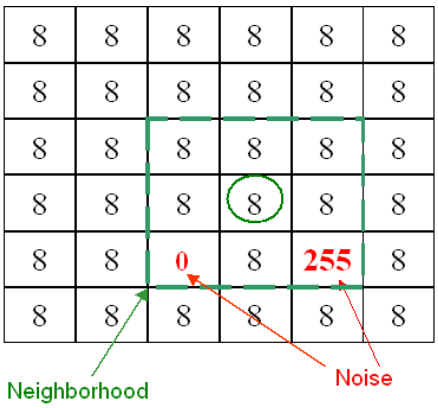

All the pixels of this image are set to the luminance value 8 except one noisy pixel which has the luminance value 255. The output value of the surrounded pixel (and of all pixels whose neighborhood contains the value 255) is equal to: $(8 \times 8 + 255)/9 \approx 35$. The output value of this pixel is thus not representative of its neighborhood, the noise is not enough reduced. This linear filtering is not appropriate for reducing the impulse noise.

3 – Here are the commands to perform the median filtering:

```
% Median filtering
If2 = medfilt2(IB,[3 3]) ; % 3-by-3 median filtering
figure(3)
image(If2)
title('Noisy Image filtered by a 3-by-3 median filter')
v=0:1/255:1; colormap([v' v' v']); % LUT for displaying in gray levels
```

Here is the image visualized:

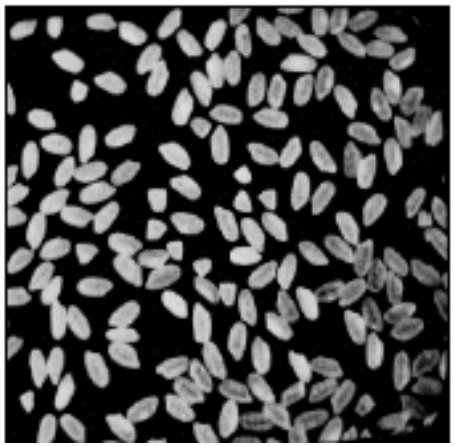

The “salt-and-pepper” noise is significantly reduced. This median filtering does a better job of removing noise, with less blurring of edges. The filter sorts the neighboring values of a pixel, the output value is then the median value of all these sorted values (non-linear operator):




Let us consider the previous example: the pixel values are sorted by increasing order: 0, 8, 8, 8, **8**, 8, 8, 8, 255. The median value is 8. **The extreme luminance values 0 and 255 have no effect on the output value by using this non-linear filtering**. The median filtering is efficient for reducing the impulse noise.

# Example of a morphological filtering application, Rice Grains (1/2)

- a : Original Image (rice grains)
  - **Preprocessing:**
- b : Binarization and regularization of the shape
- c : Deletion of the boundary objects (grains on the image boundaries)



*a : original image*

*b : segmentation of a*

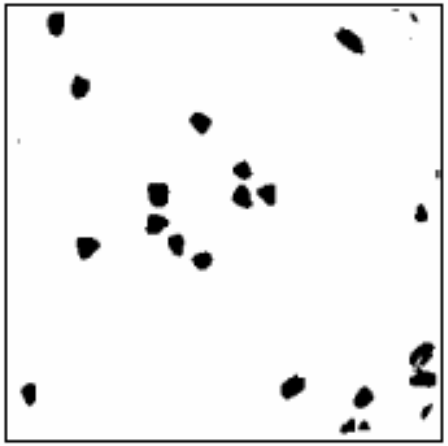
*c : internal grains*

# Example of a morphological filtering application, Rice Grains (2/2)

- **Separation and classification of the grains by morphological operations:**

- d : Intact grains
- e : Broken grains
- f : Grains with connected components

*d : intact grains*

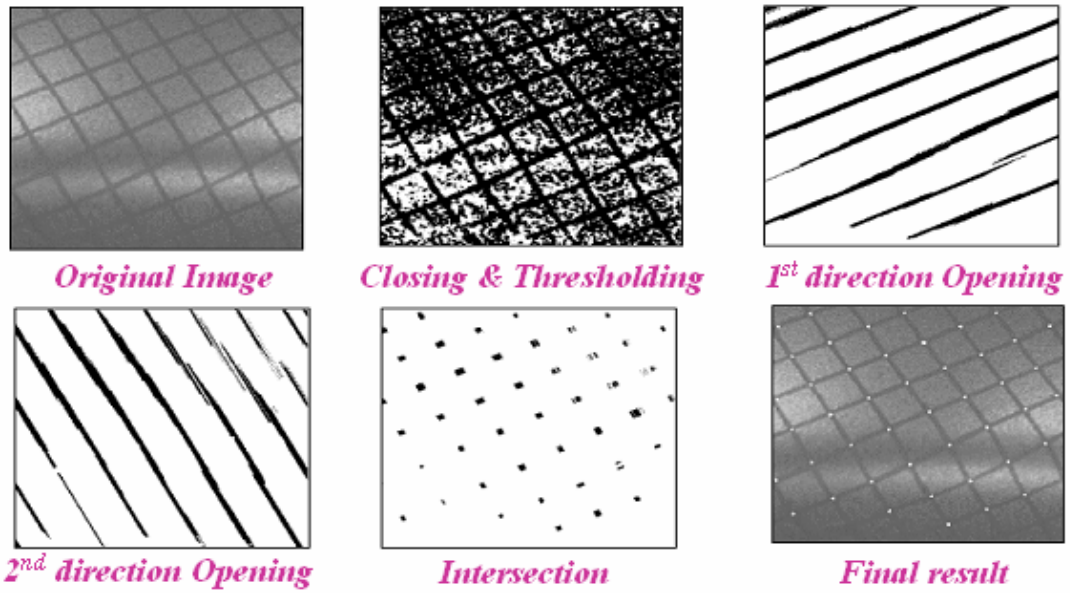


*e : broken grains*

*f : connected components*

**Possibility of measuring and counting the rice grains, etc.**

# Example of morphological filtering application, metal sheets




- Morphological closing to smooth and to regularize the shapes, then binarization of the grayscale image by thresholding
- Morphological opening along 2 known diagonal directions for detecting the lines
- Detection and localization of intersection points

**Motion detection, measurement of sheet deformations, etc.**

# Morphological filtering

## ♦ Objectives:

- Obtaining sets of connected points having simple shape (binary image)
- Obtaining principal connected components ( in lower numbers )
- Regularization of the shape of the image signal

## ♦ Applications:

- Size filtering
- Object counting
- Object measurement

# Chapter 4

## NON-LINEAR FILTERING

## Morphological operations

*Case of binary images*

# *Recall : classical binary operators*

*X , Y , Z : Boolean variables ⇒ possible states : '0' or '1'*

• operator **AND** : Z = X AND Y ( notation Z = X . Y )

| X | Y | Z |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

• operator **OU** : Z = X OU Y ( notation Z = X + Y )

| X | Y | Z |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

• operator **NON** : $Z = \overline{X}$

| X | Z |
|---|---|
| 0 | 1 |
| 1 | 0 |

A variable which can take only two states (true or false, turned on or turned off, top or bottom, positive or negative, black or white, etc.) is called a ***Boolean variable***. Typically we allow the value '1' to one of the two states, and the value '0' to the other state.
Three classical binary operators are defined:

- the operator 'AND', the output value is 'true' only if both input values are 'true';
- the operator 'OR', the output value is 'true' if at least one of the two input values is 'true';
- the operator 'NOT, the output value is the opposite value of the input value.

***"Truth tables"*** (which define the operators) are displayed on the bottom figure. Commonly we write "A . B" instead of "A and B", and "A + B" instead of "A or B".
Note that several other operators are defined from these classical operators: the operator 'NOR' (i.e. 'not or'), the operator 'NAND' (i.e. 'not and'), the operator 'XOR' (i.e. 'eXclusive or'), …

# *Morphological filtering*

## ➢ Concept :

- set theory description of images.
- take into account the shape of the image structured components.
- basic application: two-level image processing, extension to multilevel image processing

## ➢ Operators :

- 2 basic operators ⇒ "EROSION" and "DILATION"
- Combination of these 2 opertors

  ⇒ 2 complementary operators:

  "OPENING" and "CLOSING"

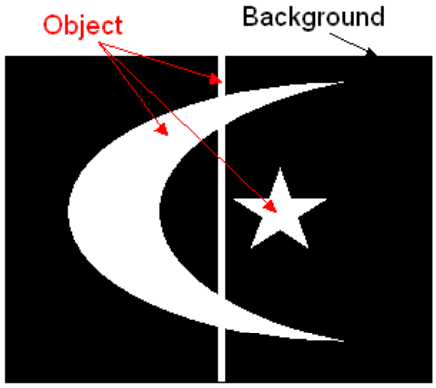
- Operators depending on a structuring element.

Morphological filtering is based on mathematical morphology (set theory description of the images). The morphological operators privilege the shape rather than the signal amplitude. They can be used to process both binary images (two levels: white or black) and grayscale images.
We will just present the binary morphological filtering here. This non-linear filtering is based on two basic operators (***erosion*** and ***dilation***) and two complementary operators which result from the combination of the two first basic operators (***opening*** and ***closing***).
A ***structuring element*** is used to define a neighborhood shape and size for morphological operations, including dilation and erosion.
We will now define all these concepts.

# *Binary Image*

- L(m,n) ∈ {0, 1} ∀(m,n) ∈ S image support
  L = 0 "background" pixel
  L = 1 "Object" pixel
  ↳ 2 categories : « Background », « Object »

- Object(s) : X
  X = { p ∈ S / L(p) = 1 }




A binary image is an image whose pixels (m, n) take their value from among two possible luminance values. These two possible values are conventionally 0 (background) and 1 (objects).
The objects 'X' are defined as the set of pixels 'P' with affix 'p' that belong to the image support 'S' and such as their luminance value L(p) is equal to 1:

$$\mathbf{X = \{ p \in S / L(p) = 1 \}}$$

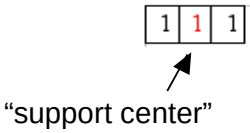
A binary image can be created by different methods: a digitalization with only two reconstruction levels, or a binarization of a grayscale image by using its histogram to choose an appropriate threshold (cf. exercise « Binarization » of chapter 2).

# *Structuring Elements (binary images)*

➢ ***Structuring element :***

set of pixels at '1' on a support with the origin located at (0, 0) coordinates (red pixel)

❖ 3 typical examples:




- a -

1-D structuring element


- b -


- c –

2-D structuring elements

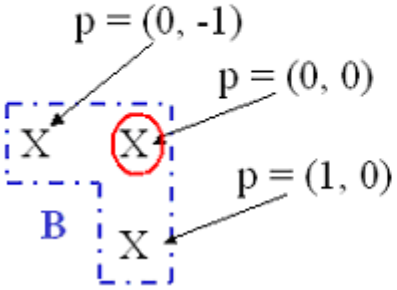
$$B = \{p \text{ so that } L(p) = 1\}$$

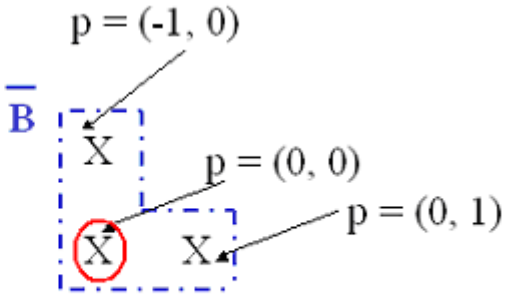
↳ *structuring element*

symmetry: $\overline{B} = \{p \in B \Rightarrow -p \in \overline{B}\}$

The **structuring element** B is a set of pixels at luminance value '1'. The structural element includes generally the origin O(0, 0), but not necessarily.
Let B be a structuring element composed by a set of pixels P of affix p. We define then the symmetrical structuring element $\overline{B}$ as being the set of pixels of affix '-p'.

Example :



# *Morphological Erosion (by B)*

Notation $X \ominus B$

Definition $X \ominus B = \{ p \in S \text{ so that } B_p \subseteq X\}$

where $B_p$ is the translation of B by p

- $X \ominus B$ : set of pixels p in S such for every p, $B_p$ is fully included in the object X

Example :

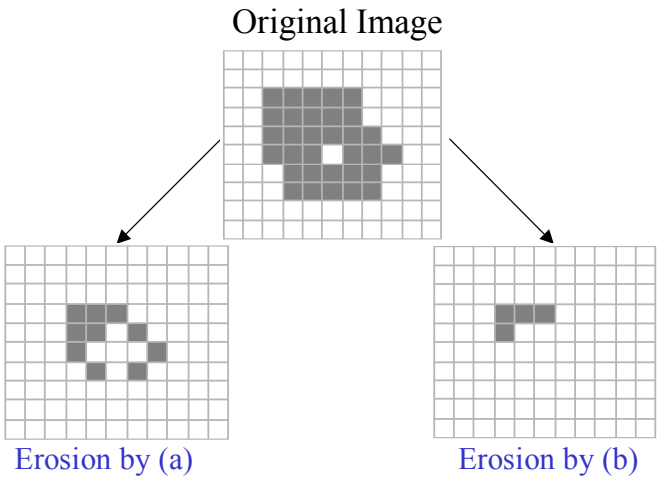
Structuring Element

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

(a) "4-connected" structuring element

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

(b) "8-connected" structuring element




**The erosion of an object X by the structural element B is noted « XΘB »**. Let $B_p$ denote the translation of B so that its origin is at p. The erosion of objects X by the structuring element B is then the set of all pixels P(p) in the image support S such that $B_p$ is a subset of X:

$$\mathbf{X \Theta B = \{ p \in S, \text{ so that } B_p \subseteq X\}}$$

The figure above displays two erosions which are performed on the same original image but with two different structuring elements:

- case a: 4-connected structuring element (the origin has 4 neighboring). Each pixel of the image support which has the luminance value 0, or one of whose 4 neighboring pixels has the value 0, is set to 0 after filtering.

- case b: 8-connected structuring element (the origin has 8 neighboring). Each pixel of the image support which has the luminance value 0, or one of whose 8 neighboring pixels has the value 0, is set to 0 after filtering.

We can observe that the two erosions remove the white pixels which are isolated in the background and erode the edges of the objects.

# *Morphological Dilation (by B)*

Notation : $X \oplus B$

Definition: $X \oplus B = \{ p \in S \text{ so that } \overline{B}_p \cap X \neq \varnothing \}$

where $\overline{B}_p$ is the symmetric of B translated by p

• $X \oplus B$ : set of pixels p in S such for every p, $\overline{B}_p$ has a non-null intersection with X

Example :

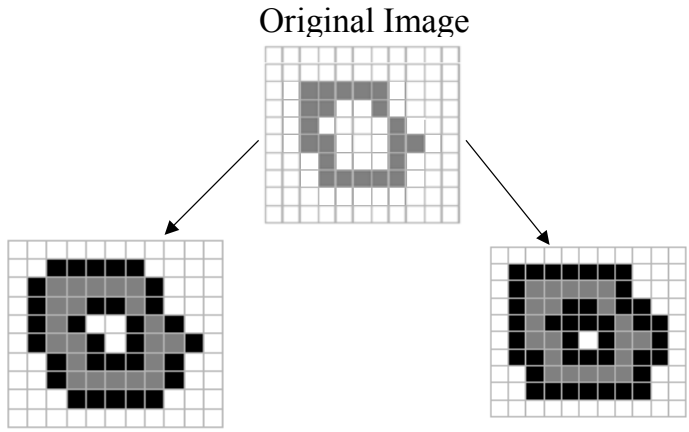
Structuring Element

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

(a) 4-connected structuring element

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

(b) 8-connected structuring element




**The dilation of objects X by a structuring element B is noted « $X \oplus B$ »**. Let $\overline{Bp}$ denote the symmetric of B translated so that its origin is at p. The dilation of objects X by the structuring element B is then defined as being the set of pixels P(p) so that $\overline{Bp}$ has a non-null intersection with X:

$$X \oplus B = \{ p \in S, \text{ so that } \overline{B_p} \cap X \neq \varnothing \}$$

The figure above displays two examples of dilations:

- case a: 4-connected structuring element. Each pixel of the image support which has the luminance value 1, or one of whose 4 neighboring pixels has the value 1, is set to 1 after filtering.

- case b: 8-connected structuring element. Each pixel of the image support which has the luminance value 1, or one of whose 8 neighboring pixels has the value 1, is set to 1 after filtering.

Note that the dilation removes the holes which are isolated in the objects and dilates the object edges by taking account of the structuring element shape.

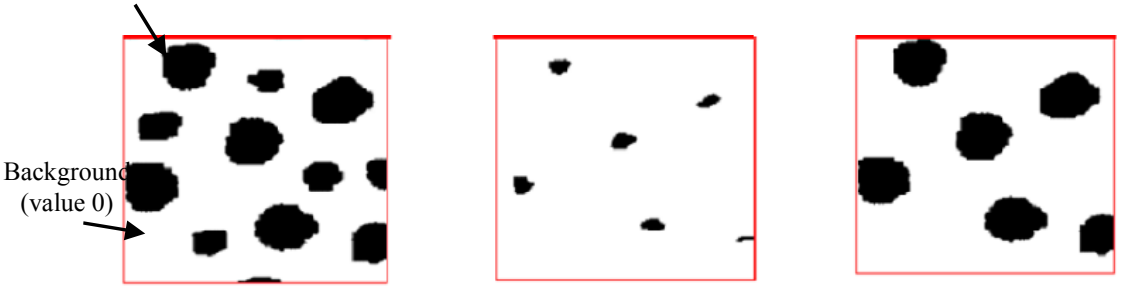
Property: To erode $X^C$ by B ($X^C$ is the complementary object of X with respect to S) is equivalent to dilating X by B. The erosion and the dilation are dual: $X \oplus B = ( X^C \ominus B )^C$

# *Filtering by Erosion and Reconstruction*

(by constrained dilations)

Objects (value 1)

Structuring element B




**Initial Image $I_a$**

**Image $I_b$ : Eroded of $I_a$ by the structuring element B**

**$I_c$: Reconstruction from $I_b$ and $I_a$**

$$I_c = (\ldots(((I_b \oplus B) \textbf{ AND } I_a) \oplus B) \textbf{ AND } I_a)\ldots \ldots\ldots)$$

*Up to idempotence of $I_c$*

*The image $I_b$ is the start point for the first dilation in the series of dilations*

The figure above shows an example of image reconstruction by successive morphological transformations T. These transformations are based on a first erosion by the structuring element B then a series of dilations by the same structuring element. The transformation T is defined by: "***T(X) = (X⊕B) AND $I_a$***". The rebuilt image $I_c$ is computed by a series of transformations T up to idempotence of $I_c$: $I_c$=ToTo…oToT($I_b$). This image reconstruction allows you to keep only the large dark patterns of the original image $I_a$. The isolated small patterns are removed.

Note: an idempotent transformation is a transformation that when composed with itself, gives itself as result i.e. ToT(X) = T(X). That is the case here insofar as we compare the dilation result with the initial image $I_a$ (via the classical binary operator AND). This property ensures the stability of the morphological filter.

## *Morphological Opening (by B)*

Notation : X o B

Definition $X \circ B = (X \ominus B) \oplus B$

Erosion then Dilation

Effects: Shape smoothing

- Deletion of small details at the object border
- Cutting of narrow isthms

## *Morphological Closing (by B)*

Notation : $X \bullet B$

Definition $X \bullet B = (X \oplus B) \ominus B$

Dilation then Erosion

Effects: Shape smoothing

- Filling up of narrow channels and small holes
- Connected components

Property: Idempotence

$(X \circ B) \circ B = X \circ B$ $\quad$ $(X \bullet B) \bullet B = X \bullet B$

By combining the two basic morphological operators which are the erosion and the dilation, we define two new morphological transformations: the "morphological opening" and the "morphological closing". These two transformations are indempotent. We define:

- The **morphological opening** of X by B, noted: X o B.

This transformation corresponds to an erosion followed by a dilation, using the same structuring element B for both operations:

$$\mathbf{X \circ B = (X \ominus B) \oplus B}$$

The morphological opening smoothes the object borders and removes small objects while preserving the shape and size of larger objects in the original binary image.

- The **morphological closing** of X by B, noted: X • B.

The related operation, morphological closing is the reverse: it consists of dilation followed by an erosion with the same structuring element B:

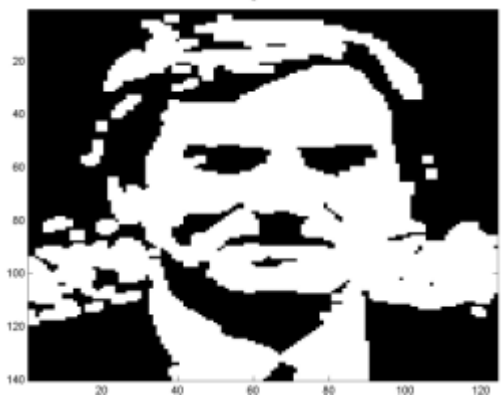
$$\mathbf{X \bullet B = (X \oplus B) \ominus B}$$

The morphological closing smoothes the object borders too, it merges together the small features in the image that are close together and the narrow channels and small holes are thus filled up.

Here are the results obtained from an original binary image which is filtered by the four morphological operators by using each time the same 8-connected structuring element:

*Original binary image*

*Eroded Image*



*Dilated Image*

*Morphological Opening*

*Morphological Closing*

The original binary image ***objects*** correspond to the white pixels: the face, hair, the shirt collar, and the vegetation behind the character. ***Erosion*** erodes the object edges and removes the isolated white pixels. On the other hand, dilation fills the holes up and dilates the edges. These results are definitely visible on the vegetation and the white zones of character hair.
The ***morphological closing*** and ***opening*** smooth the object edges (face, vegetation, etc.). The morphological opening performs initially an erosion, therefore some parts of the objects are removed, and they could not then be dilated. In the image computed by the morphological opening, several objects are smaller than the objects of the image computed by the morphological closing (hair, etc.).

Exercise Chapter 4 – digital binary image processing by morphological filtering

The morphological analysis aims at modifying the shape of the objects in the image, for example, for discriminating these objects according to their size, for filling the holes, etc.

- Recall:

The morphological processing uses the neighborhood of a pixel to compute its output value. This neighborhood is defined by a ***structuring element*** which is a binary mask.

Example of a 4-connected structuring element (the center is surrounded):

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

The morphological processing are based on two basic operators: the ***erosion*** and the ***dilation***.

*Definition:*

Let "S" be the image support. The coordinates of a pixel P are given by its affix p:

- The erosion of the object X by the structuring element B is noted "$\mathbf{X \ominus B}$". The erosion is define by: $\mathbf{X \ominus B = \{ p \in S,}$ **so that** $\mathbf{B_p \subseteq X\}}$ ;
- The dilation of the object X by the structuring element B is noted "$\mathbf{X \oplus B}$". The dilation is defined by: $\mathbf{X \oplus B = \{ p \in S,}$ **so that** $\mathbf{\overline{B_p} \cap X \neq \varnothing\}}$.
- The erosion is the complementary operator of the dilation:

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

Where:
- $X^C$ is the complementary object of X,
- $\overline{B}$ is symmetrical to the structuring element B (i.e. $\overline{B}(i, j) = B(-i, -j)$ ),
- $B_p$ denotes the translation of B so that its origin is at p.

*You are going to use Matlab to perform morphological filtering. Open the Matlab help window (command* ***helpwin****) and display the* ***Image Processing Toolbox****. The description of each function used in this exercise is given here.*

## 1. Binary image Erosion and Dilation

Update the path list in the ***path browser***. Open a new file "M-file" in which you will type your commands and of which each line (finishing with ";") will be then interpreted. Load the image *CIRCUIT.TIF* with ***imread***.

1.1 – Binarize this image by performing your choice of method (cf. exercise *Binarization*, chapter 2).

1.2 – Conventionally the binary image background is black (set to 0) and the objects are white (set to 1), therefore you can create the binary inverse video image by using the operator "~" ( **I = ~ I** ).

1.3 – Create a structuring element. You can write it directly as a binary array (ex. SE = ***ones***(3), SE = [0 1 0;1 1 1;0 1 0], … ). Erode the objects with the Matlab function ***imerode***. By looking at the Matlab help, define this morphological operator.

1.4 – In the same way, dilate the objects of the binary image with the Matlab function ***imdilate*** and define this morphological operator.

1.5 – Erode or dilate the binary image with several various structuring elements. Observe the results when the structuring element is not symmetrical with respect to its origin.

1.6 – We want to verify the duality relation between the erosion and the dilation. Create a first binary image by eroding the background, then create a second one by dilating the objects. Compare these two binary images and conclude.

## 2. Morphological Opening (erosion then dilation) of a noisy binary image

2.1 – Load the image *CIRCUIT.TIF* and add noise to it with the function ***randn***. This function allows you to generate a Gaussian N(0, 1) array that you can increase and add to the original image.

Notes:
- The noise array dimensions must be equal to the original image dimensions;
- Convert your data (functions ***double*** and ***uint8***).

2.2 – Binarize the original image and the noisy image, then compare them.

2.3 – Use a symmetrical structuring element. Erode then dilate the noisy image. The combination of these two operators is the morphological ***opening***. Why do this?

2.4 – What do you note if the structuring element is not symmetrical? Propose and test a solution for rebuilding identically the objects. Idea: you must use two different structuring element.

## 3. Morphological Closing then opening of a binary image.

3.1 – Load and binarize the grayscale image *PEARLITE.TIF*. Compute the inverse video image.

3.2 – Perform the morphological closing of this binary image (***imclose***). Define this operator (cf. Matlab help). You can use the Matlab function ***strel*** to create the structuring element (this function generates structuring elements).

3.3 – Perform the morphological opening of the previous binary image (***imopen***). Define the operator.

3.4 – Analyze the complete chain.

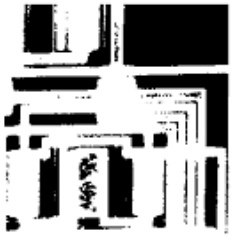
**Exercise solution: Image Processing by Morphological Filtering**

1.1 – Here is an example for binarizing the image *CIRCUIT.TIF* :

```
I = imread('CIRCUIT.TIF');
thres = graythresh(I) % computes a global threshold by using Otsu's method
Ib = im2bw(I,tres); % binarization
```

1.2 – In Matlab, the operator « ~ » corresponds to the binary logical operator 'NOT'. You can thus create the binary inverse video image by typing the command: Ibi=~Ib. Here are the commands for displaying the inverse video (on the right):

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

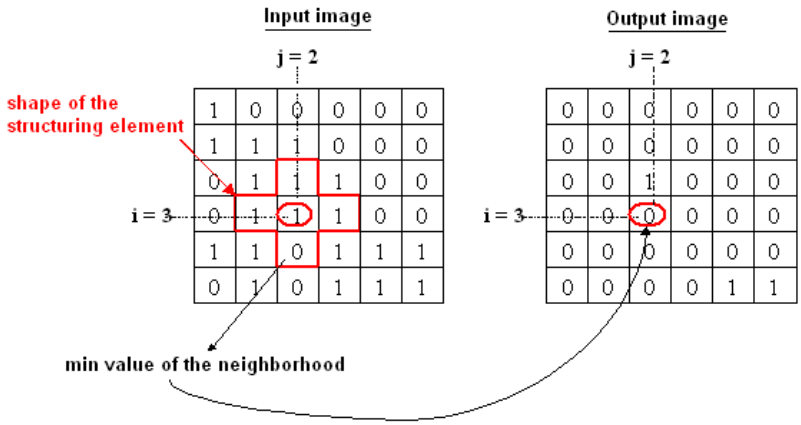
Ib  Ibi 

1.3 – Let us consider, for example, the structuring element: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

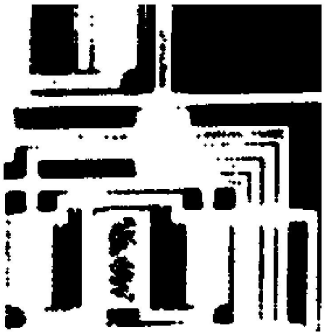
Here are the commands to erode the binary image Ibi and to display the result:

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

The structuring element defines the neighborhood of each pixel: the neighboring pixels are those equal to 1 within the structuring element. We scan the original image and we apply the structuring element to each pixel. The output value of each pixel is the minimum value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0.
The following figure illustrates the erosion. The morphological operation is performed on the pixel (3, 2) with a 4-connected structuring element.



Here is the eroded image:

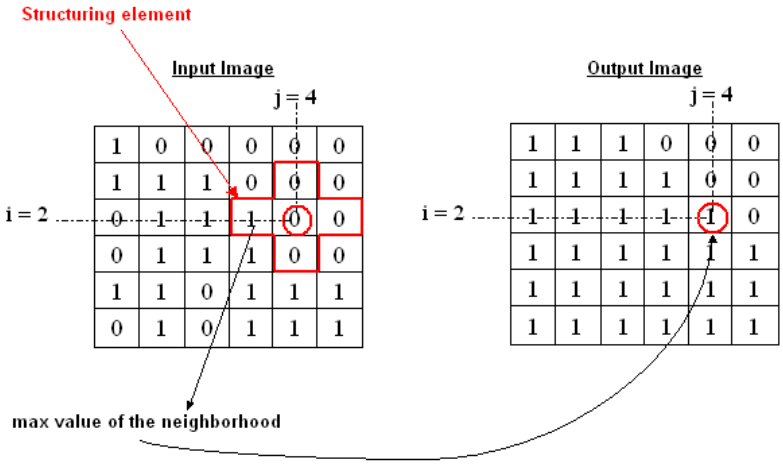


The erosion removes the white pixels which are isolated in the background and erodes the edges of the objects within the image *CIRCUIT*.

1.4 – Here are the commands to dilate the binary image Ibi and for displaying the result:

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

The output value of each pixel is the maximum value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to 1, the output pixel is set to 1.
The following figure illustrates the dilation. The morphological operation is performed on the pixel (2, 4) with a 4-connected structuring element.

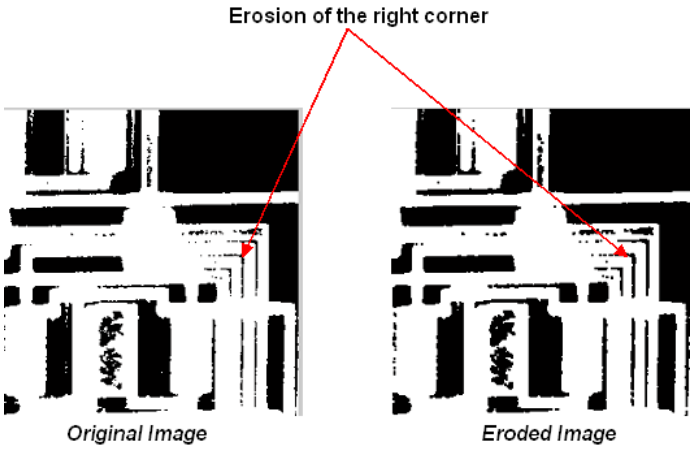



Here is the image obtained:

The dilation removes the holes which are isolated in the objects and dilates the object edges by taking account of the structuring element shape.

1.5 – We erode the image with the following structuring element: $\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

Here the structuring element is not symmetrical. Only the top right corners are eroded.




The output image depends thus strongly on the structuring element's shape.

1.6 – We want to visualize that: "to dilate the objects is equivalent to eroding the background then computing the inverse video", i.e.:

$$\mathbf{Ibi \oplus SE = ( Ibi^c \ominus SE )^c}$$

- we create Idi = Ibi $\oplus$ SE:

```
Idi = imdilate(Ibi,SE) ;
```

- we create Idi2= ( $Ibi^c \ominus SE$ )$^c$:

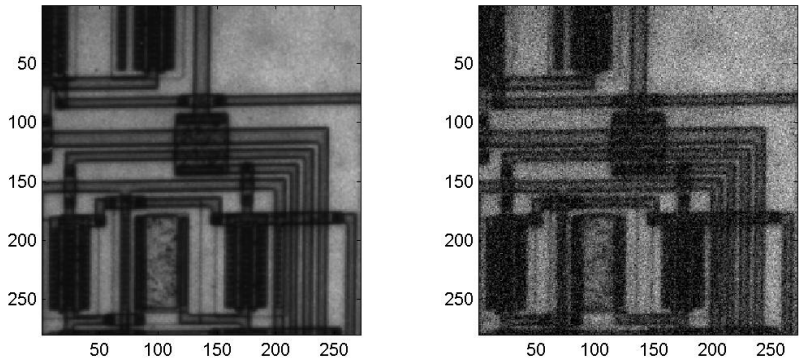
```
Idi2 = ~imerode(~Ibi,SE) ;
```

The two output images are equal. You can verify this property by typing the command: ***isequal***(Idi,Idi2). The function ***isequal*** returns logical true '1' if the input images are the same type and size and hold the same contents, and logical false (0) otherwise.

2.1 – Here are the commands for loading the image *CIRCUIT.TIF* and for adding to it a Gaussian noise N(0, 1):

```
I = imread('circuit.tif');
% Noise :
[nb_row, nb_col] = size( I ); % image size
noise = 25 * randn(nb_row, nb_col);
IB = double( I ) + noise;
% Conversion
IB = uint8( IB );
```

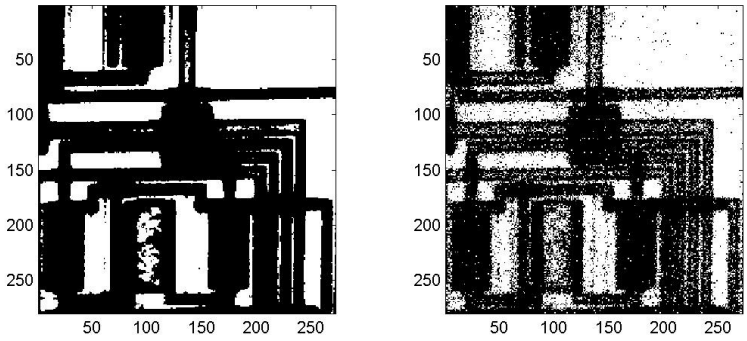
Results before (left) and after (right) having added the noise:




2.2 – Here is an example of a script to compute the binary image and the noisy binary image:

```
thres1 = graythresh(I) ;
Ib = im2bw(I, thres1);
thres2=graythresh(IB) ;
IBb = im2bw(IB, thres2);
subplot(1,2,1)
subimage(Ib)
subplot(1,2,2)
subimage(IBb)
```

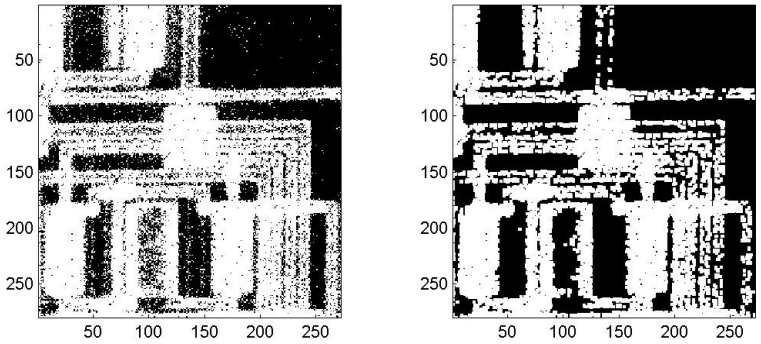
Here are the results:




2.3 – Let us consider an 8-connected structuring element. For performing the morphological opening of the noisy image, we compute the inverse video image (for displaying the objects in white and the background in black), then we enter the commands:

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

Here are the images displayed:




In this case, the interest of such a operation is to initially erode the objects of the image for removing the isolated pixels which correspond to the noise. The result is then dilated to resize the objects. The noise is thus reduced.

2.4 – If the structuring element is not symmetrical, the erosion reduces the noise again (isolated pixels in the background). For resizing the original objects, we perform dilation by the same structuring element.

- ♦ Warning:

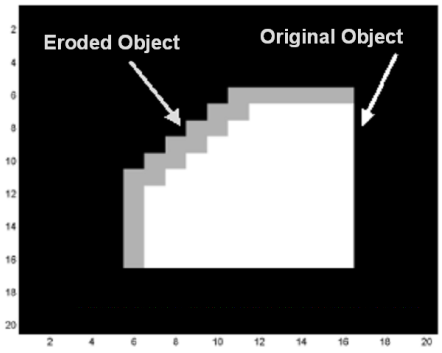
You must be really careful when you perform the dilation:

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

The dilation of an object X by a structuring element B is defined by the **symmetrical** structuring element. The result is the set of points P(p) so that the translated **symmetrical** structuring element has a non-null intersection with the object.
Let us consider, for example, a non-symmetrical structuring element:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

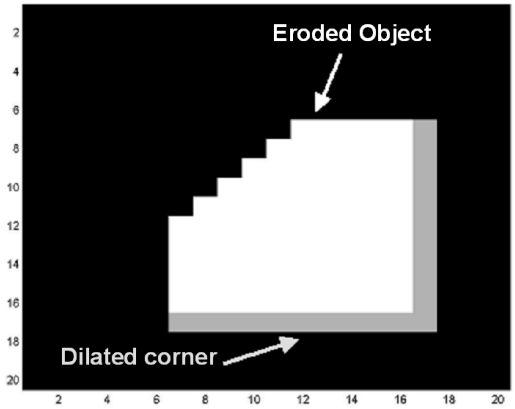
The top left corner is eroded:




For resizing the object, we want to perform dilation.
**Let us consider** that the definition of the dilation **is not based on** the symmetrical structuring element: $X \oplus B = \{ p \in S, \text{ so that } B_p \cap X \neq \emptyset \}$.

**Be careful: this definition is entirely false and aims to show you a classical mistake, made when you don't consider the symmetrical structuring element.**

Here is the result obtained with the bad definition of the dilation. We visualize that the lower right corner is dilated whereas we wanted to dilate the upper left corner:

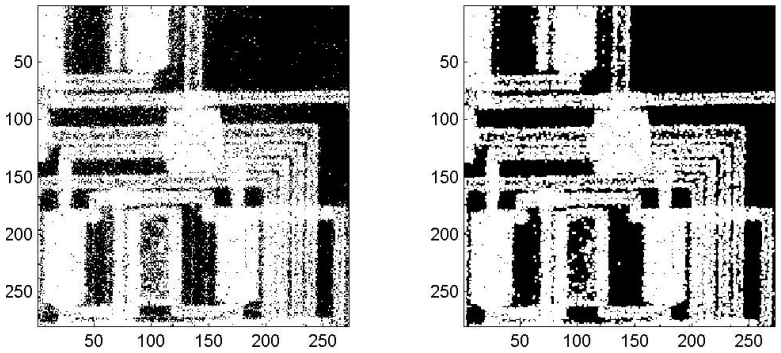



To rebuild the original top left corner, it is necessary to use the structuring element: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ which is symmetrical to the original structuring element.
In order to perform morphological openings and closings which guarantee the reconstruction of the objects, **the definition of the dilation is based on the symmetrical structuring element**. In the special case of a morphological opening by a symmetrical structuring element, the problem does not exist obviously.

Here is an example of a script for performing the morphological opening of the noisy binary image *Circuit* by a non-symmetrical structuring element:

```
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) ;
```

Here are the results:



The noise is reduced and the output image’s objects are the same size as the input image’s objects.

3.1 – Here are the commands for loading, binarizing and computing the inverse video of the grayscale image *Pearlite.tif*:

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

3.2 – For performing the morphological closing of this image, we create a structuring element with the Matlab function ***strel***:

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

Here is the result you can visualize in the command window:

```
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
```

The morphological closing of the image is computed by typing the command:

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

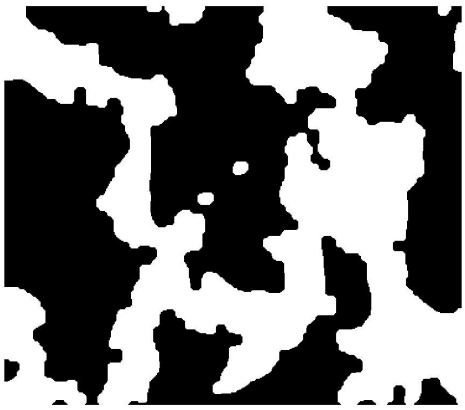
Here is displayed the result:

The morphological closing smoothes the object borders, it merges together the small features in the image that are close together and the narrow channels and small holes are thus filled up.

3.3 – For performing the morphological opening of the previous image, we use the Matlab function ***imopen***:

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

We obtain the following result:



The morphological opening smoothes the object borders and removes small objects while preserving the shape and size of larger objects in the original binary image.

3.5 – The combination of the morphological closing with the morphological opening allows you to extract the large objects within the binary image: the image is segmented. These two operators are typically used for smoothing the object edges, for removing objects or for filling holes. The structuring element shape must be chosen according to the size and the shape of the objects that we want to modify.

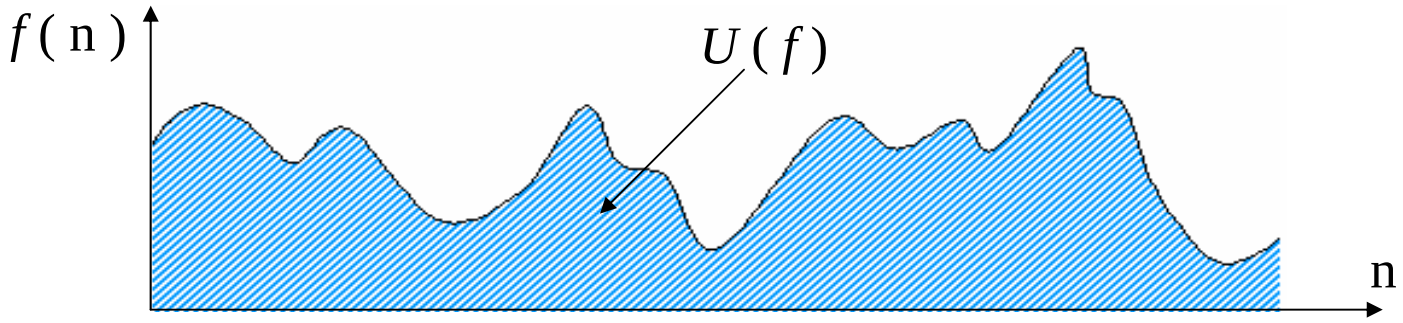
# Morphological Filtering of grayscale images

- Direct extension of binary morphology through the notion of ***shadow*** (or ***sub-graph***) of the image function

- Definitions:
  - gray level ***« f »*** of a pixel P : $f$ (m, n) where (m, n) is the affix of P
  - sub-graph ***« U ( f ) »*** : $U(f) = \{ (m, n, l), l \in R$, so that: $l \leq f(m, n) \}$
  - reconstruction **T** of the image « $f$ » from its sub-graph $U(f)$:

$$f(m, n) = T[U(f)] = \mathrm{Sup}_l \{ l \text{ so that } (m, n, l) \in U(f) \}$$

  - example ( case of 1-D signal ):



# Erosion of a grayscale image

➢ The erosion (noted Θ ) of a grayscale image « *f* » by the structuring element « B » is defined from the sub-graph notions:

- $Y = U(f) \ominus U(B)$
- $f \ominus B = T[Y] = Sup_l \{ l \text{ so that } (m, n, l) \in Y \}$

➢ Commonly we choose a plane structuring element which has zero value

on

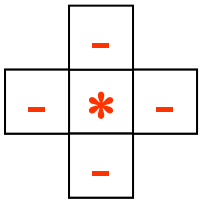
$$(f \ominus B)(P) = \text{Min} \{ \text{values of the P-neighborhood} \}$$

▪ example :

| m \ n | | | |
|---|---|---|---|
| | 115 | 91 | 77 |
| | 95 | 68 | 90 |
| | 55 | 151 | 210 |

*Original Image*

Θ




*Structuring Element (4-connected)*

Classified values of the neighborhood :

**68** , 90 , 91 , 95 , 151

min

$(f \ominus B)(P) = 68$

# Dilation of a grayscale image

- The dilation (noted $\oplus$ ) is also defined from sub-graph notions:
  - $Y = U(f) \oplus U(B)$
  - $f \oplus B = T[Y] = Sup_l \{ l \text{ so that } (m, n, l) \in Y \}$
- If you consider a plane structuring element which has zero value on its support:

$$(f \oplus B)(P) = \text{Max} \{ \text{values of the P-neighborhood} \}$$

- example:

| m \ n | | | |
|---|---|---|---|
| | 115 | 91 | 77 |
| | 95 | 68 | 90 |
| | 55 | 151 | 210 |

*Original Image*

$\oplus$

| | | |
|---|---|---|
| - | - | - |
| - | * | - |
| - | - | - |

*Structuring Element (8-connected)*

Classified values of the neighborhood:

55 , 68 , 77 , 90 , 91 , 95 , 115 , 151 , **210**

max

$(f \oplus B)(P) = 210$

# Examples of morphological filtering
(full 3×3 structuring element)

- Original image

- Dilation

- Erosion

➢ Observation:

The **dilation** of grayscale images increases the luminance value of the pixels which have high-intensity neighborhoods

➢ Note:

The **erosion** of grayscale images decreases the luminance value of the pixels which have low-intensity neighborhoods

Exercise Chapter 4 – Morphological Filtering of grayscale images

In this exercise we want to extend the morphological filtering to the grayscale images. Load the grayscale image *CAMERAMAN.TIF* et update the path list in the path browser.

♦ Note: we consider for the binary images that the objects are set to 1 and the background is set to 0, therefore we must often inverse the binary natural images. In the case of the grayscale natural images, the objects are rather dark and the background is rather white. We perform also an inverse video transformation with the Matlab function ***imcomplement*** (type *help imcomplement* fore more information). We obtain thus white objects and a dark background.

## Morphological filtering of grayscale images

1 – Erode the image *Cameraman.tif* (***imerode***) by the following structuring element: SE=***strel***('ball',5,5).
Display the result and compare with the original image. Define this operator thanks to the Matlab ***Image Processing toolbox***.

2 – In the same way, dilate the original image *Cameraman.tif* (***imdilate***) by the same structuring element. Display the result and compare with the original image. Use the Matlab *Image Processing toolbox* for defining this operator.

3 – Perform a morphological opening then a morphological closing of the original image *Cameraman.tif*. Observe and interpret the obtained results.

# Exercise solution: Morphological filtering of grayscale images

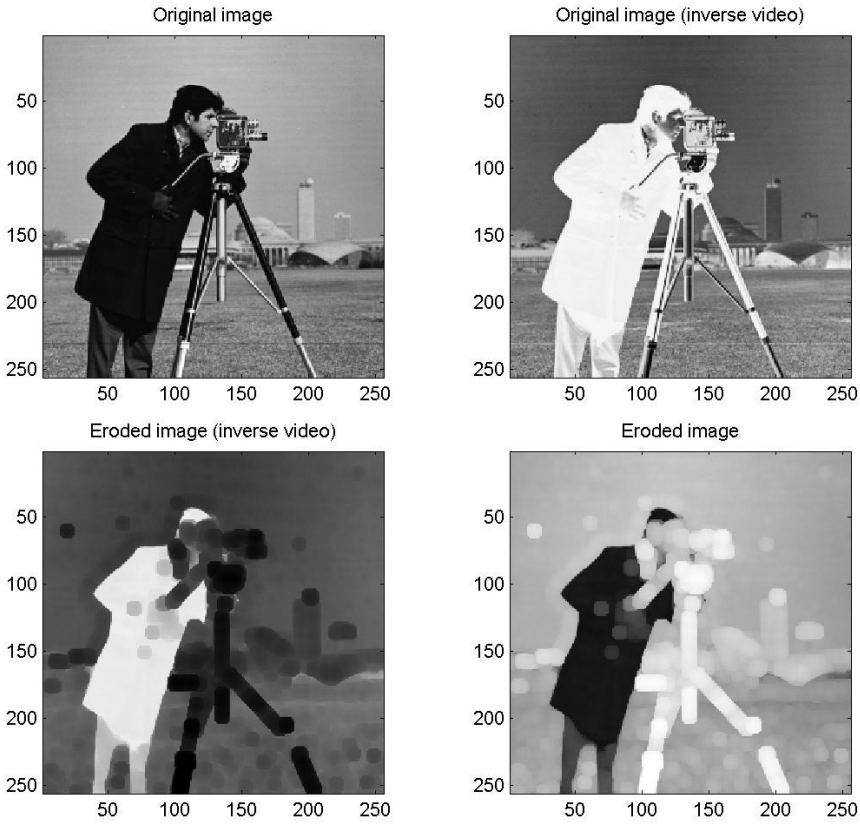
1 – Here are the commands for eroding the original image and for comparing the result with the original image:

```
% Image loading
I = imread('cameraman.tif') ;
Ic = imcomplement(I); % inverse video of original image
figure(1)
subplot(1,2,1)
subimage(I)
title('Original image');
subplot(1,2,2)
subimage(Ic)
title('Original image (inverse video)');

% Erosion
SE = strel('ball',5,5) % structuring element
Ierod = imerode(Ic,SE);
figure(2)
subplot(1,2,1)
subimage(Ierod)
title('Eroded image (inverse video)');

% Display the eroded grayscale image without inverse video effect
subplot(1,2,2)
subimage(imcomplement(Ierod))
title('Eroded image');
```

Here are the results:



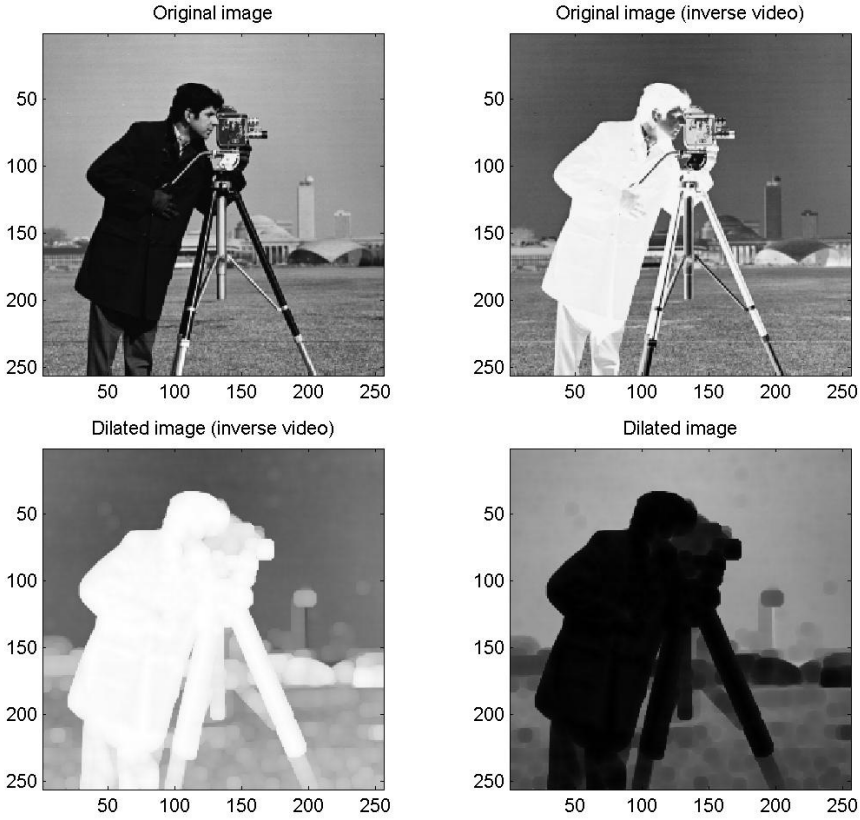

The erosion of grayscale images decreases the luminance value of the pixels which have low-intensity neighborhood (you can visualize this phenomenon on

the images with inverse video effect). The neighborhood depends on the structuring element. The pixels of the input image are scanned: each pixel is considered as the origin of the structuring element (by translation). **The output value of each pixel is then equal to the minimal value of its neighboring pixels**.

2 – Here are the commands for dilating the original image and for comparing the result with the original image:

```
% Dilation
figure(3)
Idilat = imdilate(Ic,SE);
subplot(1,2,1)
subimage(Idilat)
title('Dilated image (inverse video)');
subplot(1,2,2)
subimage(imcomplement(Idilat))
title('Dilated image');
```

We display the following results:

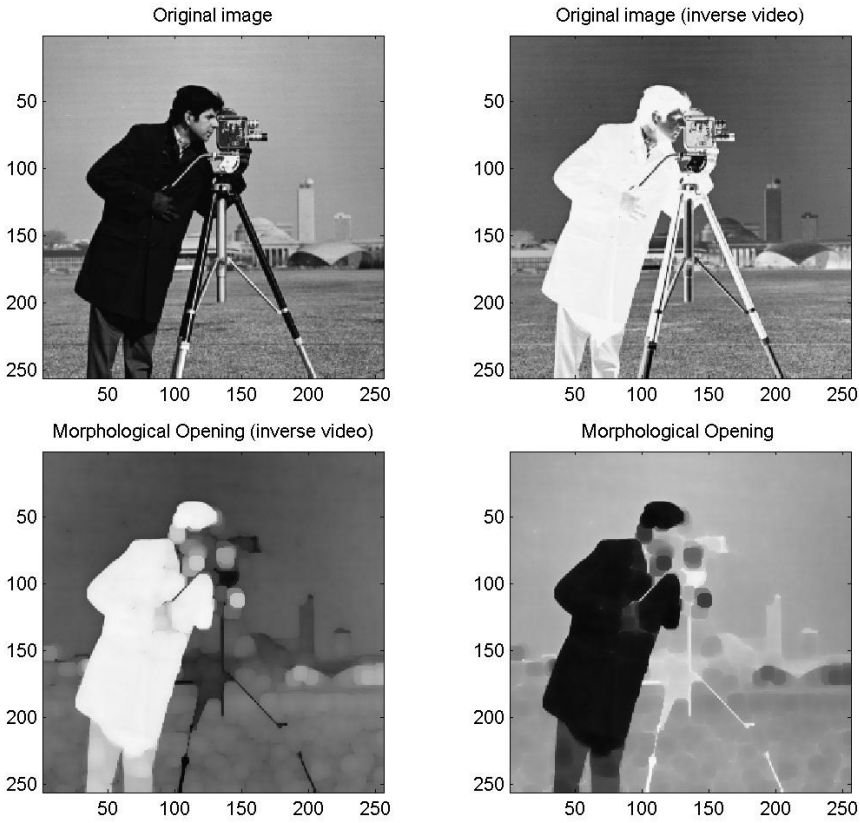



The dilation of grayscale images increases the luminance value of the pixels which have high-intensity neighborhood (you can visualize this phenomenon, on the images with inverse video effect). **The output value of each pixel is the maximal value of all its neighboring pixels.**

3 – Here is an example of a script for performing the morphological opening and for comparing the result with the original image:

```
% Morphological opening
figure(4)
Iopen = imopen(Ic,SE);
subplot(1,2,1)
subimage(Iopen)
title('Morphological Opening (inverse video)');
subplot(1,2,2)
subimage(imcomplement(Iopen))
title('Morphological Opening');
```

Here are the results:

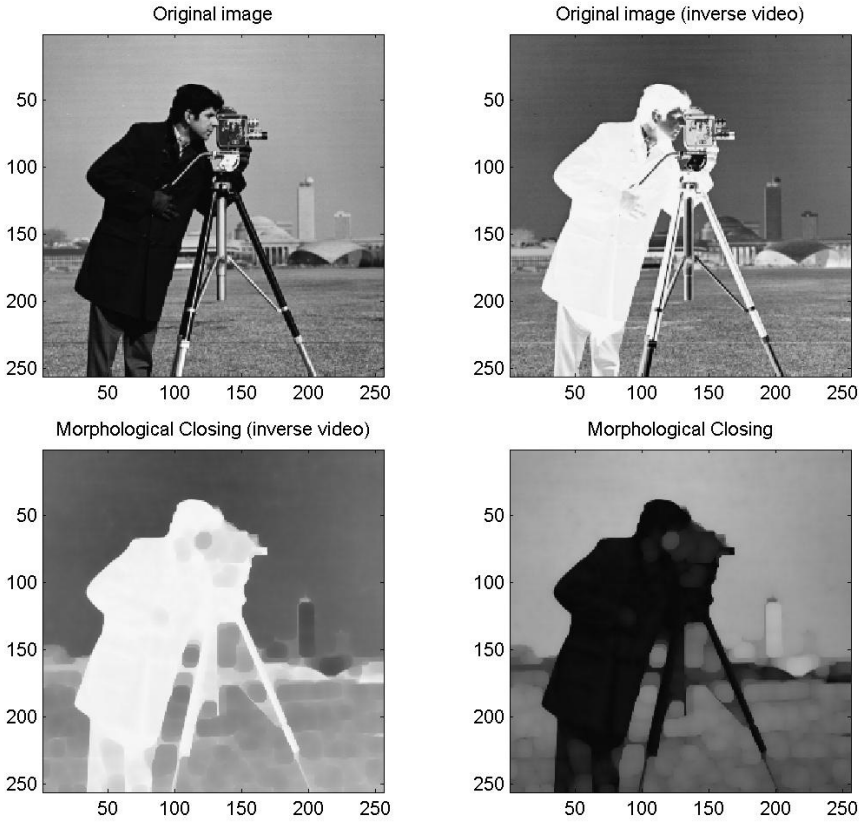



The morphological opening consists of erosion followed by the dilation of the original grayscale image. The morphological opening removes the white pixels which are isolated and smoothes the contours. We also visualize a segmentation of the objects within the image.

Here is an example of a script for performing the morphological closing:

```
% Morphological closing
figure(5)
Iclose = imclose(Ic,SE);
subplot(1,2,1)
subimage(Iclose)
title('Morphological Closing (inverse video)');
subplot(1,2,2)
subimage(imcomplement(Iclose))
title('Morphological Closing');
```

Here are the images displayed:



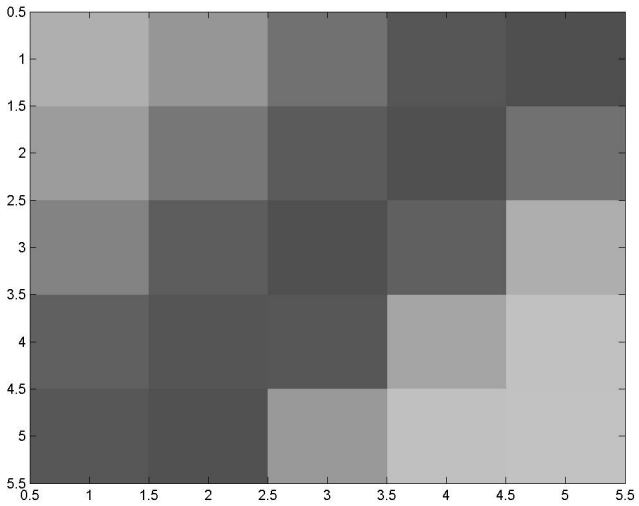

The morphological closing consists of dilation followed by the erosion of the original grayscale image. The morphological closing removes the dark pixels which are isolated and smoothes the contours.

## Chapter 4 – Non-linear Filtering

# TEST

## Exercise 1:

We want to perform several different filters for processing the following $5 \times 5$ image:




This image is a small portion of the original image "*Boats*" and contains a part of the fishing boat's boom. Here is the $5 \times 5$ matrix which contains the luminance values of this portion:

| | | | | |
|---|---|---|---|---|
| 175 | 150 | 114 | 86 | 79 |
| 156 | 119 | 91 | 80 | 113 |
| 132 | 93 | 80 | 96 | 174 |
| 96 | 85 | 87 | 165 | 193 |
| 87 | 82 | 153 | 192 | 194 |

We do not take account of the boundary effects and we consider only the results obtained by filtering of the $3 \times 3$ central pixels:

| | | | | |
|---|---|---|---|---|
| X | X | X | X | X |
| X | ? | ? | ? | X |
| X | ? | ? | ? | X |
| X | ? | ? | ? | X |
| X | X | X | X | X |

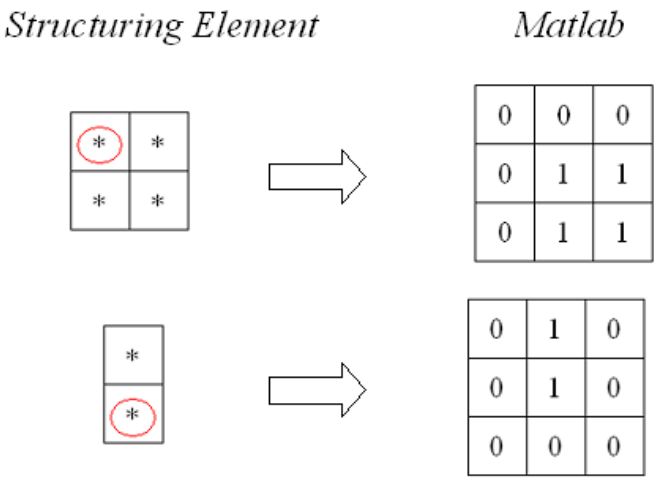
Fill this matrix for each of the following cases:

- median filtering using the full 3×3 neighborhood,
- median filtering using the 5-pixel cross-shaped neighborhood,
- erosion by a full 3×3 structuring element,
- dilation by a full 3×3 structuring element,
- morphological closing by a 5-pixel cross-shaped structuring element..

## Exercise 2:

In Matlab, create the functions for dilating and for eroding an image (do not take account of the boundary distortions). These two functions have two optional inputs: the original image that we want to filter and the structuring element. The structuring element's size will be less than or equal to $3 \times 3$.

- Examples:




By using your two new functions, create the functions for performing the morphological closing and the morphological opening.

Test your four functions by processing the image "*Boats*". Compare the results with the results that you obtain with the Matlab functions ***imdilate***, ***imerode***, ***imclose***, and ***imopen***.