Chapter 7: The noise

The noise

Definition

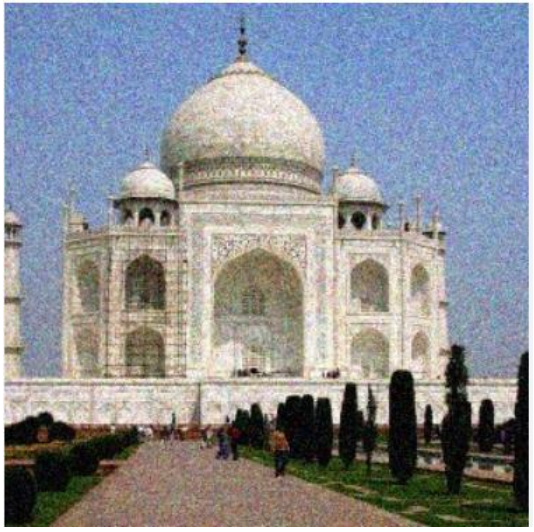
- Noise or parasite is defined as a sudden variation of the intensity of a pixel compared to its neighbors
- Noise is a stochastic (random), non-continuous signal
- The noise elimination operation is called filtering.

2	3	4	4	6
1	2	4	5	6
0	1	**192**	3	4
0	1	2	3	4
1	3	2	1	5

How to distinguish noise from edges?

The noise

“ The main barrier to effective image processing and signal processing in general is noise. By noise we mean a variation of the signal from its true value by a small (random) amount due to external or internal factors “



The noise

(a) The original image

(b) After removing the blur

The noise

Sources

1.Capture noise

“can be the result of variations in lighting, sensor temperature, electrical sensor noise, sensor nonuniformity, dust in the environment, vibration, lens distortion, focus limitations, sensor saturation (too much light), underexposure (too little light)”. Nearby electronic interference

2. Image-encoding noise

“lossy compression techniques (eg, jpeg) compress the image by removing visual information that represents detail not general perceivable to the human viewer. This loss of detail is often referred by the appearance of compression artefacts (noise) in the image”.

3. Sampling/ Quantization noise

4. Transmission errors

The noise

Types

The generation of a noisy signal, which will be added to an image, is useful for testing the robustness of filters.

Several types of noise, the most known are:

1. **Gaussian noise (additive noise)**
2. **Impulse noise (Salt and pepper)**

The noise

1. Gaussian noise

New pixel = Old pixel + random number following the Gaussian or normal law.

« The **normal (Gaussian) law** is one of the most suitable probability laws for modeling natural phenomena resulting from several random events»

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

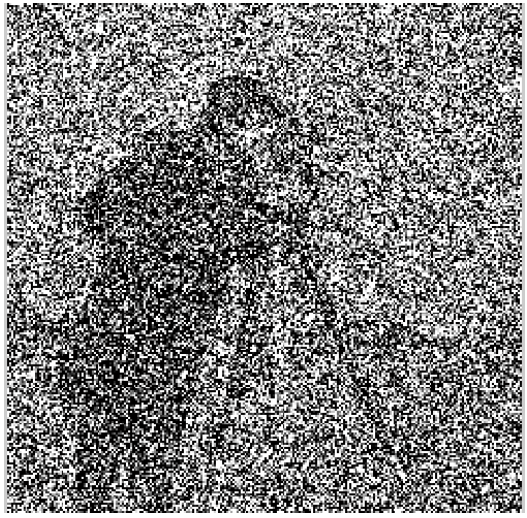
U: mean
σ: standard deviation
X: pixel gray level

The noise

original Image

σ= 0,01

σ= 0,1



σ= 0,9

The noise

2. Impulse noise

- Sudden changes in pixel value that are visible as random black and white pixels
- Sources: transmission errors, camera defect, dust…..

original Image

5%

20%

Chapter 8: Filtering

Filtrage

Definition

The main goal of **filtering** is to remove noise from image

→ enhanced image with greater clarity → improve visualization of its contents

Noised image

Filtred image

Filtering

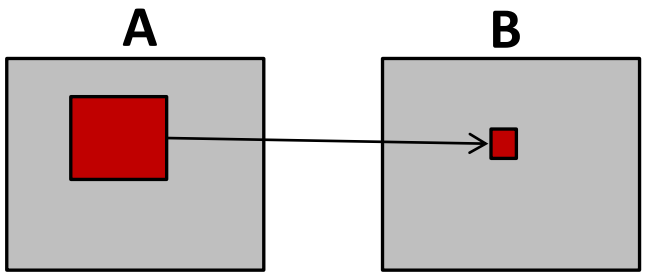
Remarks

- There is no universal filter capable of correcting all defects
- We choose the type of the filter according to the defects that we wish to correct.

Type of operations

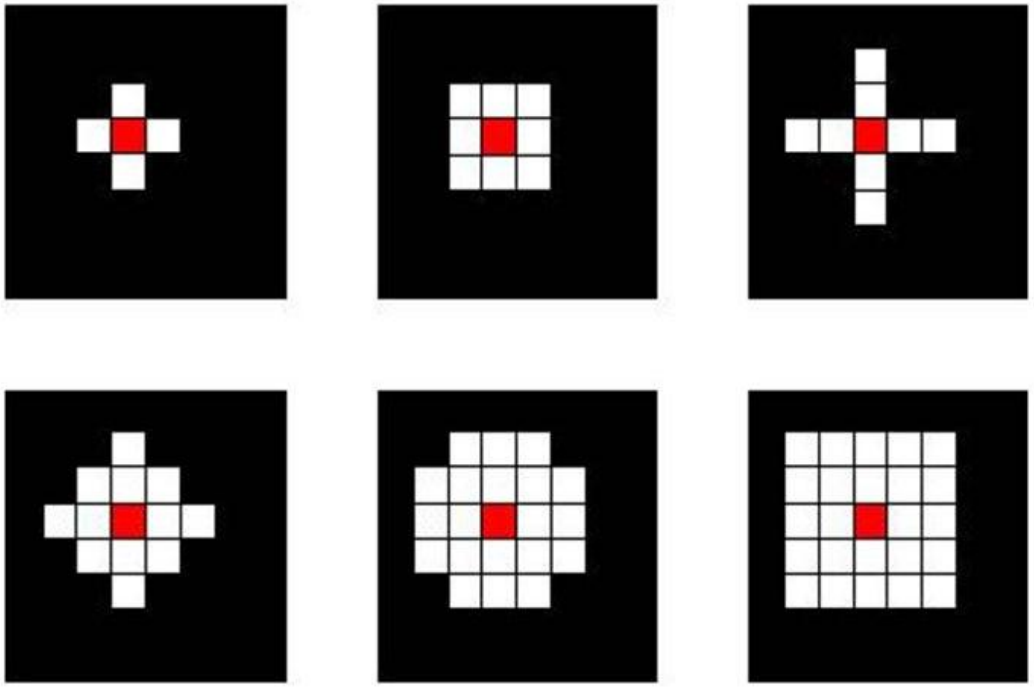
Local to point transformation

- The pixel **B(i,j)** of the output image, depends only on the values of all pixels of the mask (neighbourhood)



Filtering

Neighborhood examples :



Note: the neighborhood is often square and symmetrical (odd-sized matrix) around the pixel considered

Example: 3x3 5x5 ……..

Filters types

1. Linear filters

The output pixel (filtred image) is a linear combination of the pixels in a local neighborhood (**mask)** in the original image (noised image)

1.1 Low-pass filter

1.2 High-pass filter (see chapter 9)

2. Nonlinear filters

The output pixel (filtred image) is selected from an ordered sequence in a local neighborhood (**mask)** in the original image (noised image)

Linear filters

1.1 Low-pass filter (smoothing)

- Consisting of attenuating the components of the image having a high frequency.
- These filters tend to blur the image
- **Examples**: The mean filter and the Gaussian filter

1.1.1 Mean filter

- The gray level of the central pixel is replaced by **the average** of the gray levels of the neighboring pixels (mask)

- Convolution !!!

Filtering

Convolution

- A convolution product is a mathematical operator used to multiply matrices together.

Example

$$\begin{pmatrix} H_1 & H_2 & H_3 \\ H_4 & H_5 & H_6 \\ H_7 & H_8 & H_9 \end{pmatrix}$$

Mask 3x3

$$\begin{pmatrix} I_{x-1,y-1} & I_{x-1,y} & I_{x-1,y+1} \\ I_{x,y-1} & I_{x,y} & I_{x,y+1} \\ I_{x+1,y-1} & I_{x+1,y} & I_{x+1,y+1} \end{pmatrix}$$

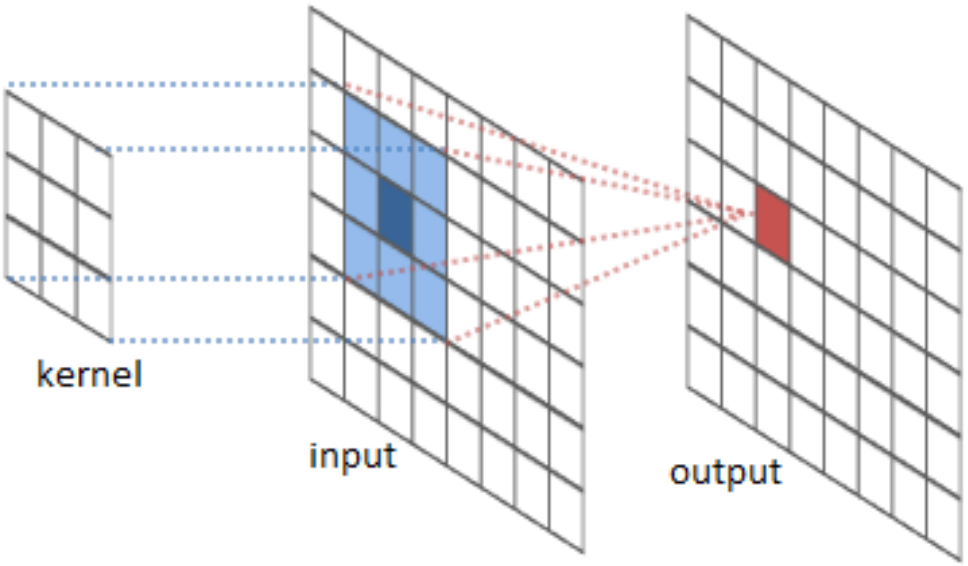
Image

The new value of pixel $I_{x,y}$ after convolution:

$$I_{x,y} = H_1 * I_{x-1,y-1} + H_2 * I_{x-1,y} + H_3 * I_{x-1,y+1} + H_4 * I_{x,y-1} +$$
$$H_5 * I_{x,y} + H_6 * I_{x,y+1} + H_7 * I_{x+1,y-1} + H_8 * I_{x+1,y} + H_9 * I_{x+1,y+1}$$

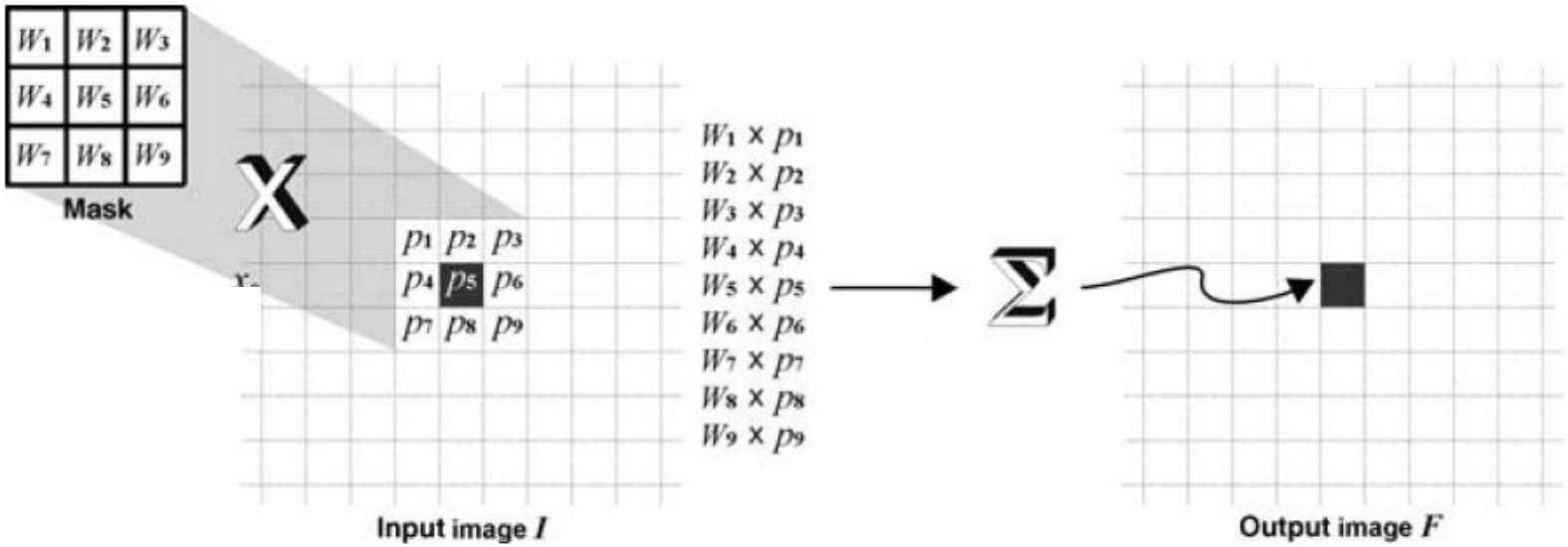
Filtering

Convolution



Mean filter

Convolution



- The new value of each pixel after convolution is saved on a copy of the image (**output**) and not on the original image (**input**).
- Why is the kernel size odd?

To center the kernel on the pixel

Mean filter

Convolution

Kernel 3x3

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

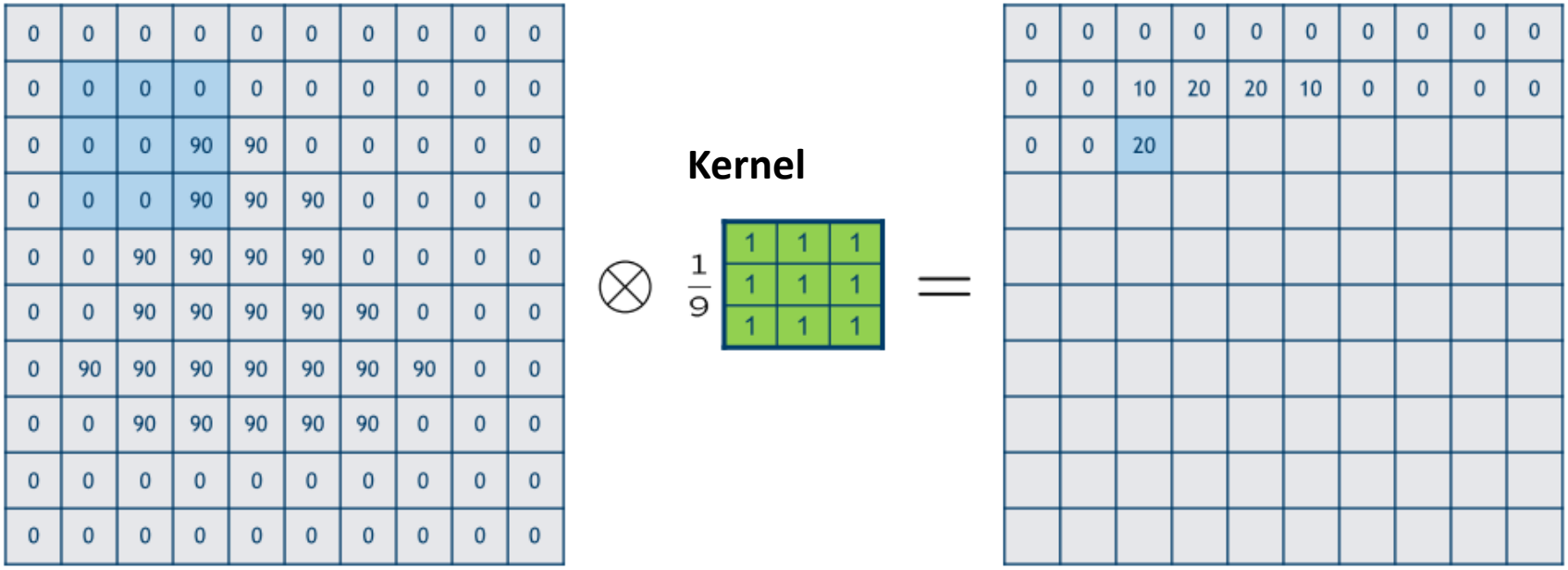
Kernel 5x5

$$\frac{1}{25}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

Mean filter

Convolution

Example

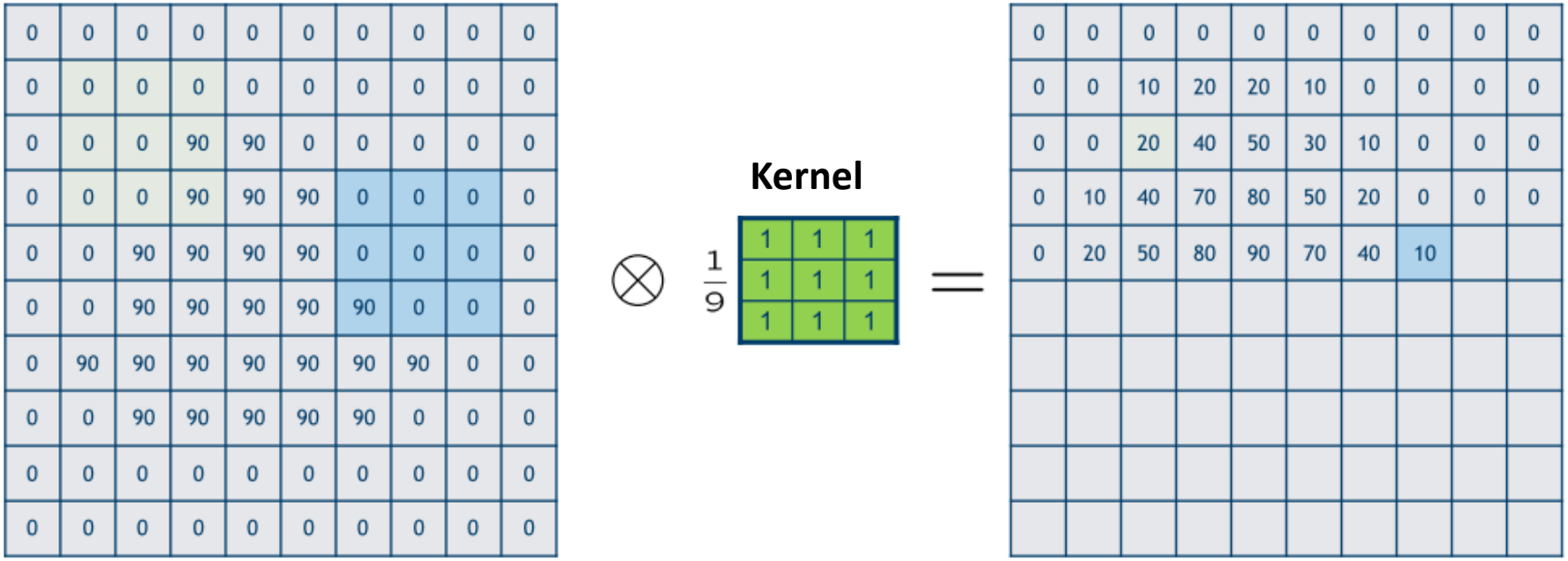


Input image **Output image**

Mean filter

Convolution

Example



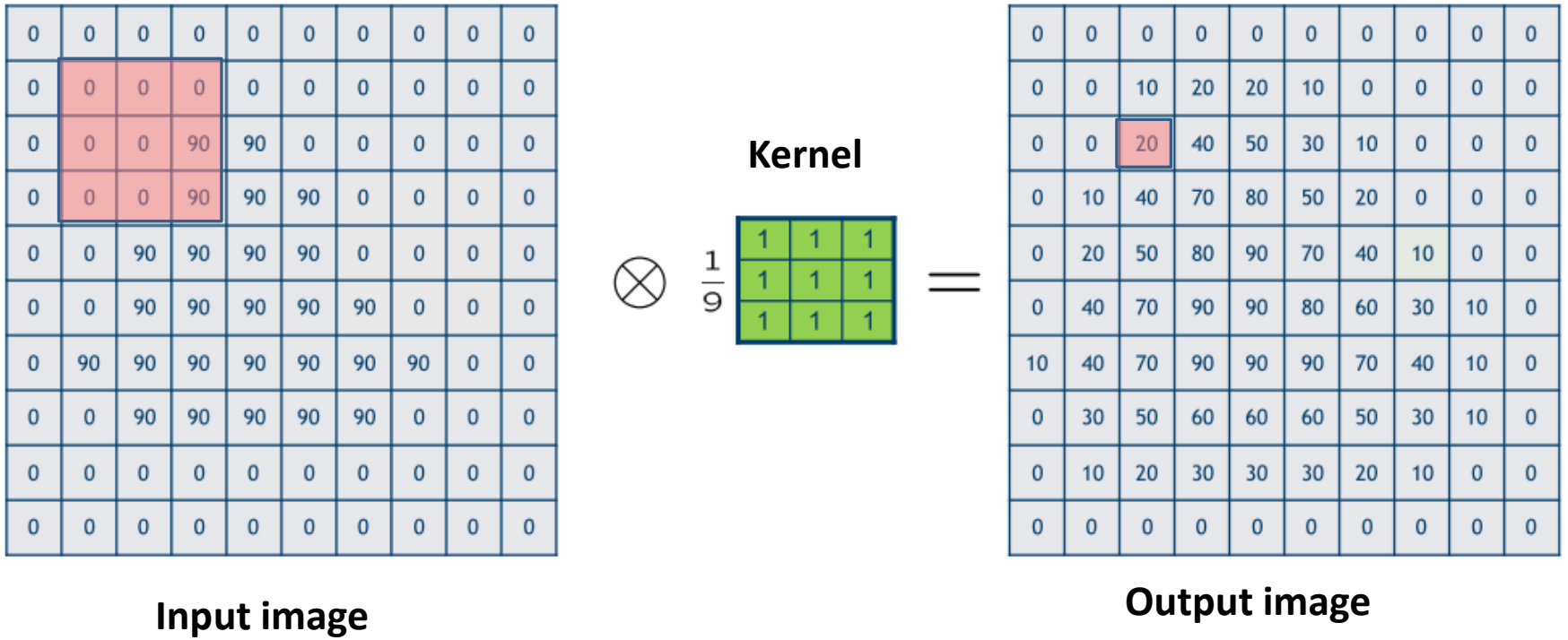
Input image

Output image

Mean filter

Convolution

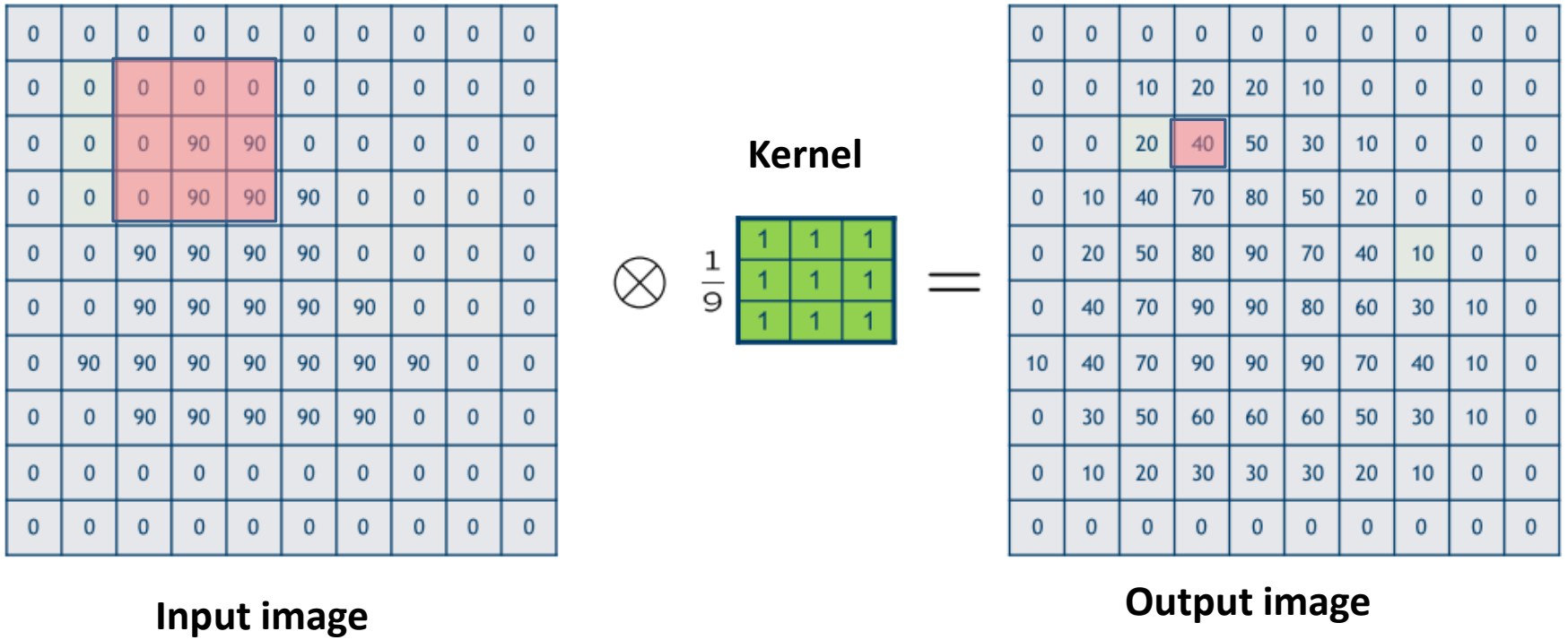
Example



Mean filter

Convolution

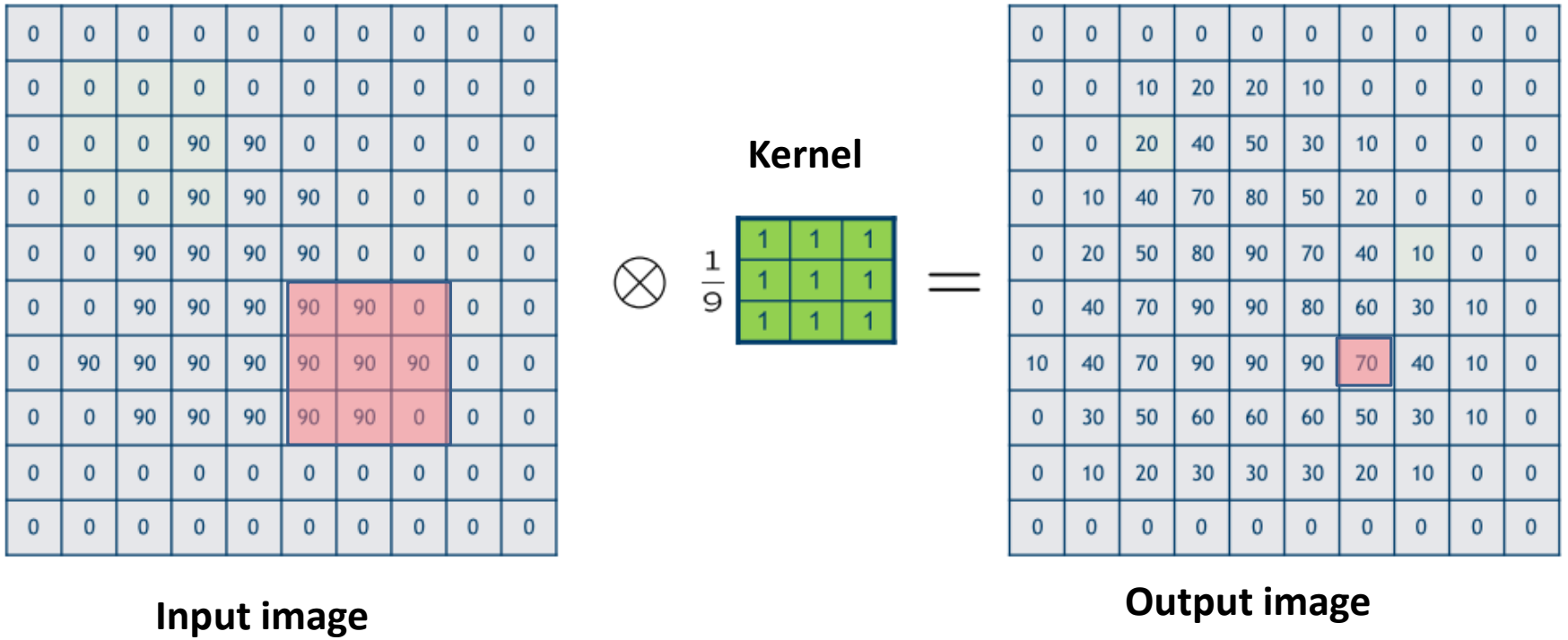
Example



Mean filter

Convolution

Example



Mean filter

Convolution

Algorithm

1. Position the mask over the current pixel,

2. Form all products of filter elements with the corresponding elements of the neighbourhood

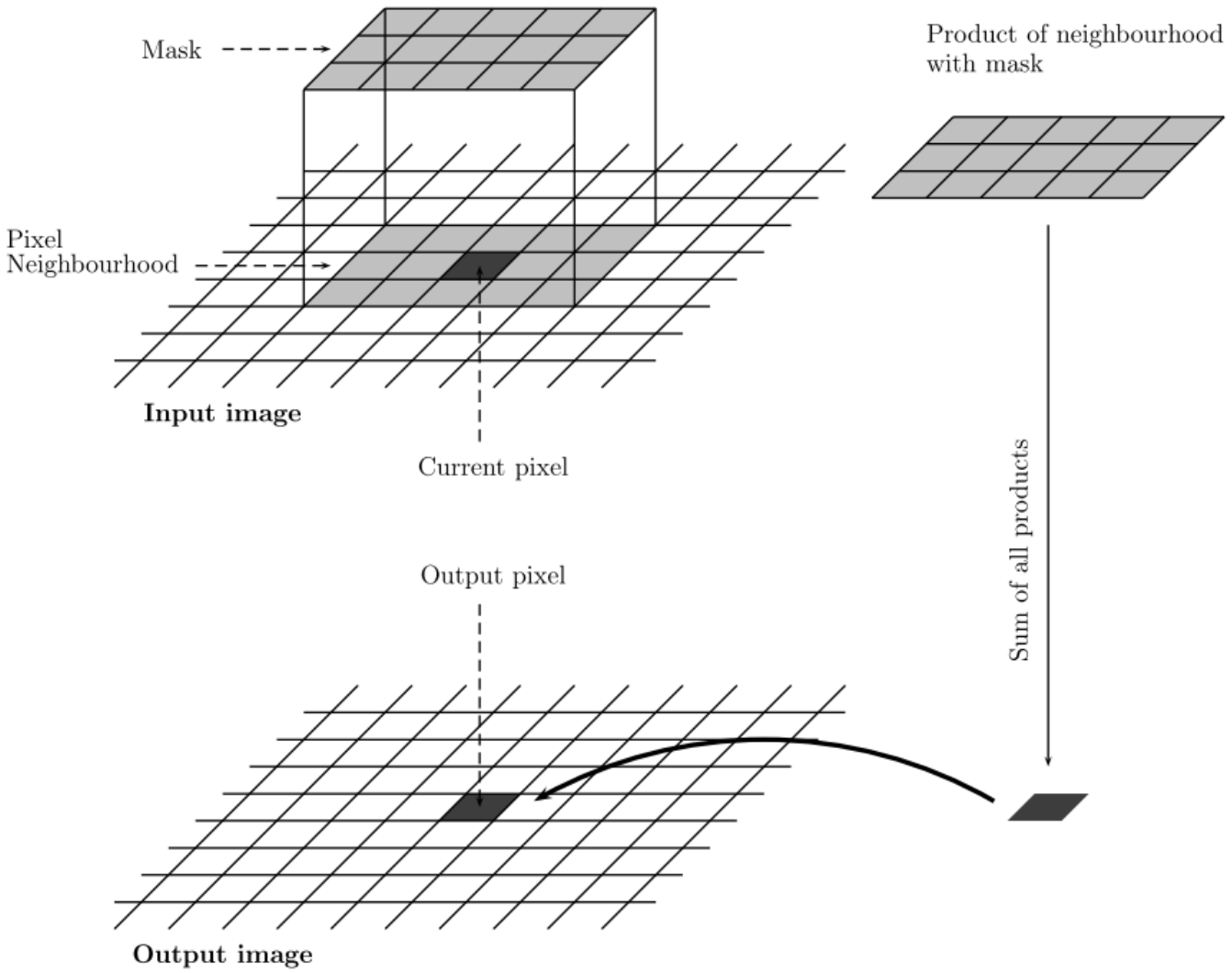
3. add up all the products.

4. Repeat steps 1, 2 and 3 for every pixel in the image.

Mean filter

Convolution

kernel of 3x5



Mean filter

Problem of edges? **Padding Techniques**

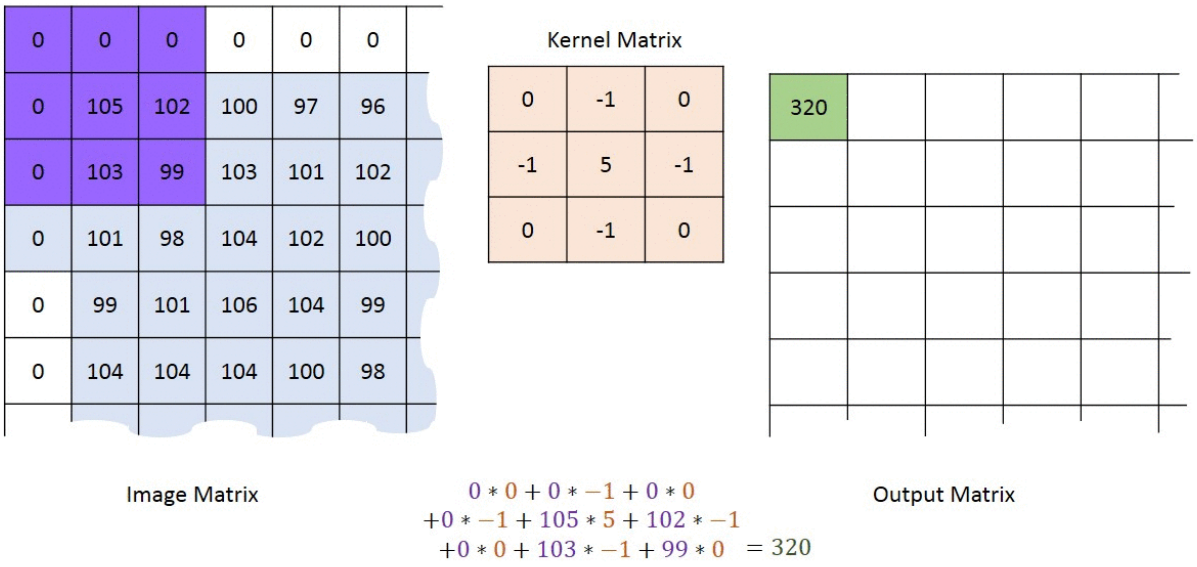
1. Zero padding : zero padding is used to add extra rows and columns of zeros to the edges of an image.

0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	0
0	7	8	9	10	11	12	0
0	1	2	3	4	5	6	0
0	7	8	9	10	11	12	0
0	1	2	3	4	5	6	0
0	7	8	9	10	11	12	0
0	0	0	0	0	0	0	0

Mean filter

Problem of edges? **Padding Techniques**

1. Zero padding : zero padding is used to add extra rows and columns of zeros to the edges of an image.



Mean filter

Problem of edges? **Padding Techniques**

2. Mirror padding : called also symmetric padding, it adds a boundary around the image by mirror-reflecting.

Example : kernel 3x3

6	5	6	7	8	7
2	1	2	3	4	3
6	5	6	7	8	7
10	9	10	11	12	11
14	13	14	15	16	15
10	9	10	11	12	11

Mean filter

Problem of edges? **Padding Techniques**

2. Mirror padding : called also symmetric padding, it adds a boundary around the image by mirror-reflecting.

Example : kernel 5x5

11	10	9	10	11	12	11	10
7	6	5	6	7	8	7	6
3	2	1	2	3	4	3	2
7	6	5	6	7	8	7	6
11	10	9	10	11	12	11	10
15	14	13	14	15	16	15	14
11	10	9	10	11	12	11	10
7	6	5	6	7	8	7	6

Mean filter

Problem of edges? **Padding Techniques**

3. Replicate padding : values outside the boundary are set equal to the nearest image border value.

Example : kernel 3x3

1	1	2	3	4	5	5
1	1	2	3	4	5	5
6	6	7	8	9	10	10
11	11	12	13	14	15	15
16	16	17	18	19	20	20
16	16	17	18	19	20	20

Mean filter

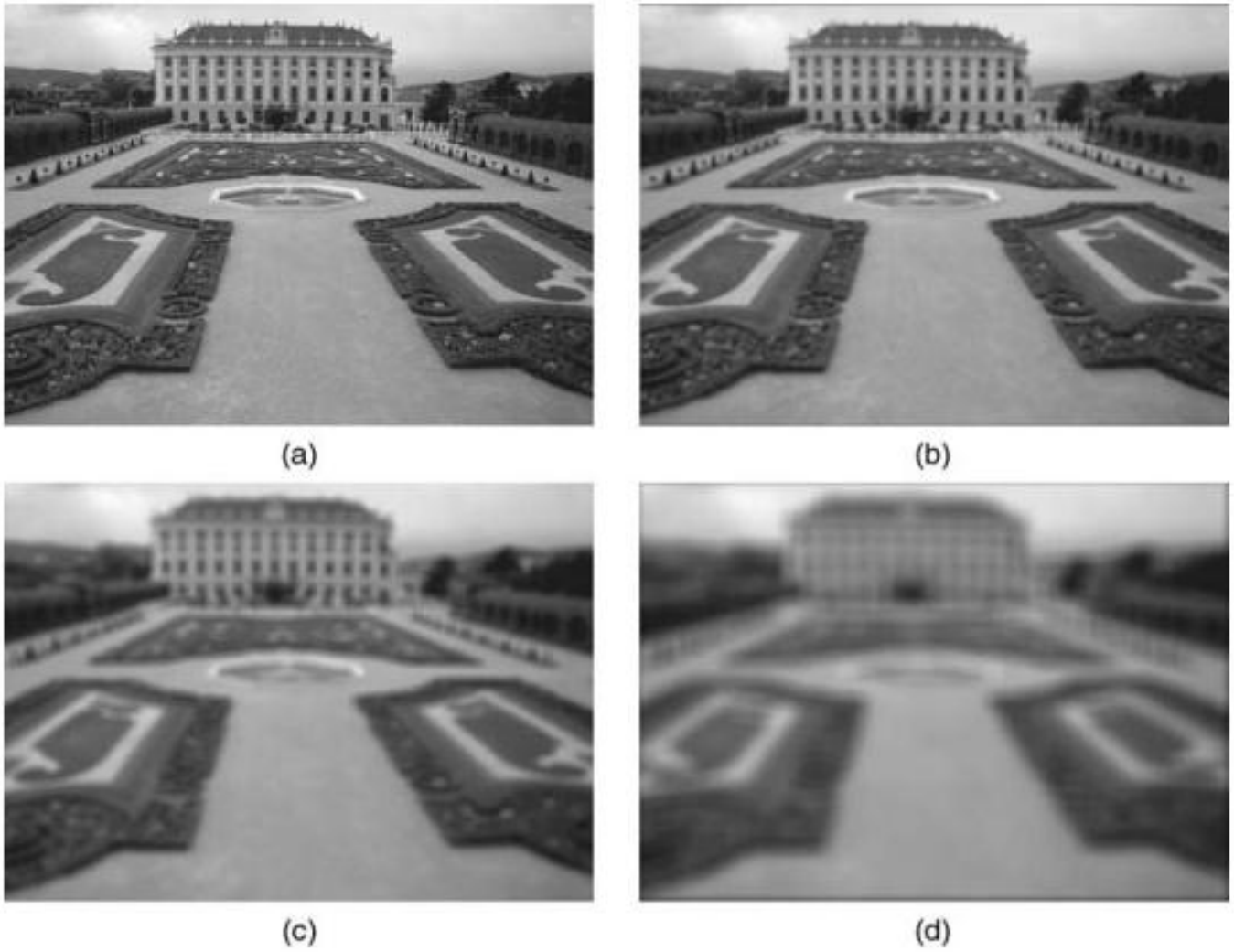
Problem of edges? **Padding Techniques**

3. Replicate padding : values outside the boundary are set equal to the nearest image border value.

Example : kernel 5x5

1	1	1	2	3	4	5	5	5
1	1	1	2	3	4	5	5	5
1	1	1	2	3	4	5	5	5
6	6	6	7	8	9	10	10	10
11	11	11	12	13	14	15	15	15
16	16	16	17	18	19	20	20	20
16	16	16	17	18	19	20	20	20
16	16	16	17	18	19	20	20	20

Mean filter

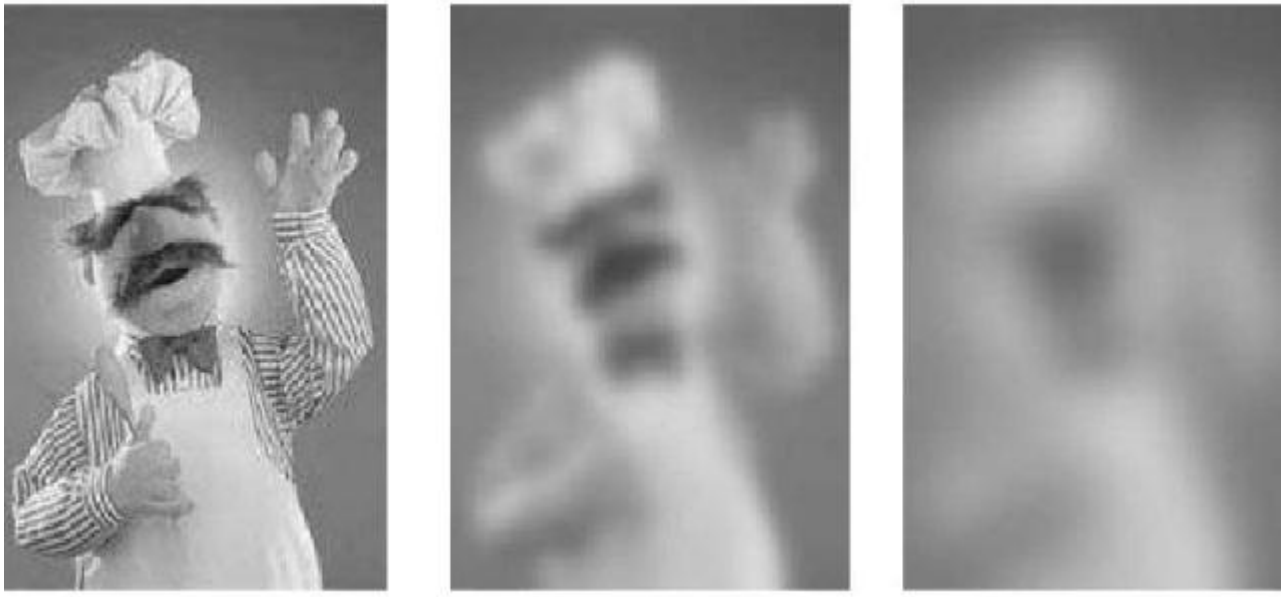


Examples of applying the averaging filter with different mask sizes: (a) input image (899 × 675 pixels); (b–d) output images corresponding to averaging masks of size 7 × 7, 15 × 15, and 31 × 31.

Mean filter

Disadvantages

- It is not robust to large noise deviations in the image (outliers)
- It blurs the image if the kernel size becomes large.
- Fine details are erased and contours are not preserved



1.1.2 Gaussian filter

- ***Kernel calculation:***

Kernel coefficients *h* are calculated using Gaussian function

$$h(x, y) = \exp\left[\frac{-(x^2 + y^2)}{2\sigma^2}\right]$$

$$h = \begin{pmatrix} h_{-1,-1} & h_{-1,0} & h_{-1,+1} \\ h_{0,-1} & h_{0,0} & h_{0,+1} \\ h_{+1,-1} & h_{+1,0} & h_{+1,+1} \end{pmatrix}$$

σ: standard deviation

x,y: coordinates of each kernel element

Gaussian filter

Example : kernel 3x3 (sigma = 1)

0.0751	0.1238	0.0751
0.1238	0.2042	0.1238
0.0751	0.1238	0.0751

Example : Kernel 5x5 (sigma =1)

0.0030	0.0133	0.0219	0.0133	0.0030
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0030	0.0133	0.0219	0.0133	0.0030

Gaussian filter

- Give more importance to the center pixels(coefficients are not equivalent)
- Neighborhood pixels that are close to the central pixel have a stronger weight (more influence) than those that are further away→ weight decreases monotonically with distance from the central pixel
- Contours and fine details are better preserved than the average filter.

Gaussian filter

- Gaussian filter is rotationally symmetric:

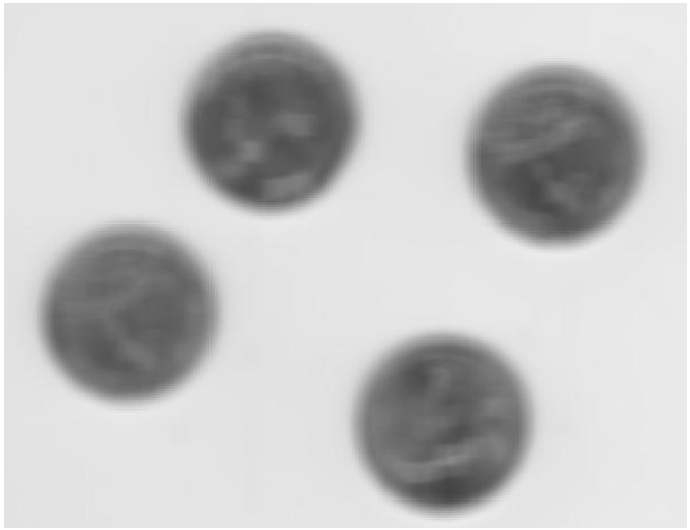
this means that the amount of smoothing performed by the filter will be the same in all directions.

Example : Kernel 5x5 (sigma =1)

0.0030	0.0133	0.0219	0.0133	0.0030
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0030	0.0133	0.0219	0.0133	0.0030

Filtering

Mean filter 3x3



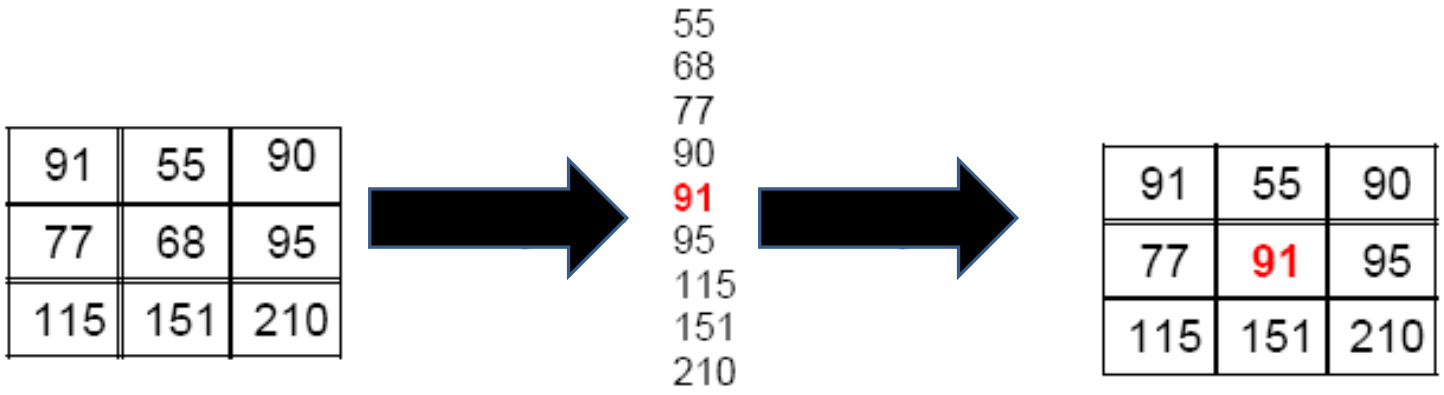
Mean filter 9x9

Gaussian filter 3x3, $\sigma=1$

Gaussian filter 9x9, $\sigma=1$

2.1 Median filtering

- The gray level of the central pixel is replaced by the median value of all neighboring pixels



Nonlinear filters

2.1 median filtering

Algorithm

1. Sorting the pixel values within a neighborhood

2. Finding the median value

3. Replacing the original pixel value with the median of that neighborhood.

4. Repeat steps 1, 2 and 3 for every pixel in the image

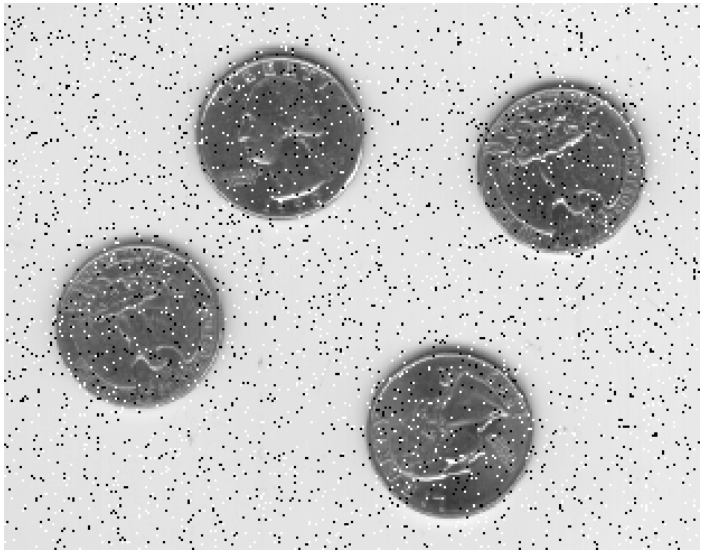
2.1 Median filtering

Advantages

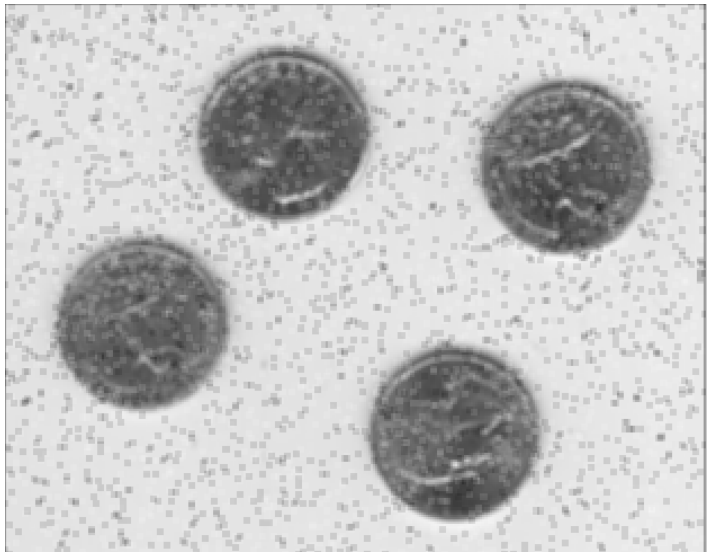
- It preserves edges for small filter sizes
- It removes outliers
- Works extremely well in removing salt and pepper noise

Filtering

Original image



impulse noise 5%



Mean filtre 3x3

Median filter 3x3

2.2 Other nonlinear filters

Replace the central pixel (Pc) with a function of neighboring pixels selection:

- **KNN** (**k-nearest neighbors**)
- **Sigma**
- **SNN** (**Symmetric nearest neighbors**)
- **Min/Max**
- **NAGAO filter**
- ……

Nonlinear filters

2.2 Other nonlinear filters

Replace the central pixel (Pc) with a function of a selection of neighboring pixels:

- **KNN** (**k-nearest neighbors**) : **Sigma** : les pixels situés dans [I(x,y)- σ, I(x,y)+ σ] $\boldsymbol{\sigma}$: l'écart-type de l'image
- **SNN** (**Symmetric nearest neighbors**) : pour chaque paire de pixels symétriques par rapport à Pc, on sélectionne celui qui est le plus proche de Pc en valeur
- **Min/Max** : si le Pc est plus grand (plus petit) que le Max (Min) des voisins, on attribue la valeur Max (Min), sinon il est inchangé
- **NAGAO filter**
- ……

Examples

- **KNN (k-nearest neighbors)**

the **k** closest pixels in value

Image

10	12	40	16	19	10
14	22	52	10	55	41
10	14	51	21	14	10
32	22	9	9	19	14
41	18	9	22	27	11
10	7	8	8	4	5

Ex : k= 4

{9, 9, 14, 19} => moyenne = 13

Examples

- **Sigma**

pixels located in [I(x,y)- σ, I(x,y)+ σ]

σ : the standard deviation of the image

Image

10	12	40	16	19	10
14	22	52	10	55	41
10	14	51	21	14	10
32	22	9	9	19	14
41	18	9	22	27	11
10	7	8	8	4	5

Ex : $\sigma = 13 \Rightarrow [0 , 22]$

{ 9, 9, 14, 19, 21, 22 } => moyenne = 16

Filtering

Examples

- **SNN**

for each pair of pixels symmetrical with central pixel Pc, we select the one which is closest of PCs in value

Image

10	12	40	16	19	10
14	22	52	10	55	41
10	14	51	21	14	10
32	22	9	9	19	14
41	18	9	22	27	11
10	7	8	8	4	5

$\{9, 14\} \Rightarrow 9$
$\{9, 19\} \Rightarrow 9$
$\{51, 27\} \Rightarrow 27$
$\{21, 22\} \Rightarrow 21$

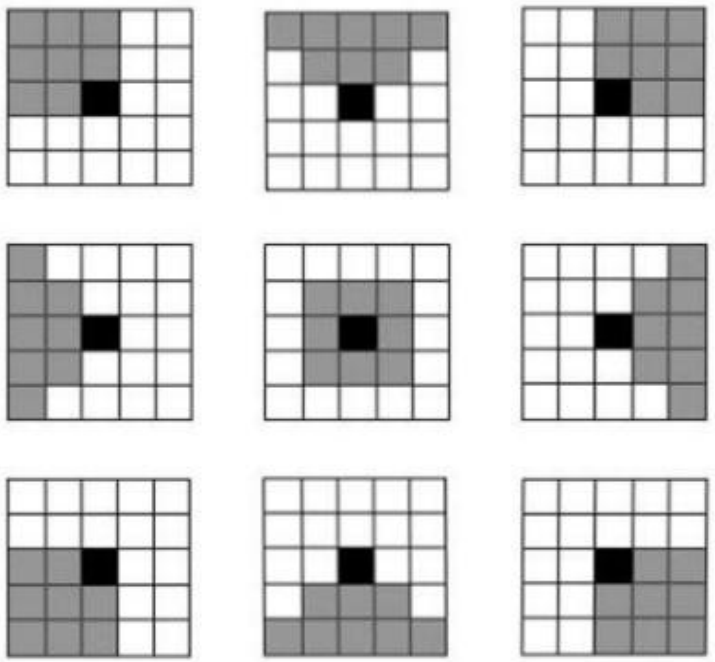
Moyenne = 16

Filtering

Examples

- **NAGAO**

- Divide the neighborhood into several zones
- Select the most homogeneous zone (minimum σ)
- Calculate the average of the selected area and assign it to the PC



Filtering

Measures

"The effectiveness of a filtering operation is quantified using performance metrics (evaluation). ……..

The most common performance metrics used in quantifying filtering operation are a peak signal to noise ratio (PSNR) and mean square error (MSE) . PSNR and MSE are widely used in performance metrics because they are simple and easy to use."

Filtering

Measures

- Mean-squared error (MSE) best value = 0

"*MSE measures the average squared difference between actual and ideal pixel values. This metric is simple to calculate but might not align well with the human perception of quality*"

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

Matlab/ Octave : immse (A, B)

Filtering

Measures

- Peak signal-to-noise ratio (PSNR) best value = inf

"*PSNR is derived from the mean square error, and indicates the ratio of the maximum pixel intensity to the power of the distortion. Like MSE, the PSNR metric is simple to calculate but might not align well with perceived quality.*"

$$PSNR = 10 \cdot \log_{10}\left(\frac{d^2}{\text{MSE}}\right)$$

d: max value of pixel (255 in gray scale, 1 binary image)

Matlab/ Octave : psnr(A, B)

Measures

Note:

For RGB image, calculate average mean square error of R, G, B

MSE= (mseR + mseG + mseB)/3;

Chapter 9: Edge detection

Edge detection

Definition

Edges are useful to :

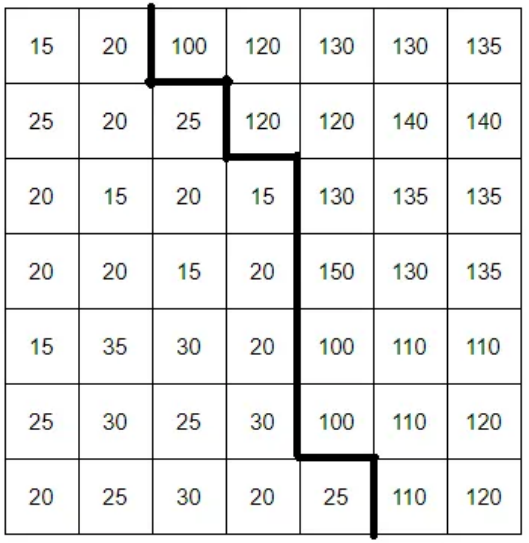
- Measure the size of objects in an image
- Isolate particular objects from their background
- Recognize or classify objects

Edge detection

Definition

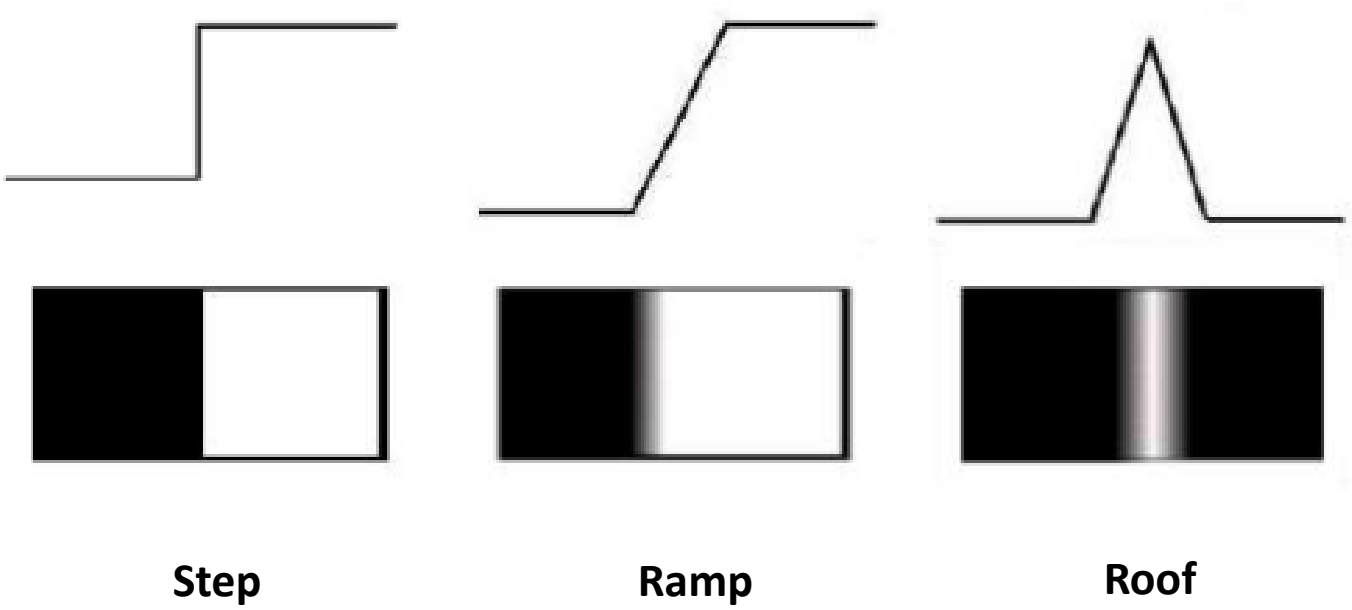
- An edge is defined as a local discontinuity in the pixel values
- An edge is a boundary between two image regions having distinct characteristics

15	20	100	120	130	130	135
25	20	25	120	120	140	140
20	15	20	15	130	135	135
20	20	15	20	150	130	135
15	35	30	20	100	110	110
25	30	25	30	100	110	120
20	25	30	20	25	110	120



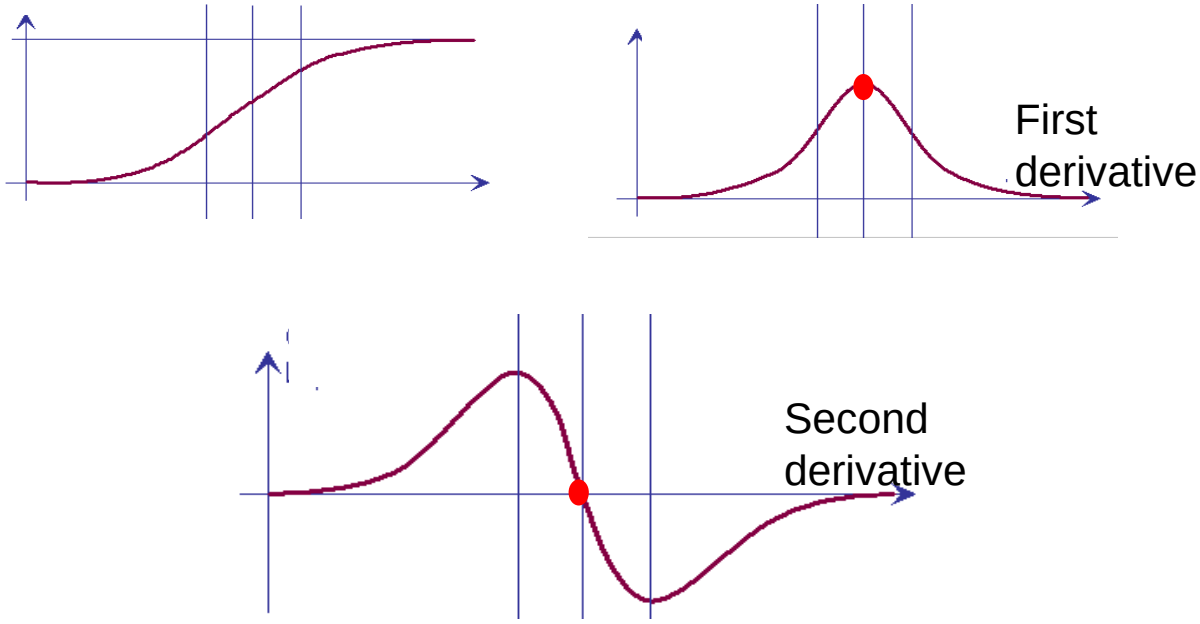
Edge detection

Types



Edge detection

- Contours are characterized by discontinuities (variations) in the function intensity
- Edge detection is based on differentiation (derivative)
- A variation will exist if the first derivative (gradient) is locally maximum or if the second derivative (Laplacian) presents a zero crossing



Edge detection

Definition of derivative

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- In an image, the smallest possible value of h is 1, being the difference between the index values of two adjacent pixels

The discrete version of the derivative is :

$$f(x+1) - f(x)$$

Edge detection

Other expressions for the derivative are:

$$\lim_{h \to 0} \frac{f(x) - f(x-h)}{h}, \qquad \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

The discrete versions are:

$$f(x) - f(x-1), \qquad (f(x+1) - f(x-1))/2$$

Edge detection

1. Gradient approch

- We consider the image as a function *f* with two variables
- We define two partial derivatives, along x-axis (lines) and along y-axis (columns)

$$\left[\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y}\right]$$

Edge detection

1. Gradient approch

- From image *f*, we calculate **Gh** and **Gv** corresponding to the filtering of *f*, by the two masks h and v
- **Gh** and **Gv** contain the horizontal and vertical contours
- The magnitude of the gradient at each pixel is given by

$$G = \sqrt{GH^2 + GV^2}$$

- The final image G is a binary image

Edge detection

1. Gradient approch

Algorithm

1. Calculate **Gh** (convolution between image *f and mask h)*

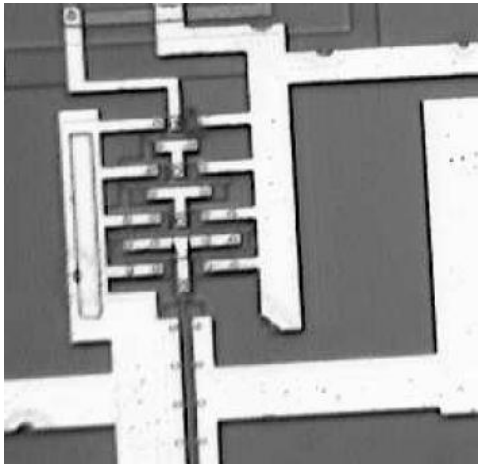
2. Calculate **Gv** (convolution between image *f and mask v)*

3. Calculate the gradient magnitude $G = \sqrt{GH^2 + GV^2}$

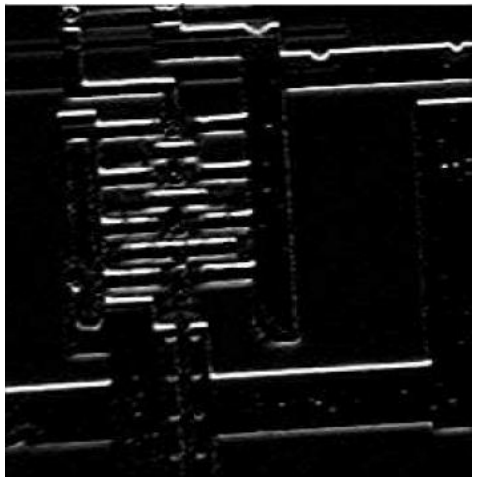
4. Thresholding the image G

Edge detection

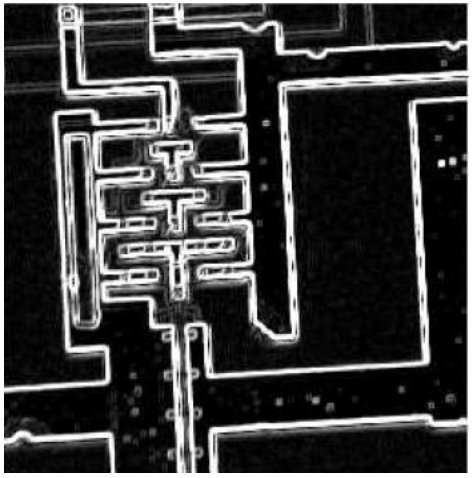
Example



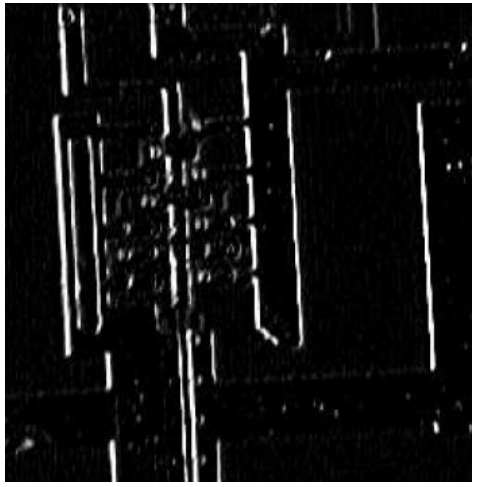
Original image



Horizontal contours



Result (magnitude)



Vertical contours

Edge detection

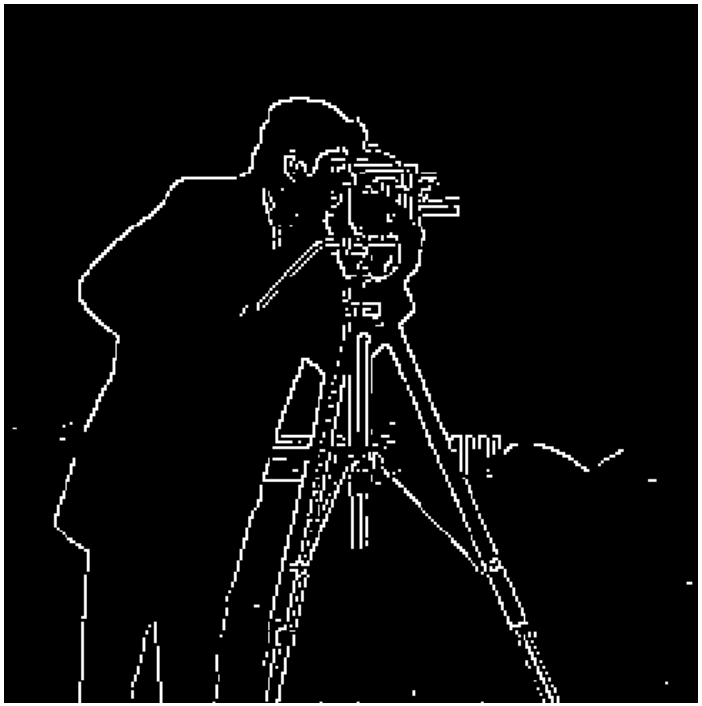
1.1. Roberts filter

GH = $\frac{\partial f(x, y)}{\partial x} = f(x+1,y+1)- f(x,y)$

GV = $\frac{\partial f(x, y)}{\partial y} = f(x+1,y)- f(x,y+1)$

H $= \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

V $= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$



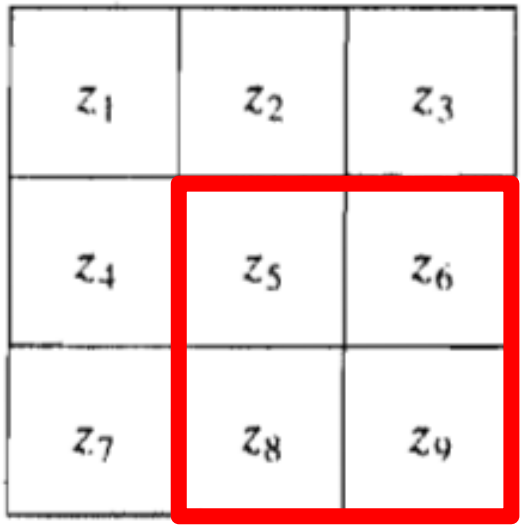
Edge detection

1.1. Roberts filter

Example:

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5)$$

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$



1.1. Filtre Roberts

Example:

15	20	100	120	130	130	135
25	20	25	120	120	140	140
20	15	20	15	130	135	135
20	20	15	20	150	130	135
15	35	30	20	100	110	110
25	30	25	30	100	110	120
20	25	30	20	25	110	120

$$\mathbf{GH} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{GV} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Edge detection

1.1. Roberts filter

Example:

15	20	100	120	130	130	135
25	20	25	120	120	140	140
20	15	20	15	130	135	135
20	20	15	20	150	130	135
15	35	30	20	100	110	110
25	30	25	30	100	110	120
20	25	30	20	25	110	120

GH

5	5	20	0	10	10	-135
-10	0	-10	10	15	-5	-140
0	0	0	135	0	0	-135
15	10	5	80	-40	-20	-135
15	-10	0	90	10	10	-110
0	0	-5	-5	0	10	-120
-20	-25	-30	-20	-25	-110	-120

Edge detection

1.1. Filtre Roberts

Example:

15	20	100	120	130	130	135
25	20	25	120	120	140	140
20	15	20	15	130	135	135
20	20	15	20	150	130	135
15	35	30	20	100	110	110
25	30	25	30	100	110	120
20	25	30	20	25	110	120

GV

5	-80	-95	-10	-10	5	140
0	-10	-100	-105	-10	-5	135
5	0	0	-110	15	-5	135
-5	20	10	-130	-30	-25	110
-10	0	5	-70	0	0	120
-10	0	0	-90	-85	-10	120
-25	-30	-20	-25	-110	-120	0

Edge detection

1.1. Filtre Roberts

GH

5	5	20	0	10	10	-135
-10	0	-10	10	15	-5	-140
0	0	0	135	0	0	-135
15	10	5	80	-40	-20	-135
15	-10	0	90	10	10	-110
0	0	-5	-5	0	10	-120
-20	-25	-30	-20	-25	-110	-120

GV

5	-80	-95	-10	-10	5	140
0	-10	-100	-105	-10	-5	135
5	0	0	-110	15	-5	135
-5	20	10	-130	-30	-25	110
-10	0	5	-70	0	0	120
-10	0	0	-90	-85	-10	120
-25	-30	-20	-25	-110	-120	0

$$G = \sqrt{GH^2 + GV^2}$$

7	80	97	10	14	11	194
10	10	100	105	18	7	194
5	0	0	174	15	5	191
16	22	11	153	50	32	174
18	10	5	114	10	10	163
10	0	5	90	85	14	170
32	39	36	32	113	163	120

Edge detection

1.1. Roberts filter

G

7	80	97	10	14	11	194
10	10	100	105	18	7	194
5	0	0	174	15	5	191
16	22	11	153	50	32	174
18	10	5	114	10	10	163
10	0	5	90	85	14	170
32	39	36	32	113	163	120

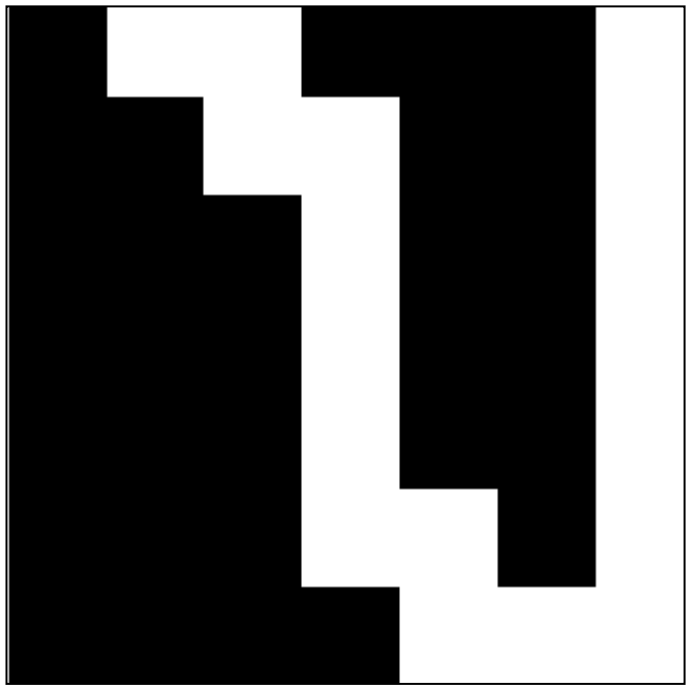
G after thresholding ***threshold=60 (average of G)***

0	1	1	0	0	0	1
0	0	1	1	0	0	1
0	0	0	1	0	0	1
0	0	0	1	0	0	1
0	0	0	1	0	0	1
0	0	0	1	1	0	1
0	0	0	0	1	1	1

Edge detection

1.1. Roberts filter

0	1	1	0	0	0	1
0	0	1	1	0	0	1
0	0	0	1	0	0	1
0	0	0	1	0	0	1
0	0	0	1	0	0	1
0	0	0	1	1	0	1
0	0	0	0	1	1	1



Edge detection

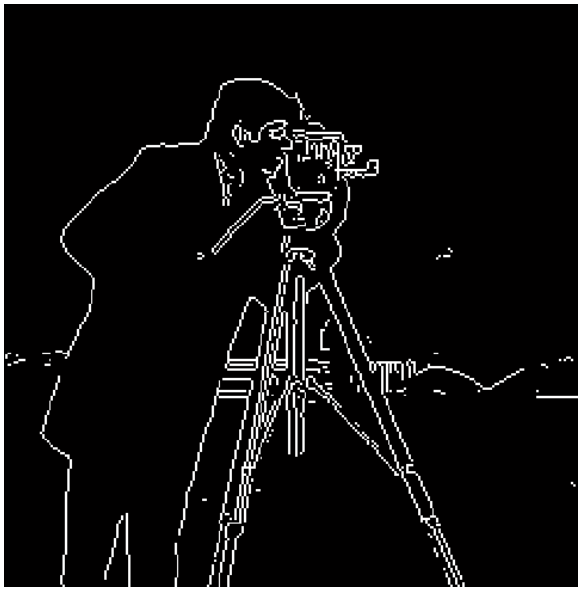
1.2. Prewitt filter

−1	−1	−1
0	0	0
1	1	1

H

−1	0	1
−1	0	1
−1	0	1

V



Edge detection

1.2. Prewitt filter

Example:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Edge detection

1.2. Prewitt filter

Example:

15	20	100	120	130	130	135
25	20	25	120	120	140	140
20	15	20	15	130	135	135
20	20	15	20	150	130	135
15	35	30	20	100	110	110
25	30	25	30	100	110	120
20	25	30	20	25	110	120

−1	−1	−1
0	0	0
1	1	1

H

−1	0	1
−1	0	1
−1	0	1

V

Edge detection

1.2. Prewitt filter

Example:

15	20	100	120	130	130	135
25	20	25	120	120	140	140
20	15	20	15	130	135	135
20	20	15	20	150	130	135
15	35	30	20	100	110	110
25	30	25	30	100	110	120
20	25	30	20	25	110	120

GH

45	70	165	265	380	400	280
0	-80	-190	-185	-100	5	5
-5	-15	-110	-80	-80	15	-15
15	25	35	-15	-50	-80	-50
15	25	30	-20	-50	-75	-35
-5	-5	-10	-75	-75	-65	10
-55	-80	-85	-165	-250	-340	-230

Edge detection

1.2. Prewitt filter

Example:

15	20	100	120	130	130	135
25	20	25	120	120	140	140
20	15	20	15	130	135	135
20	20	15	20	150	130	135
15	35	30	20	100	110	110
25	30	25	30	100	110	120
20	25	30	20	25	110	120

GV

40	85	200	125	30	25	-270
55	85	200	235	150	30	-405
55	-5	100	340	250	10	-405
70	10	-15	315	320	0	-375
85	10	-15	290	280	5	-350
90	25	-20	150	260	115	-330
55	10	-5	80	170	105	-220

Edge detection

1.2. Prewitt filter

GH

45	70	165	265	380	400	280
0	-80	-190	-185	-100	5	5
-5	-15	-110	-80	-80	15	-15
15	25	35	-15	-50	-80	-50
15	25	30	-20	-50	-75	-35
-5	-5	-10	-75	-75	-65	10
-55	-80	-85	-165	-250	-340	-230

GV

40	85	200	125	30	25	-270
55	85	200	235	150	30	-405
55	-5	100	340	250	10	-405
70	10	-15	315	320	0	-375
85	10	-15	290	280	5	-350
90	25	-20	150	260	115	-330
55	10	-5	80	170	105	-220

$$G = \sqrt{GH^2 + GV^2}$$

60	110	259	293	381	401	389
55	117	276	299	180	30	405
55	16	149	349	262	18	405
72	27	38	315	324	80	378
86	27	34	291	284	75	352
90	25	22	168	271	132	330
78	81	85	183	302	356	318

Edge detection

1.2. Prewitt filter

60	110	259	293	381	401	389
55	117	276	299	180	30	405
55	16	149	349	262	18	405
72	27	38	315	324	80	378
86	27	34	291	284	75	352
90	25	22	168	271	132	330
78	81	85	183	302	356	318

Normalization

60	110	255	255	255	255	255
55	117	255	255	180	30	255
55	16	149	255	255	18	255
72	27	38	255	255	80	255
86	27	34	255	255	75	255
90	25	22	168	255	132	255
78	81	85	183	255	255	255

Edge detection

1.2. Filtre Prewitt

60	110	255	255	255	255	255
55	117	255	255	180	30	255
55	16	149	255	255	18	255
72	27	38	255	255	80	255
86	27	34	255	255	75	255
90	25	22	168	255	132	255
78	81	85	183	255	255	255

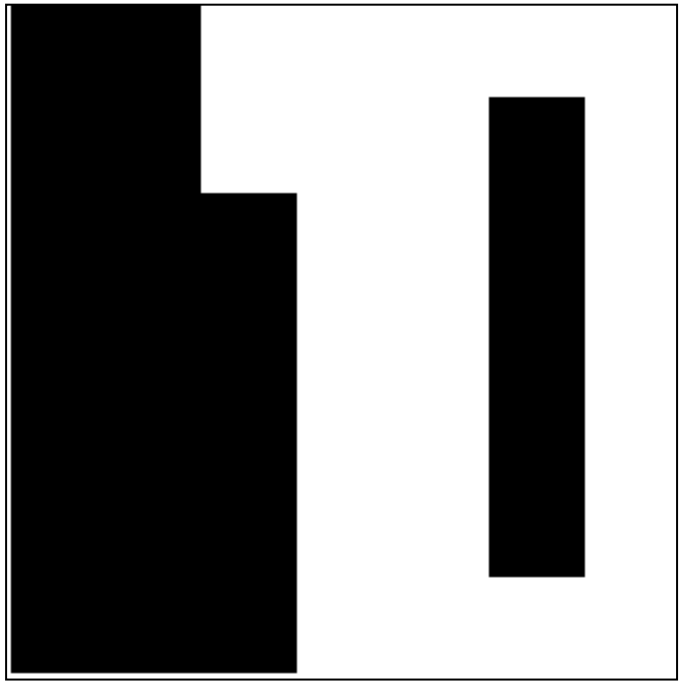
G after thresholding threshold=157 (average of G)

0	0	1	1	1	1	1
0	0	1	1	1	0	1
0	0	0	1	1	0	1
0	0	0	1	1	0	1
0	0	0	1	1	0	1
0	0	0	1	1	0	1
0	0	0	1	1	1	1

Edge detection

1.2. Filtre Prewitt

0	0	1	1	1	1	1
0	0	1	1	1	0	1
0	0	0	1	1	0	1
0	0	0	1	1	0	1
0	0	0	1	1	0	1
0	0	0	1	1	0	1
0	0	0	1	1	1	1



Edge detection

1.3. Sobel filter

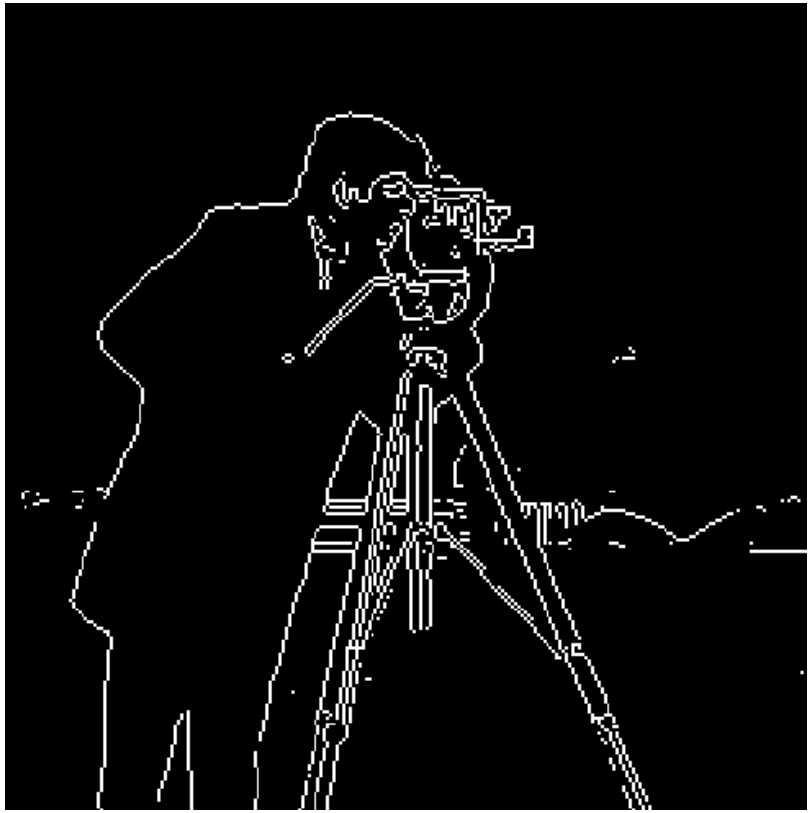
It uses the convolution product with the following masks:

−1	−2	−1
0	0	0
1	2	1

GH

−1	0	1
−2	0	2
−1	0	1

GV



1.3. Sobel filter

Example:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

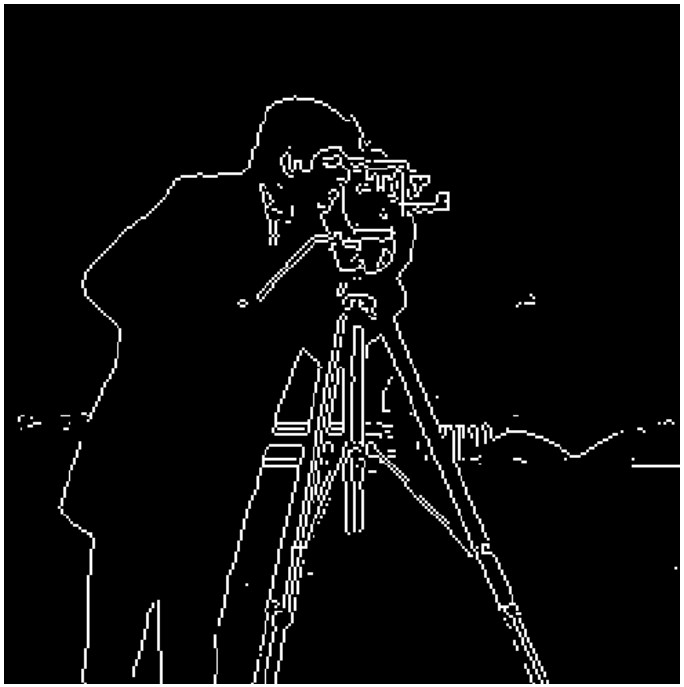
$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Edge detection

Notes

The sum of the coefficients for all masks is equal to zero→ the result will be 0 (black pixel) for homogeneous regions.

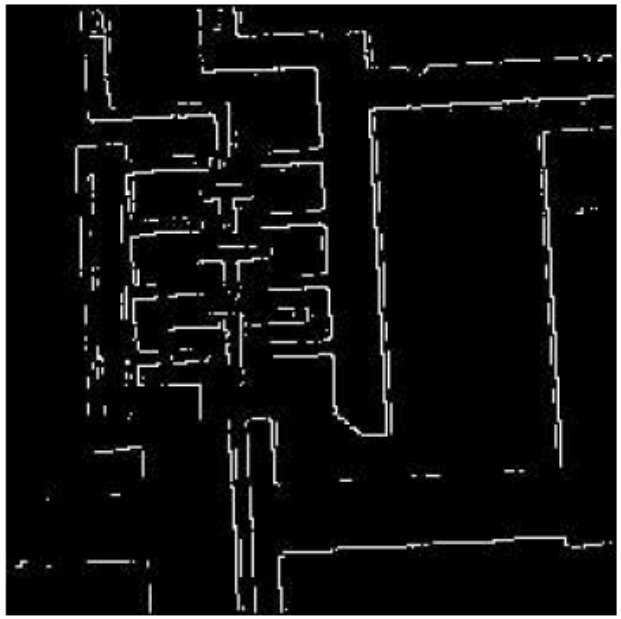


Edge detection

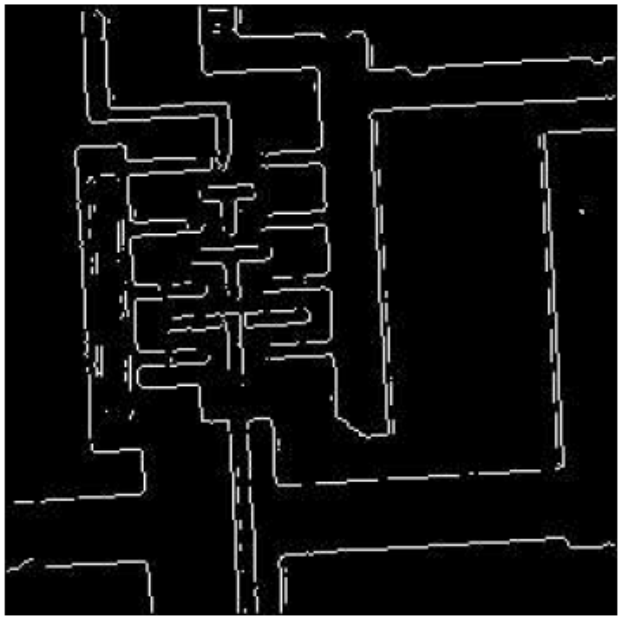
Notes

- Masks of size 2 × 2 are not symmetrical about the central pixel

- The Roberts filter is faster (kernel size) but very sensitive to noise



Roberts

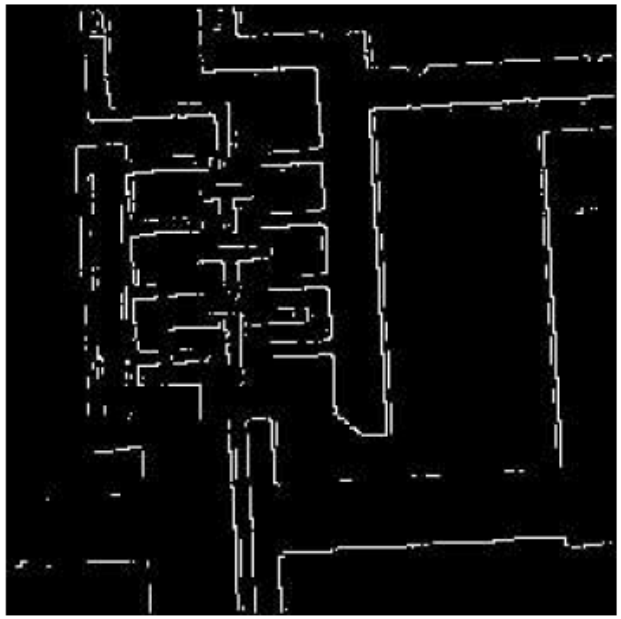


Sobel

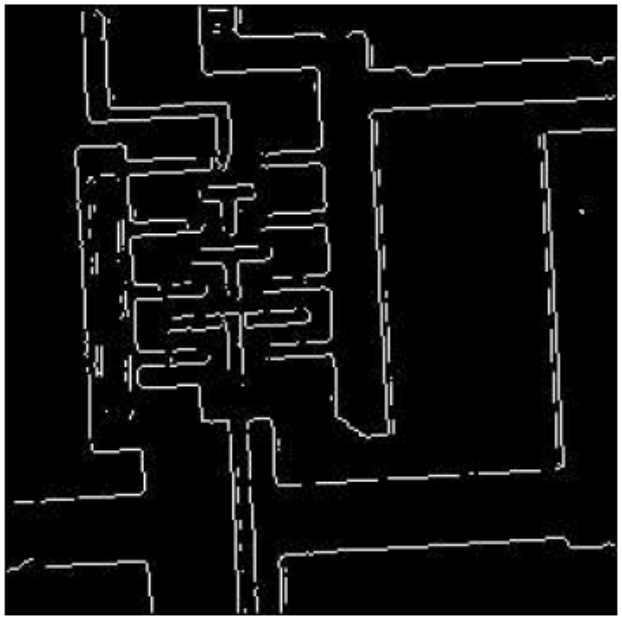
Edge detection

Notes

- The Sobel filter gives a better results than the Prewitt filter because the serie 1 2 1 is approximately a Gaussian.
-
- *“Of the three filters, the Sobel filters are probably the best, they provide good edges, and they perform reasonably well in the presence of noise”.*



Roberts

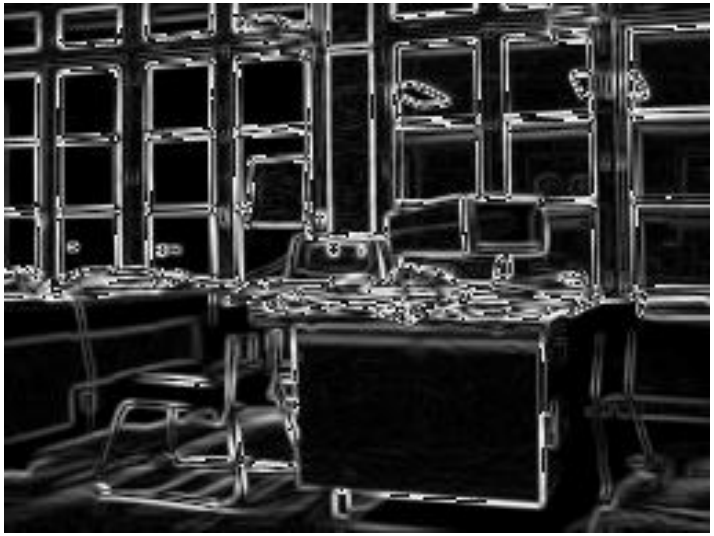


Sobel

Edge detection

Original image

Roberts filter



Prewitt filter

Sobel filter

Edge detection

Note

- There are others edge detection methods called optimal methods like **Canny, Deriche** …

Canny filter

Edge detection

2. Laplacian approch

The sum of second derivatives in both directions is called the laplacian, it is written as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

GH = $\dfrac{\partial^2 (x, y)}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$

GV = $\dfrac{\partial^2 (x, y)}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$

Edge detection

2. Laplacian approch

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\nabla^2(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

Laplacian mask

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Edge detection

2. Laplacian approch

- Several masks can be used

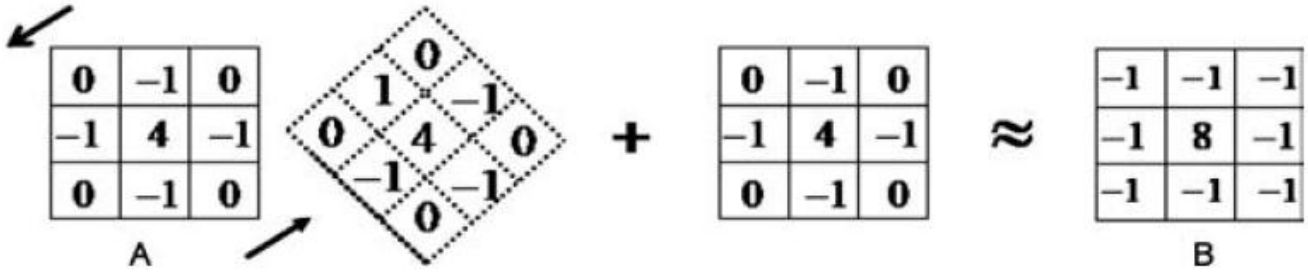
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

"Note that it is common to find implementations in which the signs of each coefficient in the convolution masks above are reversed."

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge detection

2. Laplacian approch



2. Laplacian approch

Octave

```
fspecial('laplacian',ALPHA)
```

$$\nabla^2 = \frac{4}{(\alpha+1)}\begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

Edge detection

2.Approche Laplacien

“The laplacian has the advantage over first derivative methods in that it is an isotropic filter, this means it is invariant under rotation.

That is, if the laplacian is applied to an image, and the image then rotated, the same result would be obtained if the image were rotated first, and the laplacian applied second. This would appear to make this class of filters ideal for edge detection.

However, a major problem with all second derivative filters is that they are very sensitive to noise”.

Edge detection

3. Laplacian of Gaussian

"However, a major problem with all second derivative filters is that they are very sensitive to noise".

- Problem : *sensitive to noise*
- Solution:
- Laplacian kernel is combined with the Gaussian kernel
- The filtering and derivation operators are done in a single calculation step (single pass)→ halving the calculations

Edge detection

3. Laplacian of Gaussian

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Log kernel 3x3 $\sigma = 0.5$

0.1733	0.4269	0.1733
0.4269	-3.1543	0.4269
0.1733	0.4269	0.1733

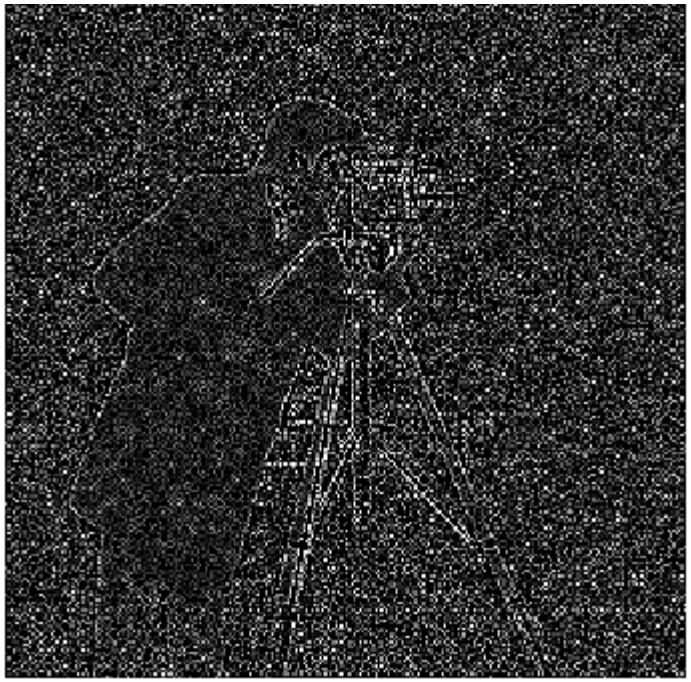
3. Laplacian of Gaussian

Noised image

Edge detection

3. Laplacian of Gaussian

Laplacian



Log