



Filters in the Spatial Domain

Lec-1 1

By

Lecturer Dr. Omar F. Mohammad

Fundamental of Spatial Filtering

- Filtering is a fundamental operation in image processing. It can be used for:

- 1 *image enhancement*

- 2 *Noise reduction*

- 3 *Edge detection,*

- 4 *Sharpening.*

- The **concept of filtering** has been applied in the:

- **Frequency domain**, where it rejects some frequency components while accepting others.

- **Spatial domain**, Spatial filtering modifies an image by replacing the value of each pixel with a function of the values of the pixel and its neighbors (filtering is pixel neighborhood operation).

- Commonly used spatial filtering techniques include **median, average, Gaussian filtering**, etc.

- The **filtering function** sometimes is called **filter mask**, or **filter kernel**.

- They can be broadly classified into two different categories :

1. **Linear filtering.**

2. **Nonlinear filtering (Order-statistic filters).**

Spatial Domain

- An image can be represented in the form of a **2D matrix** where each element of the matrix represents **pixel intensity**.
- This state of 2D matrices that depict the intensity distribution of an image is called **Spatial Domain**. It can be represented as shown below.

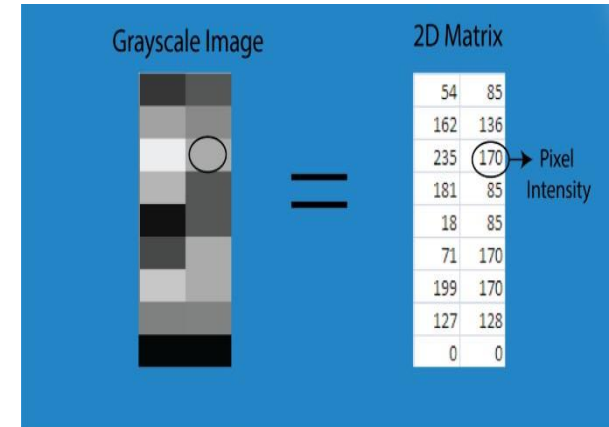
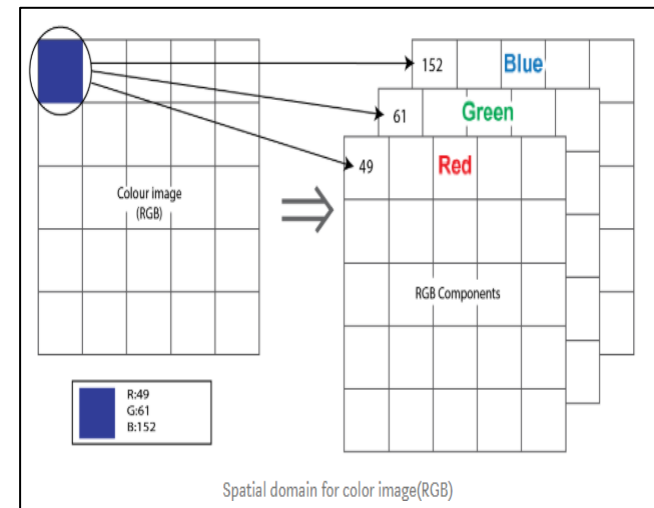


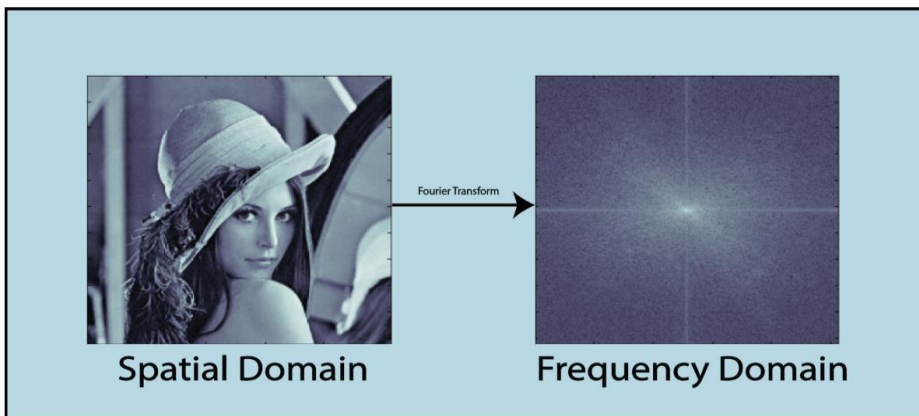
Illustration of Spatial Domain

- For the **RGB image**, the spatial domain is represented as a **3D vector of 2D matrices**. Each **2D matrix** contains the intensities for a **single color** as shown below.
- Each **pixel intensity** is represented as $I(x, y)$ where **x, y** is the **co-ordinate of the pixel** in the **2D matrix**. Different operations are carried out in this value.



Frequency Domain

- In frequency-domain methods are based on the **Fourier Transform of an image**.
- Roughly, the term frequency in an image **tells about the rate of change of pixel values**.
- The tie between spatial- and frequency-domain processing is the Fourier transform.
 - We use the **Fourier transform** to go from the spatial
 - To the frequency domain; to return to the spatial domain we use the **inverse Fourier transform**.
- The figure below depicts the conversion of the image from the **spatial domain** to the **frequency domain** using **Fourier Transformation**.



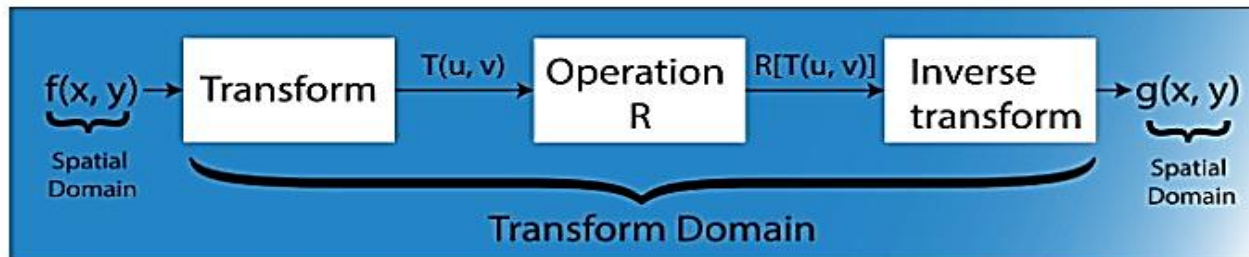
Spatial Domain

•Q: Why we need a domain other than spatial domain ?

•Answer

•Many times, image processing tasks are best performed in a domain other than the spatial domain. Moreover, **it is easy to detect some features in a particular domain, i.e., a new information can be obtained in other domains.**

Image Transformation mainly follows three steps-



Step-1. Transform the image.

Step-2. Carry the task(s) in the transformed domain.

Step-3. Apply inverse transform to return to the spatial domain.

Linear Filters

A linear spatial filter performs a sum-of-products operation between an image **f** and a filter kernel, **w**.

- The kernel is an array whose **size** defines the **neighborhood of operation**, and whose **coefficients** determine the **nature of the filter**.
- Other terms used to refer to a spatial filter kernel are mask, template, and window.
- We use the term filter kernel or simply kernel.

The equation below illustrates the mechanics of linear spatial filtering using a kernel. At any point (x, y) in the image, the response, g(x, y), of the filter, is the sum of products of the kernel coefficients and the image pixels encompassed by the kernel:

$$g(x, y) = w(-1, -1) f(x-1, y-1) + w(-1, 0) f(x-1, y) + \\ \dots + w(0, 0) f(x, y) + \dots + w(1, 1) f(x+1, y+1)$$

As coordinates x and y are varied, the center of the kernel moves from pixel to pixel, generating the filtered image, g, in the process

Linear Filters

Observe that the center coefficient of the kernel, $w(0, 0)$, aligns with the pixel at location (x, y) .

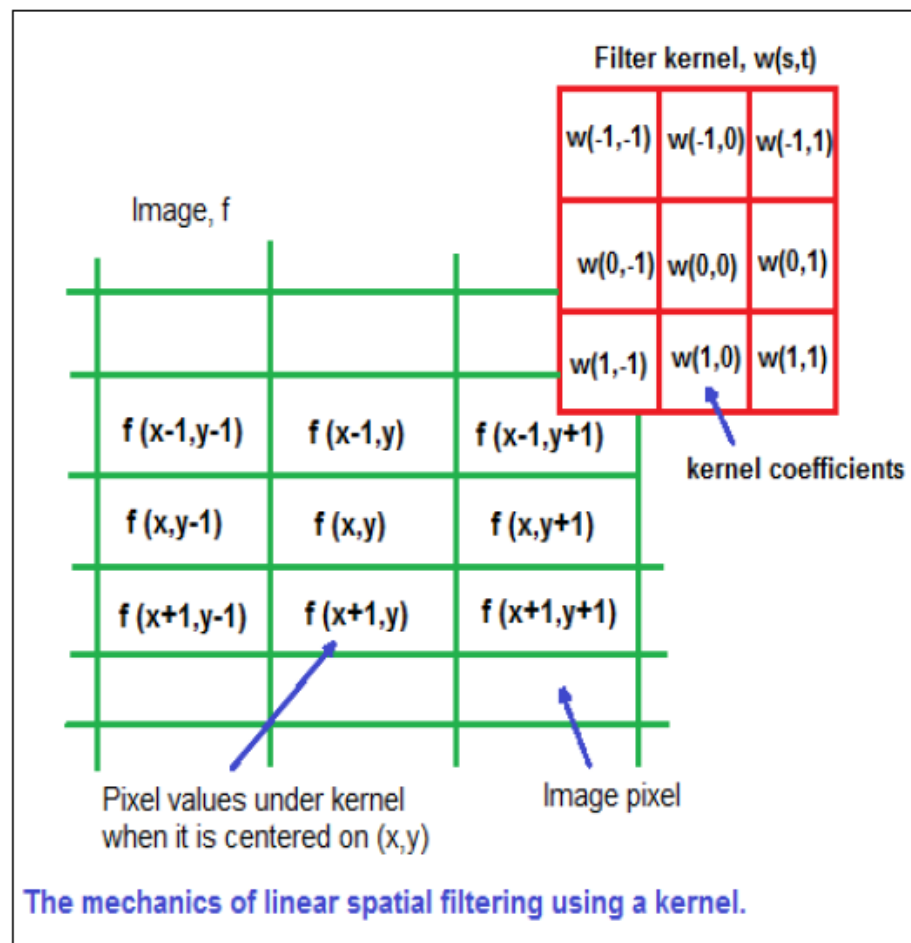
➤ For a kernel of size $m \times n$, we assume that $m=2a+1$ and $n=2b+1$, where a and b are nonnegative integers.

This means that our focus is on kernels of odd size in both coordinate directions.

➤ In general, linear spatial filtering of an image of size $M \times N$ with a kernel of size $m \times n$ is given by the expression

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)$$

where x and y are varied so that the center (origin) of the kernel visits every pixel in f once.



Smoothing Filters

Smoothing (also called averaging) spatial filters are **used to reduce sharp transitions in intensity**.

Because random noise typically consists of sharp transitions in intensity, an obvious application of smoothing is noise reduction.

Smoothing prior to image resampling to reduce aliasing, is also a common application.

- The **commonly used smoothing filters** are **Averaging** and **Median filters**.
- It can be **performed using the convolution operation**.

$$s(x, y) = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} h(m, n) f(x - m, y - n)$$

Where: $h(m, n)$ is a filtering mask of size $M \times N$.

- **Each element in this filter mask** usually represents the **weights** used in the linear combination.

Average Filter

The average filtering is also called **mean filtering**. Where the output pixel value is the mean of its neighborhood. There are two types (Standard Average and Weighted Average) . Thus, the filtering mask is as follows (3*3 as an example).

Desirable effect: the most application of smoothing is **noise reduction**, because random noise typically consists of sharp transitions in gray levels,

Undesirable effect: the undesirable side effect is **blur edges**. edges (which almost always are desirable features of an image) also are characterized by sharp transitions in gray levels.

Standard average

 $\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1



A box filter

weighted average.

 $\frac{1}{16} \times$

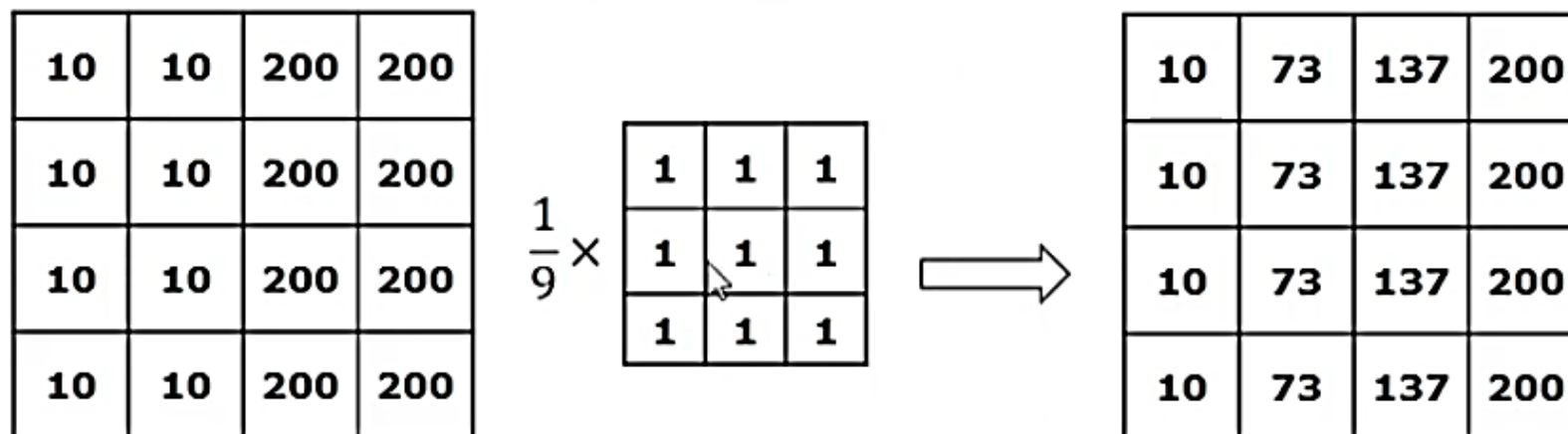
1	2	1
2	4	2
1	2	1



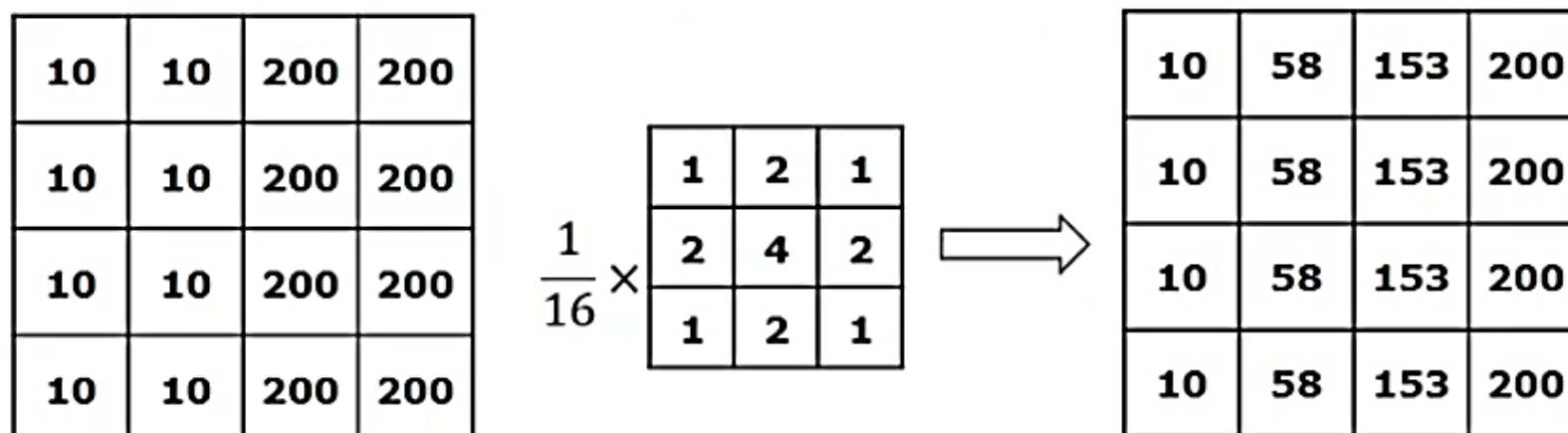
The basic strategy is

Average Filter

The following image represent the gray values of an edge. The following example will explain the effect of standard average on it:

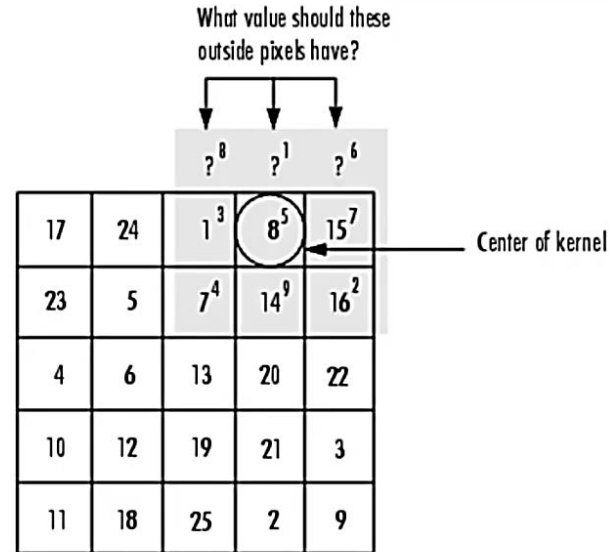


After applying weighted average:



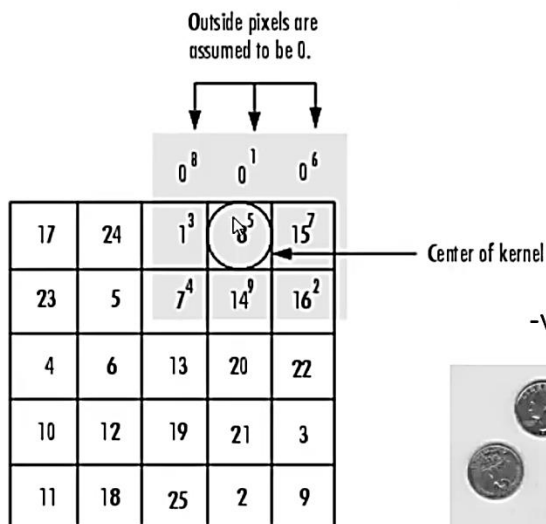
As we note applying weighted average reduce the amount of blurring on the edges

What happens when the Values of the Kernel Fall Outside the Image??!

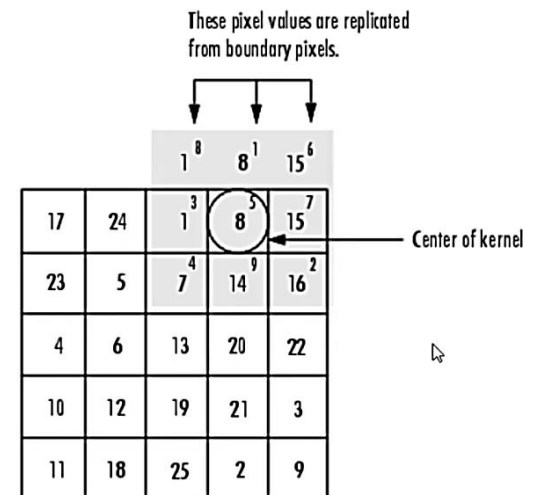


First solution :Zero padding,

border padding



-ve: black border



Example on the standard average filter with zero padding

1/9	1/9	1/9			
1/9	13	12	10	11	12
1/9	1/9	1/9			
1/9	10	11	12	10	11
	11	13	200	15	14
	10	12	13	13	14
	10	11	12	13	11

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

 \equiv

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

5				

$13 \cdot \frac{1}{9} + 12 \cdot \frac{1}{9} + 10 \cdot \frac{1}{9} + 11 \cdot \frac{1}{9} = 5$
OR
 $(13 \cdot 1 + 12 \cdot 1 + 10 \cdot 1 + 11 \cdot 1) / 9 = 5$

	1/9	1/9	1/9		
13	12	10	11	12	
1/9	1/9	1/9			
10	11	12	10	11	
1/9	1/9	1/9			
11	13	200	15	14	
10	12	13	13	14	
10	11	12	13	11	

5	8			


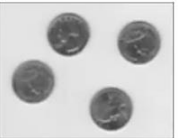
$13 \cdot \frac{1}{9} + 12 \cdot \frac{1}{9} + 10 \cdot \frac{1}{9} + 10 \cdot \frac{1}{9} + 11 \cdot \frac{1}{9} + 12 \cdot \frac{1}{9} = 8$
OR
 $(13 \cdot 1 + 12 \cdot 1 + 10 \cdot 1 + 10 \cdot 1 + 11 \cdot 1 + 12 \cdot 1) / 9 = 8$

You have to apply same process till to the end of the image

13	12	10	11	12
10	11	12	10	11
11	13	200	15	14
10	12	13	14	
10	11	12	13	11

5	8	7	7	5
8	32	33	33	8
7	32	33	34	9
7	32	34	34	9
5	8	8	8	6

$(13 \cdot 1 + 14 \cdot 1 + 13 \cdot 1 + 11 \cdot 1) / 9 = 6$

As you can see, we successfully remove the noise (200) and the gray values are very close to each other except the pixels on boundary it have low gray value because of zero padding and this will produce black border and we can use replicate border to eliminate black border

13	12	10	11	12	
10	11	12	10	11	
11	13	200	15	14	
10	12	13	13	14	
10	11	12	13	11	

5	8	7	7	5
8	32	33	33	8

$(11 \cdot 1 + 12 \cdot 1 + 10 \cdot 1 + 11 \cdot 1 + 15 \cdot 1 + 14 \cdot 1) / 9 = 8$

standard average filter with replicate border

13	12	10	11	12
10	11	12	10	11
11	13	200	15	14
10	12	13	13	14
10	11	12	13	11



12	11	11	11	11
12	32	33	33	12
11	32	33	34	13
11	32	34	34	13
10	11	12	12	12



As you can see, the gray values of boundary's pixel are not affected too much with averaging filter

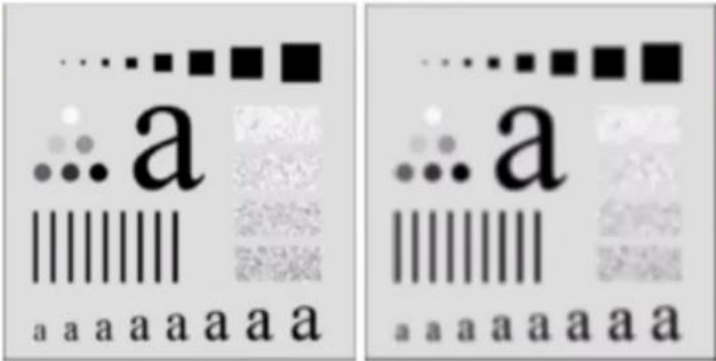
Averaging effects: blurring + reducing noise

Original image



3 x 3 averaging

5 x 5 averaging



9 x 9 averaging

5 x 15 averaging



35 x 35 averaging

Non-Linear Filters

The order-statistical filters are usually **non-linear filters**, which are hardly represented by convolution.

- The value of a given pixel in the output image is represented by some statistic within its support neighborhood in the original image, such as the median filter.

There are some other filters as well such as the **max/min filter**.

- The **median filter** simply replaces the value of the pixel with the median value within its neighborhood.
- The **max/min filter** replaces the value of the pixel with the maximum/minimum value within its neighborhood.

Those filters are normally **non-linear** and **cannot be easily implemented in the frequency domain**.

- However, the common elements of a filter are:
 - (1) A neighborhood
 - (2) An operation on the neighborhood include the pixel itself.

Median Filters

The **median filter** is a **non-linear filter** (order filter). These filters are based on a specific type of image statistics called order statistics.

- Typically, these filters operate on small sub image, “Window”, and replace the center pixel value (similar to the convolution process).
- **Order statistics** is a **technique that arranges the entire pixel in sequential order**, given an $N \times N$ window (W) the pixel values can be ordered from smallest to the largest.

$I_1 \leq I_2 \leq I_3 \dots \dots \dots < I_N$

Where: $I_1, I_2, I_3 \dots \dots \dots, I_N$

are the intensity values of the subset of pixels in the image.



Median Filters

110	120	90	130
91	94	98	200
90	95	99	100
82	96	85	90

becomes

95

the **median filter** is based on ordering (**ranking**) the pixels , then *replacing the value of a pixel by the median of the gray levels in the neighborhood of that pixel.*

Median filters are quite **popular** because, for certain types of random noise, they provide **excellent noise reduction + less blurring** than linear smoothing filters of similar size.

Median filters are particularly effective in the presence of *impulse noise, also called **salt-and-pepper noise** because of its appearance as white and black dots superimposed on an image.*

Steps:

1. Sort the pixels in ascending order:

90,90, 91, 94, 95, 98, 99, 110, 120

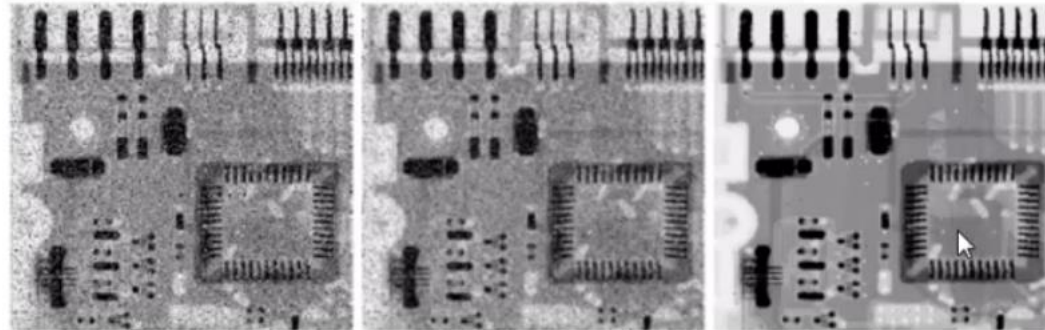
2. replace the original pixel value by the median :

95



Median Filters

Which one has removed the salt-and-pepper noise??



The original image with salt and pepper noise

The smoothed image using averaging

The smoothed image using median

a
th

Salt noise

150	151	155	150	151
152	255	153	150	152
153	154	155	0	153
157	158	159	157	155

Pepper noise

150	151	155	150	151
152	255	153	150	152
153	154	155	0	153
157	158	159	157	155

After applying
3×3 Median filter

151	152	151	151	151
152	153	153	152	151
154	155	155	153	153
157	157	157	155	155

- ❑ As we note, the salt and pepper noise is removed completely from this image without blurring effect

Median Filters

Example 1:

Using median filters to remove salt and pepper noise, from sub image $I(r, c)$.

First pixel:

1. **Rearrange** the 3x3 neighbor pixels in ascending way such as:

2, 5, 16, 19, **20**, 21, 25, 43, 51

2. By **using median filter** we select the **median value 20** and put it in buffer image in the **position (1,1)**

Second pixel:

1. **Rearrange** the 3x3 neighbor pixels in ascending way such as:

2, 15, 16, 18, **19**, 20, 23, 25, 51

2. By **using median filter** we select the **median value 19** and put it in buffer image in the **position (1,1)**

5	16	20	15	17	20
21	2	25	23	34	19
43	51	19	18	40	42
15	18	25	23	38	40
52	44	34	12	10	13

5	16	20	15	17	20
21	20	?	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

5	16	20	15	17	20
21	20	19	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

Min Filters

Example 2:

Using min. filters to remove salt and pepper noise, from sub image $I(r, c)$.

First pixel:

1. Rearrange the 3x3 neighbor pixels in ascending way such as:

5, 13, 16, 19, 20, 21, 25, 25, 43, 51

2. By using min. filter we select the **minimum value 5** and put it in buffer image in the **position (1,1)**

Second pixel:

1. Rearrange the 3x3 neighbor pixels in ascending way such as:

13, 15, 16, 19, 20, 22, 23, 25, 51

2. By using min. filter we select the **minimum value 13** and put it in buffer image in the **position (1,1)**

5	16	20	15	17	20
21	13	25	23	34	19
43	51	19	22	40	42
15	18	25	23	38	40
52	44	34	12	10	13

5	16	20	15	17	20
21	5	?	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

5	16	20	15	17	20
21	5	13	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

Max Filters

Example 3:

Using max. filters to remove salt and pepper noise, from sub image $I(r, c)$.

First pixel:

1. Rearrange the 3x3 neighbor pixels in ascending way such as:

5,13,16,19,20, 21,25,25,43,**51**

2. By using max. filter we select the **maximum value 51** and put it in buffer image in the **position (1,1)**

Second pixel:

1. Rearrange the 3x3 neighbor pixels in ascending way such as:

13,15,16,19,20,22,23,25,**51**

2. By using max. filter we select the **maximum value 51** and put it in buffer image in the **position (1,1)**

5	16	20	15	17	20
21	13	25	23	34	19
43	51	19	22	40	42
15	18	25	23	38	40
52	44	34	12	10	13

5	16	20	15	17	20
21	51	?	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

5	16	20	15	17	20
21	51	51	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

Sharpening Filters

The **objective of sharpening** is to **draw attention** to the **fine details of an image**. This is also **related to the situation** where an image that has been blurred and now needs to be de-blurred.

In contrast to the process of image :

Smoothing that normally uses **pixel averaging techniques**,
Sharpening can be conducted using **spatial differentiation**.

The **image differentiation** actually:

- Enhances edges
- Discontinuities
- Depresses the areas of slowly changing gray-level values.

The Sharpening filter indicates the filter should have **positive coefficients near its center** and **negative coefficients in the outer periphery**.

Laplacian Filters

These filters will **tend to bring out**, or **enhance details** in the image. Example of convolution masks for the Laplacian-type filters are:

0	-1	0
-1	4	-1
0	-1	0

0	1	0
1	-4	1
0	1	0

Rotating by 90

-1	-1	-1
-1	8	-1
-1	-1	-1

1	1	1
1	-8	1
1	1	1

Rotating by 45

The **sum of the coefficients** in this kernel is **zero**, this mean that:

- when the kernel is over an **area of constant** (background area) or **slowly varying gray level**, the **result of convolution** is **zero** or some **very small number**.
- when **gray level is varying rapidly within the neighborhood**, the **result of the convolution** can be **large number**.

This number can be positive or negative, because the kernel contains both positive and negative coefficients; we therefore need to choose an output image representation that supports negative number.

Laplacian Filters

The **derivatives** of a digital function are defined in terms of **differences**

Definitions of the **first and 2nd-order** derivatives of a 1-D function $f(x)$ are the differences:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x).$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x).$$

f	50	50	60	90	100	100	100
-----	----	----	----	----	-----	-----	-----

$\frac{\partial f}{\partial x}$	0	10	30	10	0	0
---------------------------------	---	----	----	----	---	---

$\frac{\partial^2 f}{\partial^2 x}$		10	20	-20	-10	0
-------------------------------------	--	----	----	-----	-----	---

Laplacian Filters

- ▣ The Laplacian, for a function (image) $f(x, y)$ of two variables, is defined and given below:

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

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

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

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

2nd Derivative - Laplacian filter

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

20	20	20	20
20	10	10	10
20	10	10	10
20	10	10	10

Ex: apply the second derivative on the shaded pixel

$$\nabla^2 f = 20 + 20 + 10 + 10 - 4 * 10$$
$$\nabla^2 f = 20$$

$$\nabla^2 f = 10 + 10 + 10 + 10 - 4 * 10$$
$$\nabla^2 f = 0$$

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

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

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

$$\left. \begin{array}{l} \frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \\ \frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y) \end{array} \right\} \rightarrow h_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Composite Laplacian filter

Composite Laplacian filter :

$g(x, y) = f(x, y) + \nabla^2 f(x, y)$ if the center coefficient of the Laplacian mask is positive.

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

$$g(x, y) = f(x, y) + \nabla^2 f(x, y)$$

$$g(x, y) = f(x, y) + 4f(x, y) - f(x+1, y) - f(x-1, y) - f(x, y+1) - f(x, y-1)$$

$$g(x, y) = 5f(x, y) - f(x+1, y) - f(x-1, y) - f(x, y+1) - f(x, y-1)$$

0	-1	0
-1	5	-1
0	-1	0

Composite Laplacian filter

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

f

50	60	90	100
50	60	90	100
50	60	90	100
50	60	90	100

Laplacian

0	-1	0
-1	4	-1
0	-1	0

$\nabla^2 f$

-10	-20	20	10
-10	-20	20	10
-10	-20	20	10
-10	-20	20	10

$g = f + \nabla^2 f$

40	40	110	110
40	40	110	110
40	40	110	110
40	40	110	110

**Composite
Laplacian**

50	60	90	100
50	60	90	100
50	60	90	100
50	60	90	100

0	-1	0
-1	5	-1
0	-1	0

40	40	110	110
40	40	110	110
40	40	110	110
40	40	110	110

Composite Laplacian filter

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

0	-1	0		
-1	50	60	90	100
0	50	60	90	100
	50	60	90	100
	50	60	90	100

$\nabla^2 f$

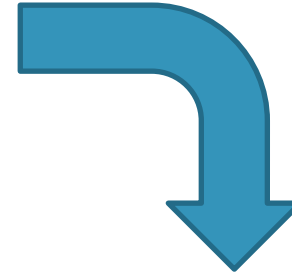
-10			

$g = f + \nabla^2 f$

40			

$$\begin{aligned}\nabla^2 f &= 4 * 50 - 50 - 50 - 60 - 50 \\ &= 200 - 210 \\ &= -10\end{aligned}$$

$$\begin{aligned}g(x, y) &= 50 - 10 \\ g(x, y) &= 40\end{aligned}$$



	0	-1	0
50	60	90	100
50	60	90	100
50	60	90	100
50	60	90	100

$\nabla^2 f$

-10	-20	20	

$g = f + \nabla^2 f$

40	40	110	

$$\begin{aligned}\nabla^2 f &= 4 * 90 - 90 - 90 - 60 - 100 \\ &= 360 - 340 \\ &= 20\end{aligned}$$

$$\begin{aligned}g(x, y) &= 90 + 20 \\ g(x, y) &= 110\end{aligned}$$

Composite Laplacian filter

50	60	90	100	
50	60	90	100	
50	60	90	100	
50	60	90	100	
		0	-1	0
		-1	4	-1
		0	-1	0

$\nabla^2 f$

-10	-20	20	10
-10	-20	20	10
-10	-20	20	10
-10	-20	20	10


$g = f + \nabla^2 f$

40	40	110	110
40	40	110	110
40	40	110	110
40	40	110	110

$$\begin{aligned}
 \nabla^2 f &= 4 * 100 - 100 - 100 - 100 - 90 \\
 &= 400 - 390 \\
 &= 10
 \end{aligned}$$

$$g(x, y) = 100 + 10$$

$$g(x, y) = 110$$



End of Lecture