

Image Processing 2022-2023

Lab 4: Neighborhood Operations

Agenda

- 1- Idea of neighborhood operations (Convolution)...!!
- 2- Linear Filter Convolution Function!
- 3- Smoothing
 - a. Usage (demo1)
 - b. Main idea
 - c. Mean filter vs. Gaussian filter (demo2)
 Generation of Gaussian Filter
 - d. 1D vs. 2D (demo3)
 - e. Mean filter with different mask size (demo4)
- 4- Sharpening
 - a. Usage
 - b. Main idea
 - c. Point sharpening (Laplacian) (demo5)
- 5- Edge Detection
 - a. Usage
 - b. Main idea
 - c. Sobel filters (Line detection) (demo6)
- 6- Practical

Quiz 1:

Quiz 1 will be available on Friday 10/11/2023 from 2:00 PM to 2:15 PM

https://docs.google.com/forms/d/e/1FAIpQLSdRk0rTQdfpSZJOT8sU5UFFs_p17WS3n4Z6QLpXvkPr8C22YA/viewform

For Demos

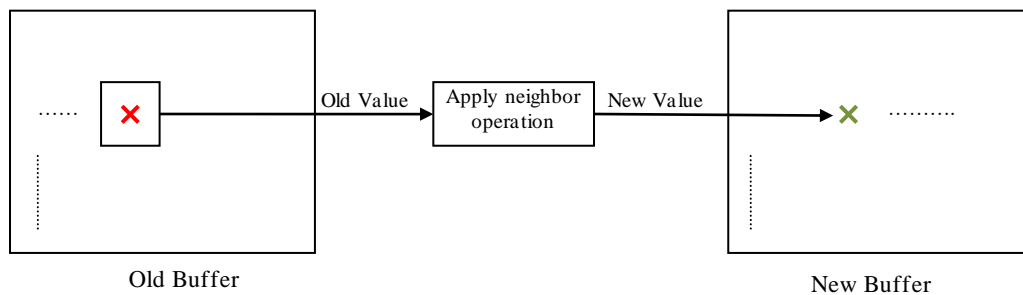
DEMO	Processing	Image Name	Values	Package
Demo1	Blurring the image to remove small details and extract large objects	Space	1) Filters→blur→ Mean 15×15 2) Image→Adjust. →black & white	\2008-2009\HDStudio
	Blurring the image to bridge gaps in text	Gappedtext	1) Filters→blur→ Mean 3×3 2) Image→Adjust. →black & white	
Demo2	Mean vs. Gaussian Discuss the blurring effects of both	Joker	Window = 15, 25	BLURRING Filters
Demo3	Mean Filter: 1D vs. 2D Discuss the time	Joker	Window = 25	BLURRING Filters
Demo4	Blurring the image with various mask size... discuss the effects on “a” letters and noisy dots!!	Joker	Mean filter by 5×5, 15×15, 25×25 and 29×29	BLURRING Filters
	3.22) why 25×25 mask is the only one that merges the vertical lines into a block?!			
Demo5	Sharpening with Laplacian mask... discuss the postprocessing!!	Moon	Conv. matrix→Load “LaplacianSharp.conv”→cutoff vs. normalization	\2006-2007\Mohammed AbdulFattah
Demo6	Edge detection with Sobel masks	Joker	Filter→Sobel→Vertical/Horiz ontal/Diagonal	\2006-2007\Mohammed AbdulFattah
	On the result of each: Discuss why there are black, white and grey regions!			

Details

1- Idea of neighborhood operations...!!

Show that it's applied by moving the mask over the original buffer in convolution manner... and then the **result should be placed in new buffer**. Following are the steps:

- place the mask at the beginning of the row
- apply operation between the mask and the overlapped region from the image
- while end of the row is **not** reached, move the mask 1 pixel right and repeat from (b)
- else, move the mask one pixel down and repeat from (a), until no more remaining rows



Problem of the Convolution

Border pixels are not reached by the mask. This will result an image without border pixels.

Possible solutions:

- Padding the original image before convolution, by **adding 0's** rows and columns.
- Padding the original image before convolution, by **replicating** rows and columns.

Complexity of the Convolution

$O(N^2 \times K^2)$; where N^2 is the image size and K^2 is the mask size

2- Linear Filter Convolution Function!

All tasks of this lab are **based on single function** that performs the convolution process using a given linear filter. So, implementing this function first will help doing all remaining tasks in much simple and fast way.

Function Signature:

```
Out = LinearFilter(Inp, Filter, FWidth, FHeight, Postprocess)
```

Parameters:

1. `Inp`: 2D buffer of the input image
2. `Out`: 2D buffer of the resulting image
3. `Filter`: 2D buffer that contains the values of the mask to be convolved
4. `FWidth`, `FHeight`: width & height of the given filter (odd values: 1, 3, 5 ... etc.)
5. `Postprocess`: enumeration that specifies the required postprocessing method to be applied after the convolution. It can be one of the following:
 - a. No postprocessing
 - b. Normalization
 - c. Cut-off
 - d. Absolute

3- Smoothing

a. Usage

(demo1)

- 1- Blurring to remove small details and extract large objects.
- 2- Blurring to bridge small gaps in lines, curves or text.
- 3- Noise reduction.

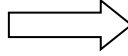
b. Main idea

- Sum of all mask coefficients equal **one** and all coefficients are +ve's...
- Explain the effect of such mask on...
 - i. background region → leave it
 - ii. edge region → blur it

Ex.: 3×3 Mean filter on 3×3 image

10	12	15
12	10	10
8	8	10

BG region

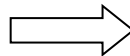


...
...	11	...
...

New value same as the BG

0	0	100	100	100
0	0	100	100	100
0	0	100	100	100

Edge region



...
...	33	66	100	...
...

Edge is blurred

- Post-processing: NONE

c. Mean filter vs. Gaussian filter

(demo2)

Both lead to smoothing, but the Gaussian filter is **less blurring** than the mean filter (i.e. less side effect on the edges).

$1/9 \times$

1	1	1
1	1	1
1	1	1

Mean 3×3

$1/15 \times$

1	2	1
2	3	2
1	2	1

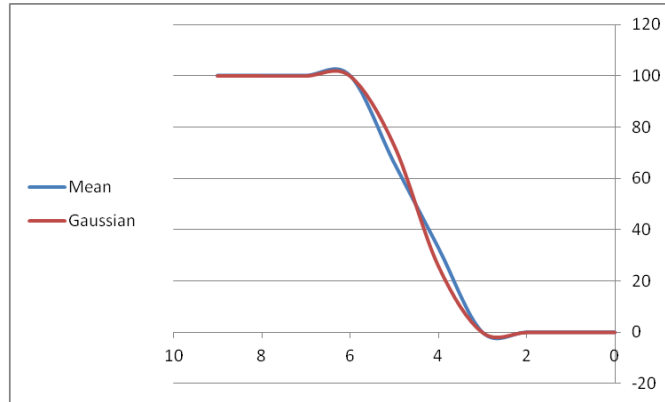
Gaussian 3×3

Ex.: 3×3 Mean filter on 3×5 image

0	0	100	100	100
0	0	100	100	100	...	33	66	100	...
0	0	100	100	100

Ex.: 3×3 Gaussian filter on 3×5 image

0	0	100	100	100
0	0	100	100	100	...	26	73	100	...
0	0	100	100	100



Mean Filter:

1. **User input:** width & height of the filter (not necessarily square)
2. Construct the mean filter with the desired width & height
3. Fill all filter values by: $1 / (\text{width} \times \text{height})$
4. Convolute the Image with constructed filter using the **LinearFilter** function (no postprocessing)

Gaussian Filter:

1. **User input:** Gaussian sigma (σ)
2. **Construct the Gaussian Filter with the given σ**
 - a. **Compute Mask Size** that preserve most of Gaussian area under the curve

$$N = \text{integer} [3.7 \times \sigma - 0.5]$$

$$\text{MaskSize} = 2 \times N + 1$$

- b. **Fill the Mask**

$$\text{GaussMask}(x, y) = [1 / (2\pi \times \sigma^2)] e^{-(x^2 + y^2) / (2 \times \sigma^2)}$$

$$x: \quad - \text{MaskSize} / 2 \rightarrow + \text{MaskSize} / 2$$

$$y: \quad - \text{MaskSize} / 2 \rightarrow + \text{MaskSize} / 2$$

3. Convolute the Image with constructed filter using the **LinearFilter** function (no postprocessing)

d. 1D vs. 2D

(demo3)

- Sometimes, there are some 2D filters that can be processed by applying 1D version of it twice over the image (e.g. Mean filter)
- The complexity is reduced to $O(N^2 \times K)$ instead of $O(N^2 \times K^2)$

e. Mean filter with different mask size

(demo4)

All coefficients = 1 / (mask size)

[BONUS] SELF READING: 1D Filters

1- 1D Mean filter:

the **FAST** implementation of the mean filter is done by using 1D filter and apply it twice on the image (first: apply it on the rows... then apply it on the columns of the resulting image).

The complexity becomes $O(N^2 \times K)$ instead of $O(N^2 \times K^2)$

Ex.: the same results of using a 3*3 mean filter can be obtained by a pass of the 1D mask [1 1 1] through an image. The result of this pass is then followed by a pass of the mask:

1
1
1

The final result is then scaled by 1/9

2- 1D Gaussian filter:

1. **Compute the 1D Gaussian Filter with σ** a. **Compute Mask Size**

$$N = \text{integer} [3.7 \times \sigma - 0.5]$$

$$\text{MaskSize} = 2 \times N + 1$$

b. **Fill the Mask**

$$1D\text{-GaussMask}(x) = [1/(\sqrt{2\pi} \times \sigma)] e^{-x^2 / (2\sigma^2)}$$

$$x: \quad -\text{MaskSize} / 2 \rightarrow +\text{MaskSize} / 2$$

2. **Convolute the Image in X-Direction**

$$X\text{-BlurredImg} = \text{OrigImg CONVOLVED BY } 1D\text{-GaussMask}$$

3. **Convolute the Result in Y-Direction**

$$\text{BlurredImg} = X\text{-BlurredImg CONVOLVED BY } 1D\text{-GaussMask}$$

4- Sharpening

a. Usage


- 1- Highlight fine details
- 2- Enhance detail that has been blurred during image acquisition

b. Main idea

- Sum of mask coefficients equal **one** and coefficients alternating between +ve & -ve
- Explain the effect of such mask on...
 - i. background region → leave it
 - ii. edge region → sharp it

Ex.: 3×3 Laplacian filter on 3×3 image

X ₁	X ₂	X ₃	0	-1	0
X ₄	Y	X ₅	-1	5	-1
X ₆	X ₇	X ₈	0	-1	0



	Y X ₁	Y X ₂	Y X ₃	Y X ₄	Y	Result	Explanation
If region is BG	≈ 0				Y	≈ Y	Leave BG region as it's
If Y > its neighbor	≈ large +ve value				Y	≈ Y + value	Increase the difference between Y and its neighbors (i.e. sharp it)
If Y < its neighbor	≈ large -ve value				Y	≈ Y - value	Increase the difference between Y and its neighbors (i.e. sharp it)

- Post-processing: CUT OFF

c. Point sharpening (Laplacian)

(demo5)

-1	-1	-1
-1	9	-1
-1	-1	-1

5- Edge Detection

1. Usage

- Preprocessing step for automated inspection
- Highlight small specs that may not readily visible in the image (e.g. imperfections in the lens)
- Segmentation of objects

2. Main idea

- Sum of mask coefficients equal **zero** and coefficients alternating between +ve & -ve
- Explain the effect of such mask on...
 - i. background region → eliminate it
 - ii. edge region → show it

Ex.: 3×3 Laplacian filter on 3×3 image

X ₁	X ₂	X ₃	0	-1	0
X ₄	Y	X ₅	-1	4	-1
X ₆	X ₇	X ₈	0	-1	0



	Y X ₁	Y X ₂	Y X ₃	Y X ₄	Result	Explanation
If region is BG	≈ 0				≈ 0	Eliminate the BG region
If Y > its neighbor	≈ large +ve value				≈ +ve value	Show the edge (by large +ve value)
If Y < its neighbor	≈ large -ve value				≈ -ve value	Show the edge (by large -ve value)

- Post-processing: ABSOLUTE

3. Sobel filters (line detection)

(demo7)

1. Horizontal line detection

Convolve the image with the following mask using **LinearFilter** function (with ABSOLUTE postprocessing)

-1	-2	-1
0	0	0
1	2	1
Horizontal		

2. Vertical line detection

Convolve the image with the following mask using **LinearFilter** function (with ABSOLUTE postprocessing)

-1	0	1
-2	0	2
-1	0	1
Vertical		

3. Edge Magnitude

1. Apply horizontal line detection on the original image (ABSOLUTE postprocess.)
2. Apply vertical line detection on the original image (ABSOLUTE postprocess)
3. Calculate **edge magnitude** at each pixel by adding the two images from 1, 2

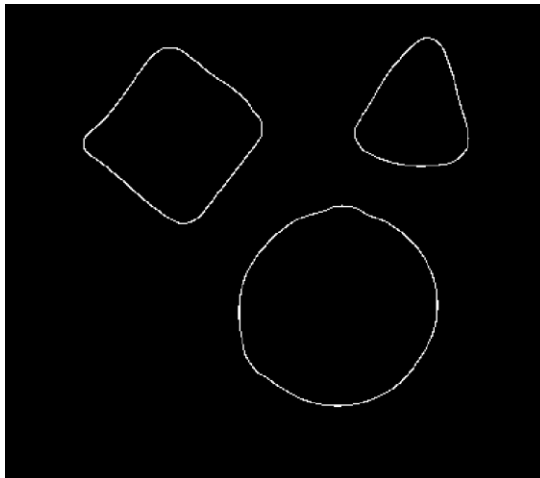
$$\text{Mag}(x,y) = |\text{Hor}(x,y)| + |\text{Ver}(x,y)|$$
4. Apply CUTOFF to trim all values that exceed 255

6- Practical

For the following image, it consists of 3 main objects (triangle, rotated square, and circle) and some other noise (dots and lines). The required task is to extract the boundary pixels of the 3 main objects only and remove all noise using methods mention in this lab



The result should look like this:



Helper:

`J = medfilt2(I,[m n])` performs median filtering, where each output pixel contains the median value in the m-by-n neighborhood around the corresponding pixel in the input image.

`BW = im2bw(I,level)` converts the grayscale image `I` to binary image `BW`, by replacing all pixels in the input image with luminance greater than `level` with the value 1 (white) and replacing all other pixels with the value 0 (black).

`BW = edge(I,method)` detects edges in image `I` using the edge-detection algorithm specified by `method`.