Module 4

Image Filtering

▼ What is image filtering used for?

smoothing, sharpening, removing noise, and edge detection

▼ What is image filtering?

a process that modifies all the pixels within a local neighborhood

▼ What is a filter?

A filter is defined by a kernel, which is a small, square matrix with an odd number (3, 5, 7, etc.) of elements in each dimension.

▼ What is a convolution?

- the process of applying filters to an image
- the kernel is applied to each pixel and its neighbors within an image with the center of the kernel aligned with the current pixel
- when you apply a filter, you want to enhance image properties and/or aid in the extraction of feature information such as edges and corners

▼ What is low pass filtering (or image blurring)?

convolving an image with a low pass filter kernel to remove high frequency content such as noise and edges from an image

▼ What is mean filtering?

- This technique is done by convolving an image with a normalized box filter.

- It simply **replaces each pixel with the average pixel value of it and a neighborhood window of adjacent pixels**.
- The effect is a more smooth image with sharp features removed.

▼ What is median filtering?

- This technique is done by **replacing each pixel with the median pixel value of it and a neighborhood window of adjacent pixels**.
- This is highly effective against salt-and-pepper noise in an image.
- In mean filtering, the central element is a newly calculated value which may be a pixel value in the image or a new value.
- But in median filtering, the central element is always replaced by some pixel value already in the image. It reduces the noise effectively

▼ What is Gaussian Blurring?

- This technique **uses a kernel that represents the shape of a Gaussian (or bell-shaped) curve.**
- The degree of smoothing is determined by the standard deviation of the Gaussian.
- Larger standard deviation Gaussians require larger convolution kernels in order to be accurately represented.
- The Gaussian **outputs a weighted average of each pixel's neighborhood, with the average weighted more towards the value of the central pixels**.
- Because of this, a Gaussian provides gentler smoothing and preserves edges better than a similarly sized mean filter.
- Gaussian blurring is highly effective in removing Gaussian noise from an image.

▼ What is high pass filtering?

- convolving an image with a high pass filter kernel
- high pass filtering retains the high frequency content of an image
- the kernel of a high pass filter is **designed to increase the contrast between light and dark regions and bring out image features**

▼ What is Laplacian filtering?

- Laplacian filtering, which involves the Laplacian operator (a second derivative operation), is one implementation of a high pass filter.
- It **eliminates constant and low frequencies leaving only high-frequency edges**.
- For application of this filtering technique in edge-detection, **the output of the Laplacian operator is subtracted from the original image to produce edge enhancement or sharpening of that image.**

▼ What is the Positive Laplacian operator?

- The Positive Laplacian operator is **used to take out outward edges in an image**.
- For Positive Laplacian, the kernel matrix's center element should be negative, corner elements should be zero and the rest should be 1.

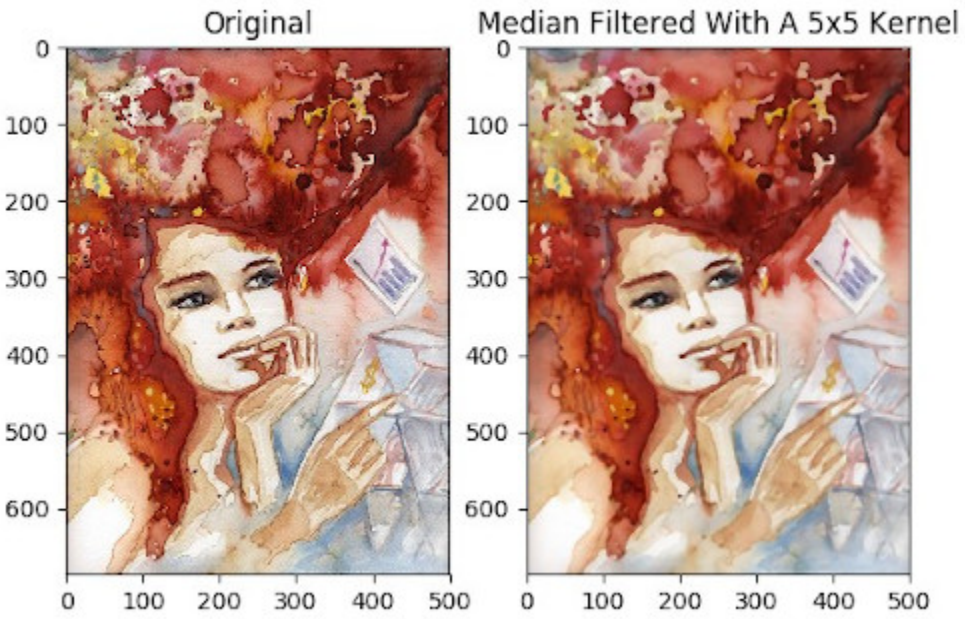
▼ What is the Negative Laplacian operator?

- The Negative Laplacian operator is **used to take out inward edges in an image**.
- For Negative Laplacian, the kernel matrix's center element should be positive, corner elements should be zero and the rest should be -1.

▼ OpenCV example: convolves the *portrait.jpg* image with a 5x5 median filter

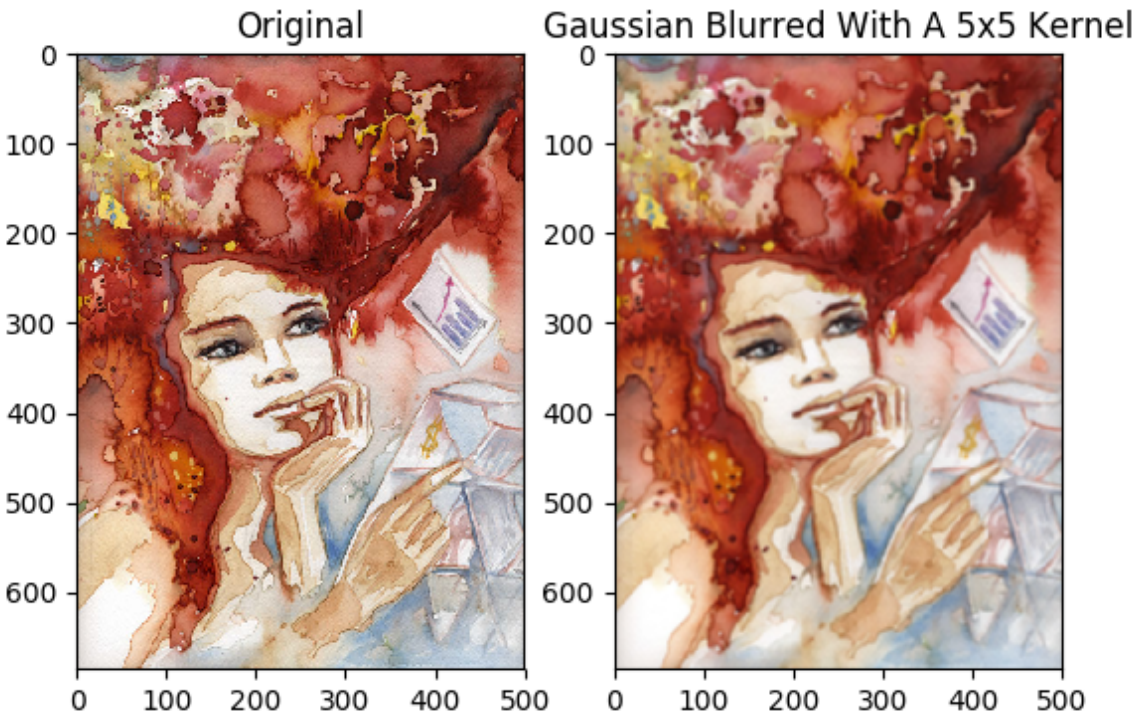
```
import numpy as np
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('portrait.jpg')

blur = cv2.medianBlur(img, 5)

fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2)
# when showing images in matplotlib, convert image from BGR to RGB
ax1.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
ax1.set_title('Original')
ax2.imshow(cv2.cvtColor(blur, cv2.COLOR_BGR2RGB))
ax2.set_title('Median Filtered With A 5x5 Kernel')
```

```
plt.show()
```



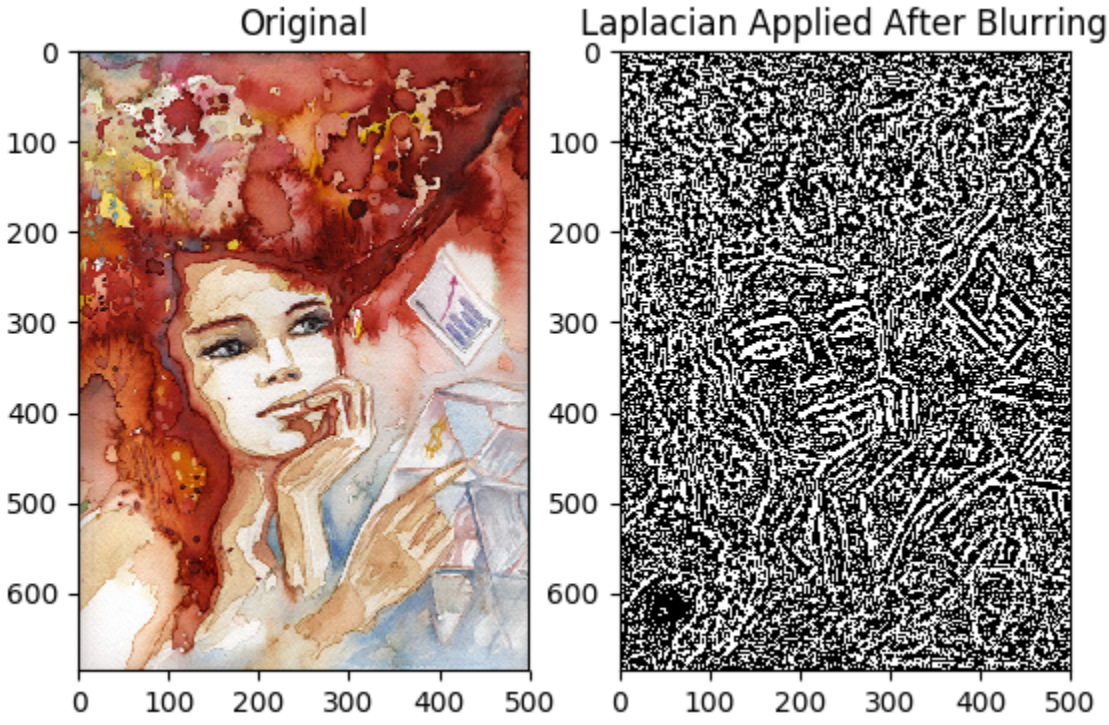
▼ OpenCV example: convolves the *portrait.jpg* image with a 5x5 Gaussian kernel

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('portrait.jpg')

blur = cv2.GaussianBlur(img, (5,5), 3)

fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2)
# when showing images in matplotlib, convert image from BGR to RGB
ax1.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
ax1.set_title('Original')
ax2.imshow(cv2.cvtColor(blur, cv2.COLOR_BGR2RGB))
ax2.set_title('Gaussian Blurred With A 5x5 Kernel')

plt.show()
```



▼ OpenCV example: applying a Laplacian operator

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('portrait.jpg')

blur = cv2.GaussianBlur(img, (7,7), 10)
gray = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)

laplacian = cv2.Laplacian(gray, cv2.CV_32F)


fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2)
# when showing images in matplotlib, convert image from BGR to RGB
ax1.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
ax1.set_title('Original')
ax2.imshow(cv2.cvtColor(laplacian, cv2.COLOR_BGR2RGB))
ax2.set_title('Laplacian Applied After Blurring')

plt.show()
```


Original
Laplacian Applied After Blurring
0
100
200
300
400
500
600
0
100
200
300
400
500