Numerical Programming
TTF 5

Install libraries with pip

```
pip install opencv-python
pip install numpy
pip install matplotlib
```

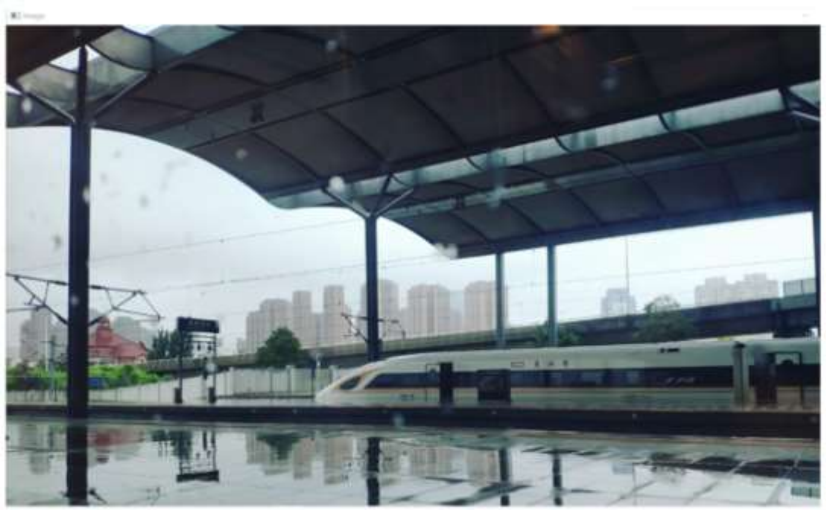
Reading images

```python
import cv2

# Load the image
image = cv2.imread("jg.png", cv2.IMREAD_GRAYSCALE)

# Display the image
cv2.imshow("Image", image)

# Wait for the user to press a key
cv2.waitKey(0)

# Close all windows
cv2.destroyAllWindows()
```

- 1. Read an image using imread() function.
- 2. Create a GUI window and display image using imshow() function.
- 3. Use function waitkey(0) to hold the image window on the screen by the specified number of seconds, o means till the user closes it, it will hold GUI window on the screen.

When you use cv2.imread(filename, cv2.IMREAD_GRAYSCALE), the image is converted into a single-channel grayscale image where pixel values represent shades of gray (0 for black, 255 for white, and values in between for different shades)

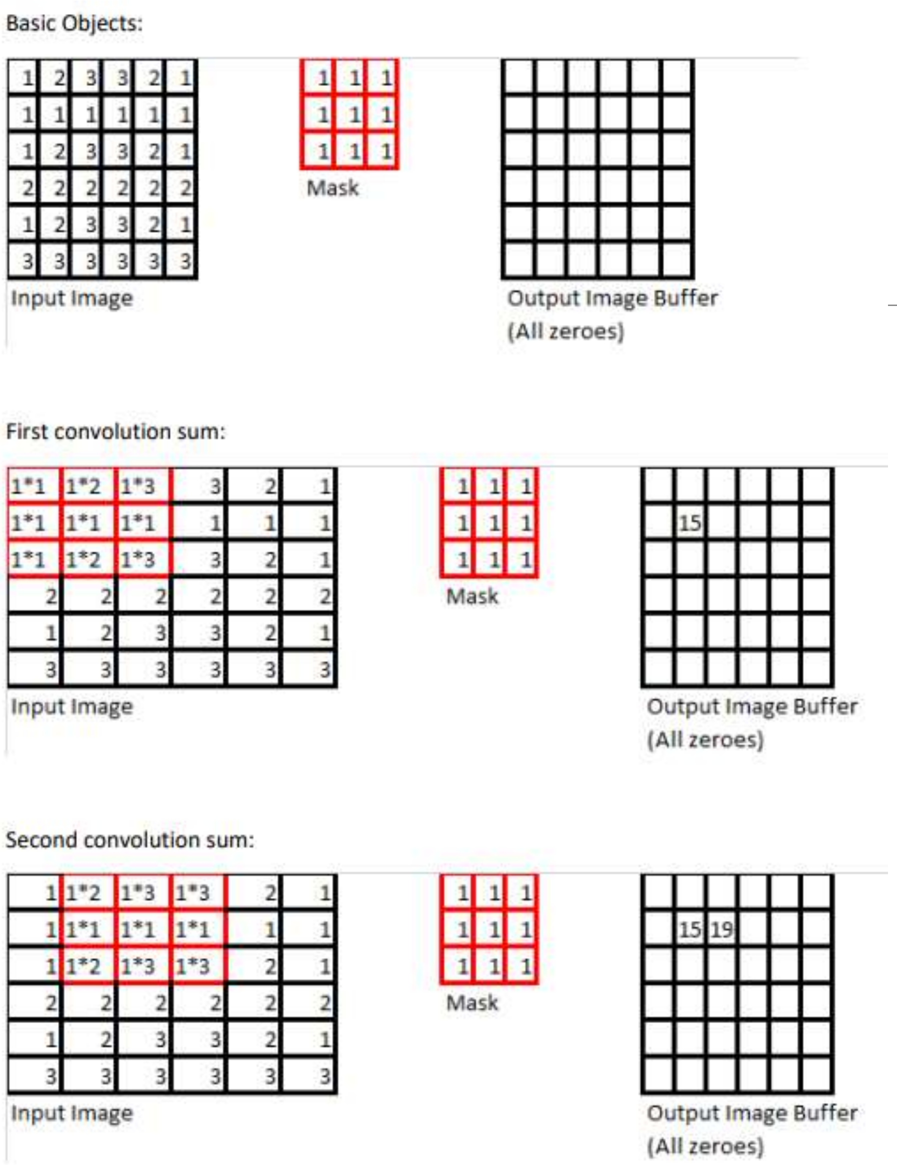
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('dumb.jpg', cv2.IMREAD_GRAYSCALE)

cv2.imshow('image',img)
cv2.waitKey(0)
```



cv2.VideoCapture

```python
import cv2

cap = cv2.VideoCapture('1.mp4')

if not cap.isOpened():
    print("Error: Could not open video.")
    exit()

i = 0
while True:
    ret, frame = cap.read()

    if not ret:
        print("End of video or error reading frame.")
        break

    cv2.imwrite(f'{i}_frame.jpg', frame)
    i += 1

cap.release()
print(f"Frames saved: {i}")
```

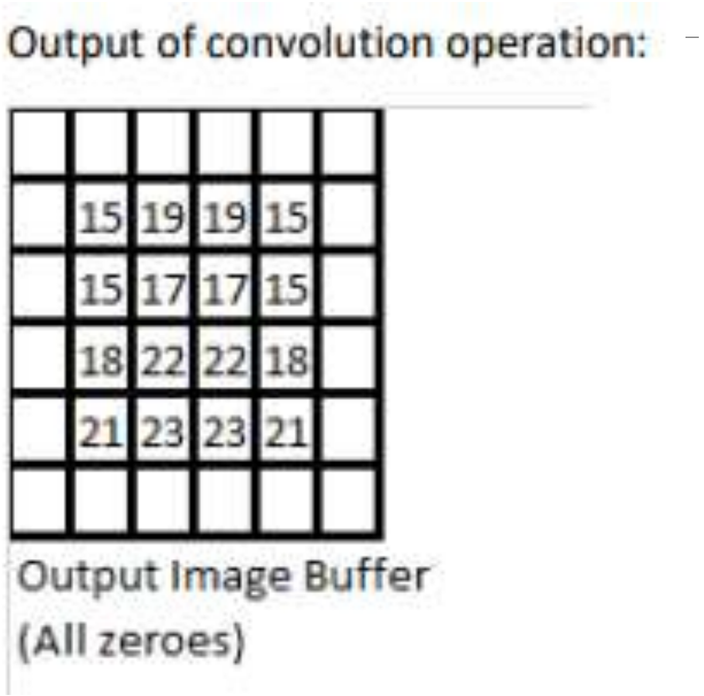
Convolution



Convolution is a mathematical operation applied to images that blends a small matrix (kernel) with the image to perform effects like blurring, sharpening, or edge detection.

A kernel is a small matrix, typically 3x3 or 5x5, used to scan over the image.

The kernel slides over the entire image. At each position, the kernel overlaps with part of the image, multiplying corresponding values, then summing them up to get the output for that position.



Popular filters

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

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$
Sharpening Filter

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

$$\begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}$$
Vertical Scharr Filter

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
Box Blur Filter

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
Vertical Sobel Filter

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$
Emboss Filter

$$\begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$$
Horizontal Scharr Filter

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$
Gaussian Blur Filter

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$
Horizontal Sobel Filter

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$
Outline Filter

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load the image
image = cv2.imread('path_to_your_image.jpg', cv2.IMREAD_GRAYSCALE)

# Apply Gaussian Blur
gaussian_blur = cv2.GaussianBlur(image, (3, 3), 0)

# Apply Sobel filter (Edge detection in X and Y directions)
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)  # Sobel in x-direction
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)  # Sobel in y-direction
sobel_combined = cv2.magnitude(sobel_x, sobel_y)  # Combine both

# Apply Scharr filter (Edge detection in X and Y directions)
scharr_x = cv2.Scharr(image, cv2.CV_64F, 1, 0)
scharr_y = cv2.Scharr(image, cv2.CV_64F, 0, 1)
scharr_combined = cv2.magnitude(scharr_x, scharr_y)

# Apply Laplacian filter
laplacian = cv2.Laplacian(image, cv2.CV_64F)

# Display results using Matplotlib
plt.figure(figsize=(10, 8))
```

```python
plt.subplot(231), plt.imshow(image, cmap='gray'), plt.title('Original Image')
plt.subplot(232), plt.imshow(gaussian_blur, cmap='gray'), plt.title('Gaussian Blur')
plt.subplot(233), plt.imshow(sobel_combined, cmap='gray'), plt.title('Sobel Filter')
plt.subplot(234), plt.imshow(scharr_combined, cmap='gray'), plt.title('Scharr Filter')
plt.subplot(235), plt.imshow(laplacian, cmap='gray'), plt.title('Laplacian Filter')

plt.show()
```

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(x) + \frac{h^5}{5!}f^{(5)}(x) + \cdots$$

Change h to $-h$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(x) - \frac{h^5}{5!}f^{(5)}(x) + \cdots$$

Subtract:

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{2h^3}{3!}f'''(x) + \frac{2h^5}{5!}f^{(5)}(x) + \cdots$$

Divide by $2h$:

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{h^2}{3!}f'''(x) + \frac{h^4}{5!}f^{(5)}(x) + \cdots$$

Move around :

$$\underset{\substack{\uparrow \\ \mathbf{A}}}{f'(x)} = \underset{\substack{\uparrow \\ \mathbf{D(h)}}}{\frac{f(x+h) - f(x-h)}{2h}} - \frac{h^2}{3!}f'''(x) - \frac{h^4}{5!}f^{(5)}(x) - \cdots$$

in short we can write:

$$A = D(h) + \gamma_1 h^2 + \gamma_2 h^4 + \cdots \qquad (1)$$

Now let us change h to $\frac{h}{2}$:

$$A = D(\frac{h}{2}) + \gamma_1 \left(\frac{h}{2}\right)^2 + \gamma_2 \left(\frac{h}{2}\right)^4 + \cdots$$

equivalently :

$$A = D(\frac{h}{2}) + \gamma_1 \frac{h^2}{4} + \gamma_2 \frac{h^4}{16} + \cdots$$

multiply by 4 :

$$4A = 4D(\frac{h}{2}) + \gamma_1 h^2 + \gamma_2 \frac{h^4}{4} + \cdots \qquad (2)$$

so in this discussion we have had these two:

$$\begin{cases} A &= D(h) + \gamma_1 h^2 + \gamma_2 h^4 + \cdots & (1) \\ 4A &= 4D(\frac{h}{2}) + \gamma_1 h^2 + \gamma_2 \frac{h^4}{4} + \cdots & (2) \end{cases}$$

subtract (1) from (2) :

$$3A = \left\{4D(\frac{h}{2}) - D(h)\right\} - \frac{3}{4}\gamma_2 h^4 + \cdots$$

divide by 3 :

$$A = \frac{1}{3}\left\{4D(\frac{h}{2}) - D(h)\right\} - \frac{1}{4}\,\gamma_2\,h^4 + \cdots$$

so now we have found an approximation for A whose error is of a better order of $O(h^4)$

$$\mathbf{A} \;\approx\; \tfrac{\mathbf{1}}{\mathbf{3}}\left\{\mathbf{4D}(\tfrac{\mathbf{h}}{\mathbf{2}}) - \mathbf{D}(\mathbf{h})\right\} \qquad \textbf{with truncation error of order } \mathbf{O}(\mathbf{h}^4)$$

This technique of improving the approximation is called **Richardson extrapolation**. We will use it in numerical integration as well.

In the particular case of $D(h) = \frac{f(x+h)-f(x-h)}{2h}$ we get:

$$D(\frac{h}{2}) = \frac{f(x+\frac{h}{2}) - f(x-\frac{h}{2})}{2(\frac{h}{2})} = \frac{f(x+\frac{h}{2}) - f(x-\frac{h}{2})}{h}$$

so then :

$$\frac{1}{3}\left\{4\,D(\frac{h}{2}) - D(h)\right\} = \frac{1}{3}\left\{\frac{4\left\{f(x+\frac{h}{2}) - f(x-\frac{h}{2})\right\}}{h} - \frac{f(x+h) - f(x-h)}{2h}\right\}$$

$$= \frac{8f(x+\frac{h}{2}) - 8f(x-\frac{h}{2}) - f(x+h) + f(x-h)}{6h}$$

so :

$$A \approx \frac{f(x-h) - 8f(x-\frac{h}{2}) + 8f(x+\frac{h}{2}) - f(x+h)}{6h} + O(h^4)$$

Richardson's Extrapolation for $f(x) = x^2$

We want to approximate the derivative of $f(x) = x^2$ at $x = 1$ using Richardson's Extrapolation.

The central difference formula for the derivative is given by:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

We compute the derivative at $x = 1$ using two step sizes $h_1 = 0.1$ and $h_2 = 0.05$.

Step 1: Approximation using $h_1 = 0.1$

For $h_1 = 0.1$, the approximation is:

$$f'(1) \approx \frac{(1+0.1)^2 - (1-0.1)^2}{2 \times 0.1} = \frac{1.21 - 0.81}{0.2} = \frac{0.4}{0.2} = 2$$

Step 2: Approximation using $h_2 = 0.05$

For $h_2 = 0.05$, the approximation is:

$$f'(1) \approx \frac{(1+0.05)^2 - (1-0.05)^2}{2 \times 0.05} = \frac{1.1025 - 0.9025}{0.1} = \frac{0.2}{0.1} = 2$$

Step 3: Richardson's Extrapolation

Assuming the error is of order $O(h^2)$, we apply Richardson's Extrapolation:

$$f_{\text{extra}}(1) = \frac{4 \cdot f'(1, h_2) - f'(1, h_1)}{3}$$

Substituting the values:

$$f_{\text{extra}}(1) = \frac{4 \cdot 2 - 2}{3} = \frac{8 - 2}{3} = \frac{6}{3} = 2$$

Thus, the extrapolated derivative is $f'(1) = 2$, which is the exact value.

Consider the following function:

1. Write the central difference approximation for the derivative $f'(x)$ at $x = 1$ using a step size $h = 0.1$. 2. Use Richardson extrapolation to improve the approximation to an error of $O(h^4)$. 3. Write out the calculations and the final answer.

$$f(x) = \cos(x)$$