

**Computer Vision**

**MCA-574**

**Lab – 03**

***BY***

**HIMANSHU HEDA (24225013)**

**SUBMITTED TO**

**Dr. Preety Shoran**

**SCHOOL OF SCIENCES**

**2024-25**

**# Computer Vision Lab 2: Image Filtering & Effects**

This notebook demonstrates various image processing techniques using OpenCV and custom kernels:

1. Edge Detection & Sharpening

2. Blurring & Smoothing

3. Embossing & Outline Effect

4. Sobel & Laplacian Transformation

5. High-pass & Low-pass Filtering

6. Motion Detection Effect

7. Custom Artistic Filters

The sample image used is 'pika wallpaper.webp'.

# Import required libraries

import cv2

import numpy as np

from matplotlib import pyplot as plt

# Load image

image = cv2.imread('pika wallpaper.webp')

if image is None:

    raise Exception('Image not found!')

image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

plt.imshow(image\_rgb)

plt.title('Original Image')

plt.axis('off')

plt.show()

****

# 1. Edge Detection & Sharpening

# Edge Detection Kernel

edge\_kernel = np.array([[-1, -1, -1],

                        [-1, 8, -1],

                        [-1, -1, -1]])

# Sharpening Kernel

sharpen\_kernel = np.array([[0, -1, 0],

                           [-1, 5, -1],

                           [0, -1, 0]])

edge\_img = cv2.filter2D(image, -1, edge\_kernel)

sharpen\_img = cv2.filter2D(image, -1, sharpen\_kernel)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)

plt.imshow(cv2.cvtColor(edge\_img, cv2.COLOR\_BGR2RGB))

plt.title('Edge Detection')

plt.axis('off')

plt.subplot(1,2,2)

plt.imshow(cv2.cvtColor(sharpen\_img, cv2.COLOR\_BGR2RGB))

plt.title('Sharpened')

plt.axis('off')

plt.show()

****

# 2. Blurring & Smoothing

# Average Filter

avg\_blur = cv2.blur(image, (7,7))

# Gaussian Filter

gauss\_blur = cv2.GaussianBlur(image, (7,7), 0)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)

plt.imshow(cv2.cvtColor(avg\_blur, cv2.COLOR\_BGR2RGB))

plt.title('Average Blur')

plt.axis('off')

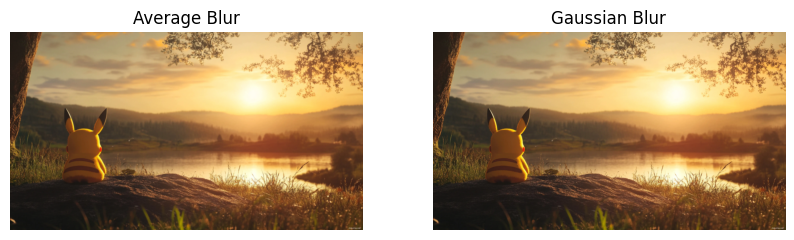
plt.subplot(1,2,2)

plt.imshow(cv2.cvtColor(gauss\_blur, cv2.COLOR\_BGR2RGB))

plt.title('Gaussian Blur')

plt.axis('off')

plt.show()

****

# 3. Embossing & Outline Effect

# Emboss Kernel

emboss\_kernel = np.array([[-2, -1, 0],

                          [-1, 1, 1],

                          [0, 1, 2]])

# Outline Kernel

outline\_kernel = np.array([[-1, -1, -1],

                           [-1, 8, -1],

                           [-1, -1, -1]])

emboss\_img = cv2.filter2D(image, -1, emboss\_kernel)

outline\_img = cv2.filter2D(image, -1, outline\_kernel)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)

plt.imshow(cv2.cvtColor(emboss\_img, cv2.COLOR\_BGR2RGB))

plt.title('Emboss Effect')

plt.axis('off')

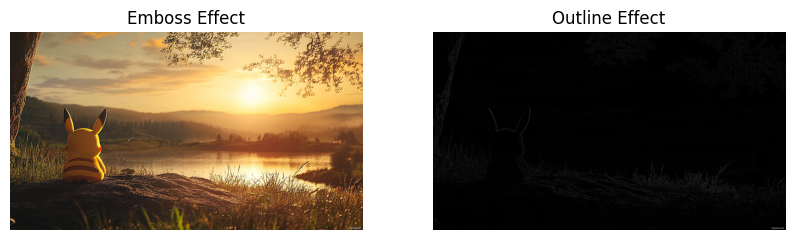
plt.subplot(1,2,2)

plt.imshow(cv2.cvtColor(outline\_img, cv2.COLOR\_BGR2RGB))

plt.title('Outline Effect')

plt.axis('off')

plt.show()

****

# 4. Sobel & Laplacian Transformation

# Sobel X

sobelx = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=5)

# Sobel Y

sobely = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=5)

# Laplacian

laplacian = cv2.Laplacian(image, cv2.CV\_64F)

plt.figure(figsize=(15,4))

plt.subplot(1,3,1)

plt.imshow(np.abs(sobelx).astype(np.uint8))

plt.title('Sobel X')

plt.axis('off')

plt.subplot(1,3,2)

plt.imshow(np.abs(sobely).astype(np.uint8))

plt.title('Sobel Y')

plt.axis('off')

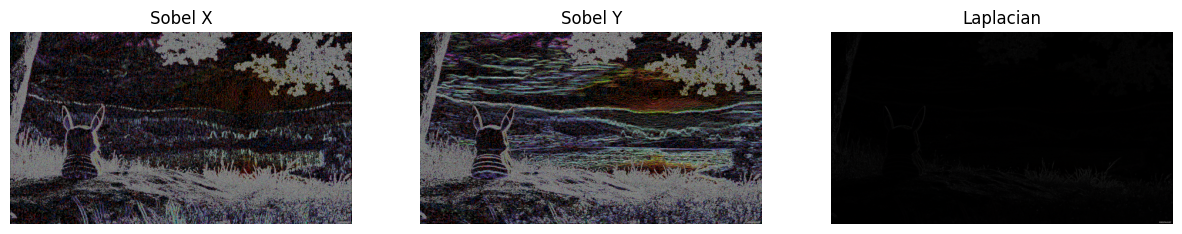
plt.subplot(1,3,3)

plt.imshow(np.abs(laplacian).astype(np.uint8))

plt.title('Laplacian')

plt.axis('off')

plt.show()

****

# 5. High-pass & Low-pass Filtering

# High-pass kernel

high\_pass = np.array([[-1, -1, -1],

                     [-1, 9, -1],

                     [-1, -1, -1]])

# Low-pass kernel

low\_pass = np.ones((5,5), np.float32) / 25

high\_pass\_img = cv2.filter2D(image, -1, high\_pass)

low\_pass\_img = cv2.filter2D(image, -1, low\_pass)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)

plt.imshow(cv2.cvtColor(high\_pass\_img, cv2.COLOR\_BGR2RGB))

plt.title('High-pass Filter')

plt.axis('off')

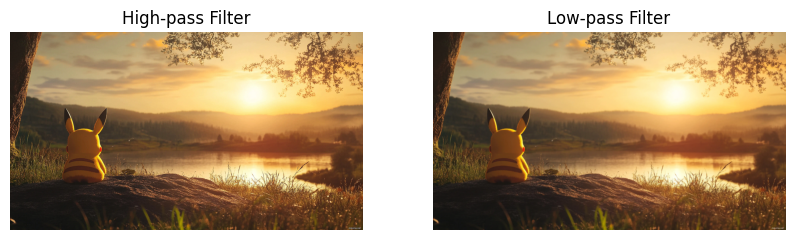
plt.subplot(1,2,2)

plt.imshow(cv2.cvtColor(low\_pass\_img, cv2.COLOR\_BGR2RGB))

plt.title('Low-pass Filter')

plt.axis('off')

plt.show()

****

# 6. Motion Detection Effect

# Motion Blur Kernel

motion\_blur\_kernel = np.zeros((15, 15))

motion\_blur\_kernel[7, :] = np.ones(15)

motion\_blur\_kernel = motion\_blur\_kernel / 15

motion\_blur\_img = cv2.filter2D(image, -1, motion\_blur\_kernel)

# Diagonal Edge Kernel

diag\_edge\_kernel = np.array([[2, -1, -1],

                             [-1, 2, -1],

                             [-1, -1, 2]])

diag\_edge\_img = cv2.filter2D(image, -1, diag\_edge\_kernel)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)

plt.imshow(cv2.cvtColor(motion\_blur\_img, cv2.COLOR\_BGR2RGB))

plt.title('Motion Blur')

plt.axis('off')

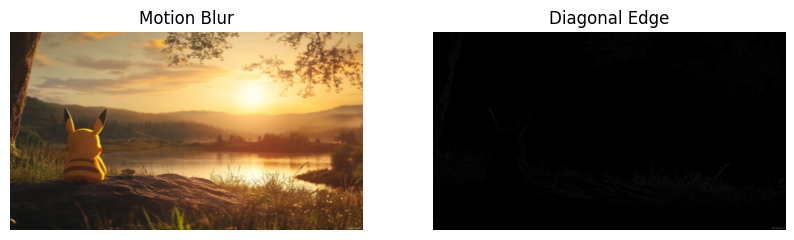
plt.subplot(1,2,2)

plt.imshow(cv2.cvtColor(diag\_edge\_img, cv2.COLOR\_BGR2RGB))

plt.title('Diagonal Edge')

plt.axis('off')

plt.show()

****

# 7. Custom Artistic Filters (Sharpen + Emboss)

custom\_kernel = sharpen\_kernel + emboss\_kernel

custom\_art\_img = cv2.filter2D(image, -1, custom\_kernel)

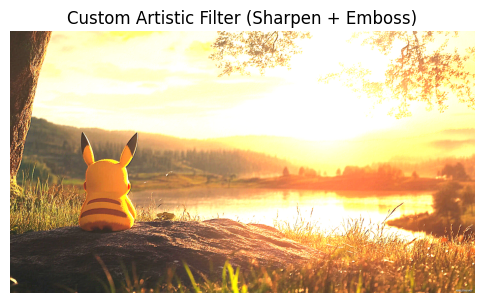
plt.figure(figsize=(6,6))

plt.imshow(cv2.cvtColor(custom\_art\_img, cv2.COLOR\_BGR2RGB))

plt.title('Custom Artistic Filter (Sharpen + Emboss)')

plt.axis('off')

plt.show()

****