**Basic Image Operations (Reading, Writing, Resizing, and Grayscale Conversion)**

This lab covers fundamental image operations like loading, resizing, and converting images.

import cv2

import numpy as np

# Load an image

image = cv2.imread("sample.jpg") # Replace with your image path

# Convert to grayscale

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Resize image

resized\_image = cv2.resize(image, (300, 300))

# Save processed images

cv2.imwrite("grayscale.jpg", gray\_image)

cv2.imwrite("resized.jpg", resized\_image)

# Display images

cv2.imshow("Original", image)

cv2.imshow("Grayscale", gray\_image)

cv2.imshow("Resized", resized\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Histogram Equalization (Improve Image Contrast)**

Enhances image contrast using histogram equalization.

import cv2

import numpy as np

# Load an image in grayscale

image = cv2.imread("sample.jpg", cv2.IMREAD\_GRAYSCALE)

# Apply histogram equalization

equalized = cv2.equalizeHist(image)

# Display images

cv2.imshow("Original", image)

cv2.imshow("Equalized", equalized)

cv2.waitKey(0)

cv2.destroyAllWindows()

## ****Edge Detection using Canny Algorithm****

Detects edges in an image using the **Canny** edge detector.

import cv2

import numpy as np

# Load an image

image = cv2.imread("sample.jpg", cv2.IMREAD\_GRAYSCALE)

# Apply Gaussian blur

blurred = cv2.GaussianBlur(image, (5, 5), 0)

# Apply Canny edge detection

edges = cv2.Canny(blurred, threshold1=50, threshold2=150)

# Display images

cv2.imshow("Original", image)

cv2.imshow("Edges", edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

## ****Image Filtering using Gaussian, Median, and Bilateral Filters****

This lab demonstrates different smoothing techniques.

import cv2

import numpy as np

# Load an image

image = cv2.imread("sample.jpg")

# Apply different filters

gaussian\_blur = cv2.GaussianBlur(image, (5, 5), 0)

median\_blur = cv2.medianBlur(image, 5)

bilateral\_filter = cv2.bilateralFilter(image, 9, 75, 75)

# Display images

cv2.imshow("Original", image)

cv2.imshow("Gaussian Blur", gaussian\_blur)

cv2.imshow("Median Blur", median\_blur)

cv2.imshow("Bilateral Filter", bilateral\_filter)

cv2.waitKey(0)

cv2.destroyAllWindows()

## ****Image Thresholding (Binary, Adaptive & Otsu's Method)****

Thresholding is useful for segmenting objects from the background.

import cv2

import numpy as np

# Load an image in grayscale

image = cv2.imread("sample.jpg", cv2.IMREAD\_GRAYSCALE)

# Apply simple binary thresholding

\_, binary\_thresh = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

# Apply Otsu's thresholding

\_, otsu\_thresh = cv2.threshold(image, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

# Apply adaptive thresholding

adaptive\_thresh = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,

cv2.THRESH\_BINARY, 11, 2)

# Display images

cv2.imshow("Original", image)

cv2.imshow("Binary Threshold", binary\_thresh)

cv2.imshow("Otsu Threshold", otsu\_thresh)

cv2.imshow("Adaptive Threshold", adaptive\_thresh)

cv2.waitKey(0)

cv2.destroyAllWindows()

## ****Morphological Operations (Erosion, Dilation, Opening, Closing)****

Used to remove noise and enhance object structures.

import cv2

import numpy as np

# Load an image

image = cv2.imread("sample.jpg", cv2.IMREAD\_GRAYSCALE)

# Create a kernel

kernel = np.ones((5,5), np.uint8)

# Apply morphological operations

erosion = cv2.erode(image, kernel, iterations=1)

dilation = cv2.dilate(image, kernel, iterations=1)

opening = cv2.morphologyEx(image, cv2.MORPH\_OPEN, kernel) # Removes noise

closing = cv2.morphologyEx(image, cv2.MORPH\_CLOSE, kernel) # Fills gaps

# Display images

cv2.imshow("Original", image)

cv2.imshow("Erosion", erosion)

cv2.imshow("Dilation", dilation)

cv2.imshow("Opening", opening)

cv2.imshow("Closing", closing)

cv2.waitKey(0)

cv2.destroyAllWindows()

## ****Contour Detection and Drawing****

Finds and highlights objects in an image.

import cv2

import numpy as np

# Load an image

image = cv2.imread("sample.jpg")

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Find edges using Canny

edges = cv2.Canny(gray, 50, 150)

# Find contours

contours, \_ = cv2.findContours(edges, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Draw contours

cv2.drawContours(image, contours, -1, (0, 255, 0), 2)

# Display images

cv2.imshow("Edges", edges)

cv2.imshow("Contours", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

## ****Face Detection using Haar Cascades****

Detects faces in an image.

import cv2

# Load the pre-trained Haar cascade classifier

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml")

# Load an image

image = cv2.imread("face\_sample.jpg")

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Detect faces

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

# Draw rectangles around detected faces

for (x, y, w, h) in faces:

cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 2)

# Display result

cv2.imshow("Face Detection", image)

cv2.waitKey(0)

cv2.destroyAllWindows()