**Lab Report: Computer Vision - Lab 2**

**Student Name:** Anmol Pandey  
**Student ID:** E22CSEU1069  
**Course:** Computer Vision  
**Lab Number:** 2

## ****1. Objective****

The objective of this lab was to understand and implement fundamental image processing techniques using OpenCV. The primary focus was on operations such as image resizing, grayscale conversion, edge detection, blurring, and thresholding. These techniques are essential for preprocessing images in various computer vision applications.

## ****2. Introduction****

OpenCV (Open Source Computer Vision Library) provides powerful tools for image processing. This lab explores basic transformations applied to an image, including:

* **Image Resizing:** Changing the dimensions of an image using different interpolation methods.
* **Grayscale Conversion:** Converting a color image to grayscale to simplify analysis.
* **Edge Detection:** Identifying edges in an image using the Canny edge detector.
* **Blurring:** Reducing noise and details using Gaussian and median blurring techniques.
* **Thresholding:** Binarizing an image to differentiate objects from the background.

## ****3. Implementation Details****

### ****3.1 Libraries Used****

The following Python libraries were used:

* cv2 (OpenCV): For image processing functions.
* numpy: For numerical computations.
* google.colab.patches.cv2\_imshow: For displaying images in Google Colab.

### ****3.2 Image Loading****

import cv2

import numpy as np

from google.colab.patches import cv2\_imshow

# Load an image

image\_path = 'picture.png' # Replace with actual image path

image = cv2.imread(image\_path)

**Explanation:**

* The image is read using cv2.imread().
* The variable image\_path should be set to a valid image file.
* cv2\_imshow() is used for displaying images in Google Colab.

### ****3.3 Image Resizing****

def resize\_image(image, width, height, interpolation\_method):

if interpolation\_method == 'linear':

interpolation = cv2.INTER\_LINEAR

elif interpolation\_method == 'nearest':

interpolation = cv2.INTER\_NEAREST

elif interpolation\_method == 'cubic':

interpolation = cv2.INTER\_CUBIC

else:

raise ValueError("Invalid interpolation method")

resized\_image = cv2.resize(image, (width, height), interpolation=interpolation)

return resized\_image

**Explanation:**

* The function resizes an image based on the specified interpolation method.
* Three interpolation methods are used: Linear, Nearest Neighbor, and Cubic.
* cv2.resize() is applied to resize the image.

### ****3.4 Grayscale Conversion****

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

cv2\_imshow(gray\_image)

**Explanation:**

* Converts the color image to grayscale using cv2.cvtColor().
* Grayscale images simplify processing for tasks like edge detection.

### ****3.5 Edge Detection (Canny Method)****

edges = cv2.Canny(gray\_image, 100, 200)

cv2\_imshow(edges)

**Explanation:**

* cv2.Canny() detects edges using gradient intensity.
* Two threshold values (100 and 200) define strong and weak edges.

### ****3.6 Blurring Techniques****

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

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

cv2\_imshow(gaussian\_blur)

cv2\_imshow(median\_blur)

**Explanation:**

* **Gaussian Blur:** Uses a Gaussian filter to smooth the image.
* **Median Blur:** Uses median filtering to remove noise.

### ****3.7 Image Thresholding****

ret, binary\_threshold = cv2.threshold(gray\_image, 127, 255, cv2.THRESH\_BINARY)

cv2\_imshow(binary\_threshold)

**Explanation:**

* Converts the grayscale image to a binary image.
* Pixels above the threshold (127) are set to 255 (white), and below are set to 0 (black).

## ****4. Results and Observations****

The notebook applies image transformations and displays the results. The observed outputs are:

1. **Original Image**: The loaded image before processing.
2. **Resized Images**: The image resized using different interpolation methods.
3. **Grayscale Image**: A simplified version of the original image.
4. **Edge Detection Output**: Highlights edges present in the image.
5. **Blurred Images**: Images with Gaussian and median blurring applied.
6. **Thresholded Image**: A binary representation of the image.

**Observations:**

* The choice of interpolation method impacts image quality. Nearest neighbor is fastest but produces blocky results, while cubic gives smoother results.
* Grayscale conversion reduces computational complexity for further processing.
* The Canny edge detector effectively highlights image edges, which can be useful in object detection.
* Blurring helps in reducing noise but can also remove important details.
* Thresholding creates a high-contrast binary image, useful for segmentation tasks.

## ****5. Conclusion****

This lab provided hands-on experience with fundamental image processing techniques using OpenCV. Understanding these basic operations is essential for tasks such as feature extraction, object detection, and image preprocessing in deep learning applications.

## ****6. Future Scope****

* Experiment with adaptive thresholding for better segmentation.
* Apply edge detection on real-world datasets.
* Implement morphological operations like erosion and dilation for further image processing.
* Use OpenCV functions for image contour detection.

**End of Report**