**Lab Report: Advanced Computer Vision - Lab 3**

**Student Name:** Anmol Pandey  
**Student ID:** E22CSEU1069  
**Course:** Advanced Computer Vision  
**Lab Number:** 3

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

The objective of this lab was to explore advanced image processing techniques using OpenCV and Scikit-Image. The tasks focused on fundamental image manipulations such as reading, displaying, grayscale conversion, binary thresholding, and edge detection using the Sobel filter.

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

Image processing is a crucial step in computer vision, enabling feature extraction and object detection. This lab involves:

* **Image Loading and Displaying**
* **Color Space Conversion (BGR to RGB and Grayscale)**
* **Binary Thresholding for Segmentation**
* **Edge Detection using Sobel Filtering**

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

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

The following Python libraries were used:

* cv2 (OpenCV): For image processing and manipulations.
* numpy: For numerical computations.
* skimage.filters: For advanced filtering techniques.
* matplotlib.pyplot: For plotting images.

### ****3.2 Image Loading and Display****

import cv2

import numpy as np

from skimage import filters

import matplotlib.pyplot as plt

from google.colab.patches import cv2\_imshow

def task1\_image\_manipulation(image\_path):

# Read and display the image

image\_bgr = cv2.imread(image\_path)

cv2\_imshow(image\_bgr)

**Explanation:**

* The image is read using cv2.imread().
* cv2\_imshow() is used for displaying images in Google Colab.

### ****3.3 RGB Conversion and Image Properties****

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

cv2.imwrite('image\_rgb.jpg', image\_rgb)

height, width, channels = image\_rgb.shape

print(f'Image Dimensions: {width}x{height}, Channels: {channels}')

total\_pixels = height \* width

print(f'Total Pixels: {total\_pixels}')

**Explanation:**

* Converts BGR format (used by OpenCV) to RGB.
* Extracts image dimensions and calculates the total number of pixels.

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

gray\_image = cv2.cvtColor(image\_rgb, cv2.COLOR\_RGB2GRAY)

cv2.imwrite('image\_gray.jpg', gray\_image)

cv2\_imshow(gray\_image)

**Explanation:**

* Converts RGB to grayscale for easier processing and analysis.

### ****3.5 Binary Thresholding****

\_, binary\_image = cv2.threshold(gray\_image, 128, 255, cv2.THRESH\_BINARY)

cv2.imwrite('image\_binary.jpg', binary\_image)

cv2\_imshow(binary\_image)

**Explanation:**

* Converts the grayscale image into a binary image based on a threshold of 128.
* Pixels above 128 are set to 255 (white), and below are set to 0 (black).

### ****3.6 Edge Detection using Sobel Filter****

sobel\_edges = filters.sobel(gray\_image)

plt.imshow(sobel\_edges, cmap='gray')

plt.title("Sobel Edge Detection")

plt.show()

**Explanation:**

* Uses the Sobel filter from skimage.filters to detect edges.
* Displays the edge-detected image using matplotlib.

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

The implemented code produces the following outputs:

1. **Original Image**: The loaded image displayed in its original form.
2. **RGB Image**: The BGR image converted to RGB format.
3. **Grayscale Image**: A single-channel version of the image useful for further processing.
4. **Binary Image**: A thresholded image where objects are clearly segmented.
5. **Edge Detection Output**: Highlights the edges present in the image using the Sobel operator.

**Observations:**

* Converting images to grayscale simplifies processing while preserving structural information.
* Thresholding effectively separates foreground objects from the background.
* The Sobel filter enhances edge features, useful for contour detection and shape analysis.

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

This lab introduced fundamental image processing operations, including reading, color space transformations, thresholding, and edge detection. These techniques are essential building blocks for applications like object recognition and image segmentation.

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

* Experiment with different thresholding techniques like adaptive thresholding.
* Apply other edge detection techniques such as the Canny filter.
* Explore morphological operations like dilation and erosion for noise reduction.
* Implement contour detection to identify distinct objects in an image.

**End of Report**