****

**Superior University Lahore**

**Programming for AI (Lab)**

**Lab Task 5**

**OpenCV Implementation Image Processing**

**BS in Artificial Intelligence**

*Department of Software Engineering*

*Faculty of Computer Science & Information Technology*

*The Superior University, Lahore*

**Submitted by:**

|  |  |
| --- | --- |
| **Student Name:** | **Alisha Kanwal** |
| **Roll Number** | **Su92-bsaim-f23-105** |
| **Section** | **BSAI-4B** |

**Report**

# **Introduction:**

## Problem Statement:

Image processing is a crucial aspect of computer vision, enabling various applications such as object detection, image enhancement, and automation. The objective of this lab task is to apply different image processing techniques using OpenCV and analyze their impact on images.

## Problem Understanding:

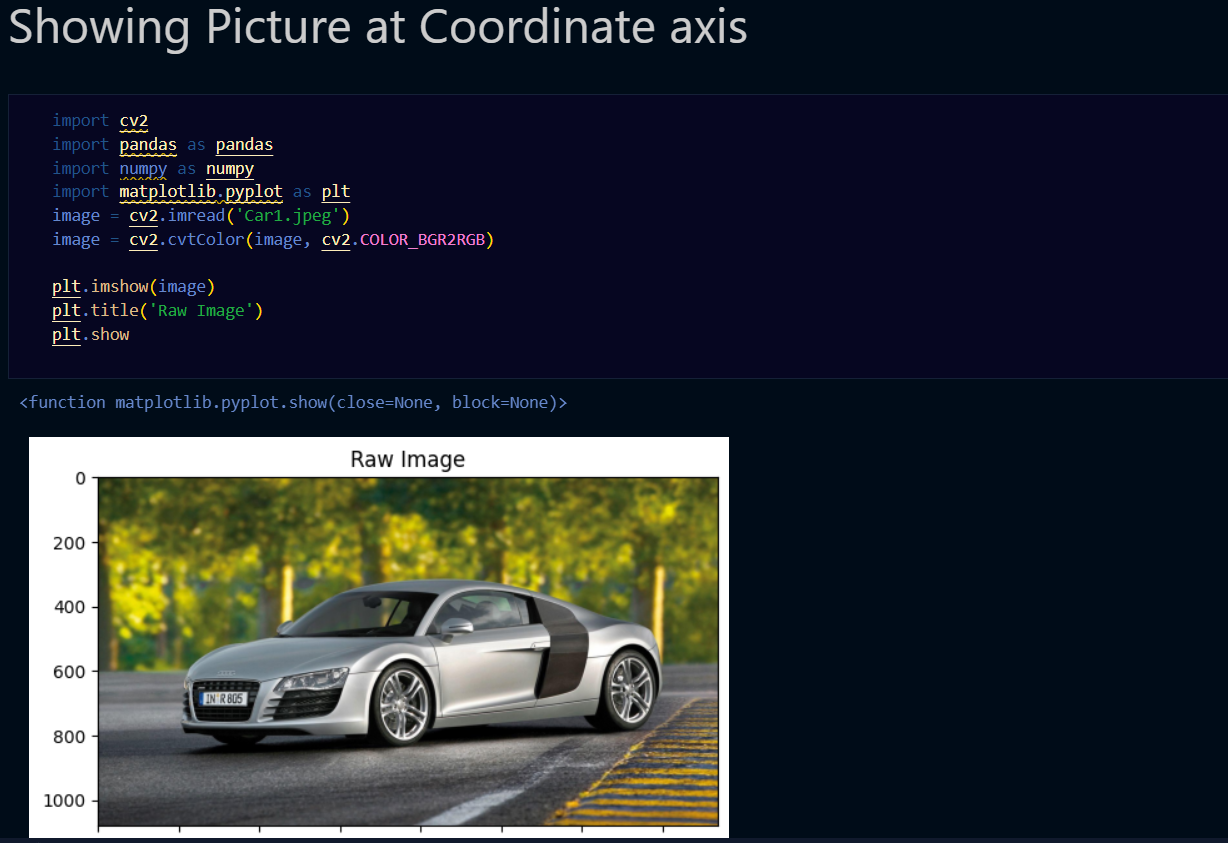
Understanding image processing techniques is essential for applications like computer vision, object recognition, and automation. This task explores fundamental image manipulation operations and their effects on an image. Each technique applied provides insights into how images can be processed for various purposes.

# **Methodology**:

The following steps outline the methodology used in the implementation:

## **Reading and Displaying an Image**

* + An image is loaded using **cv2.imread()** and converted to RGB using **cv2.cvtColor().**
  + The image is displayed using Matplotlib.



## **Image Resizing**

The image is resized to a fixed dimension using **cv2.resize().**

## **Image Cropping**

A region of interest (ROI) is extracted by selecting a portion of the image array.

## **Image Rotation**

The image is rotated counterclockwise using **cv2.rotate().**

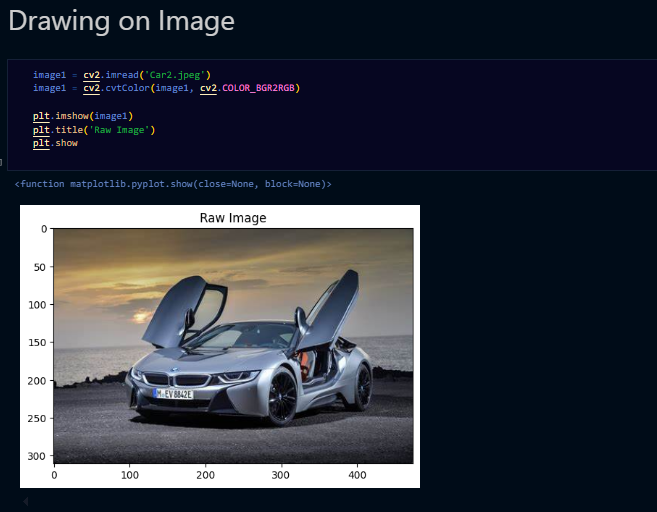
## **Image Flipping**

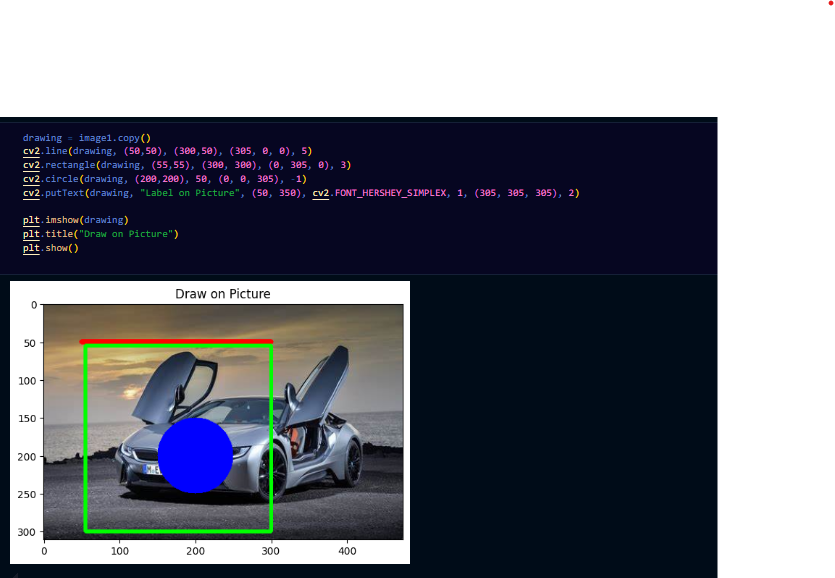
The image is flipped horizontally using **cv2.flip().**

## **Drawing on Image**

Shapes and text are drawn on the image using OpenCV functions like **cv2.line(), cv2.rectangle(), and cv2.putText()**.





## **Grayscale Conversion**

The image is converted to grayscale using **cv2.cvtColor()**.

## **Binary Thresholding**

Thresholding is applied to convert the image to a binary format using **cv2.threshold()**.

## **Edge Detection**

Edges are detected using the Canny Edge Detector **(cv2.Canny())**.



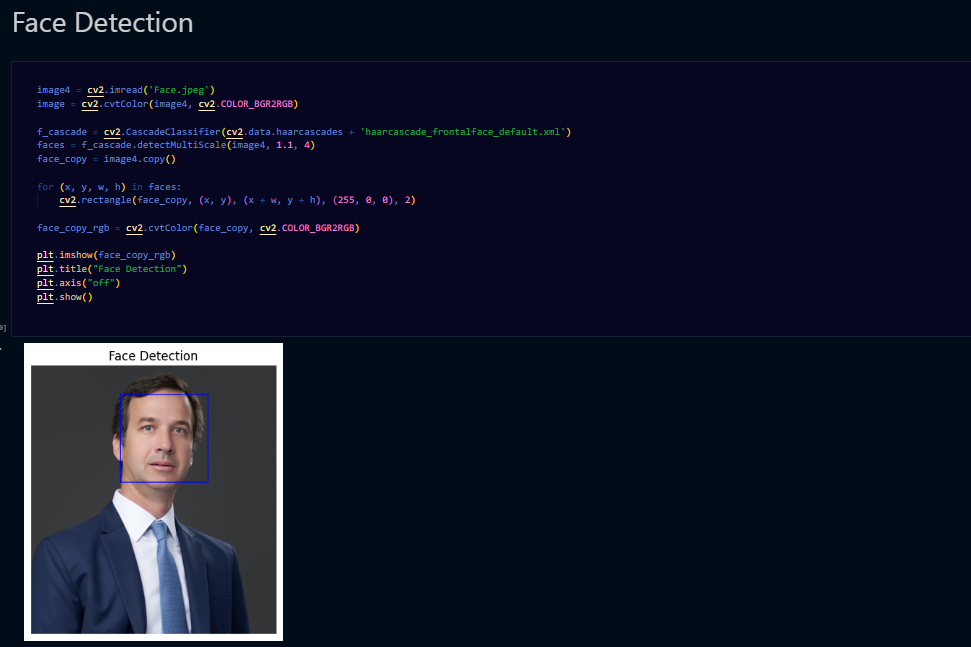
## **Contour Detection**

Contours are identified using **cv2.findContours()** to detect shapes within the image.



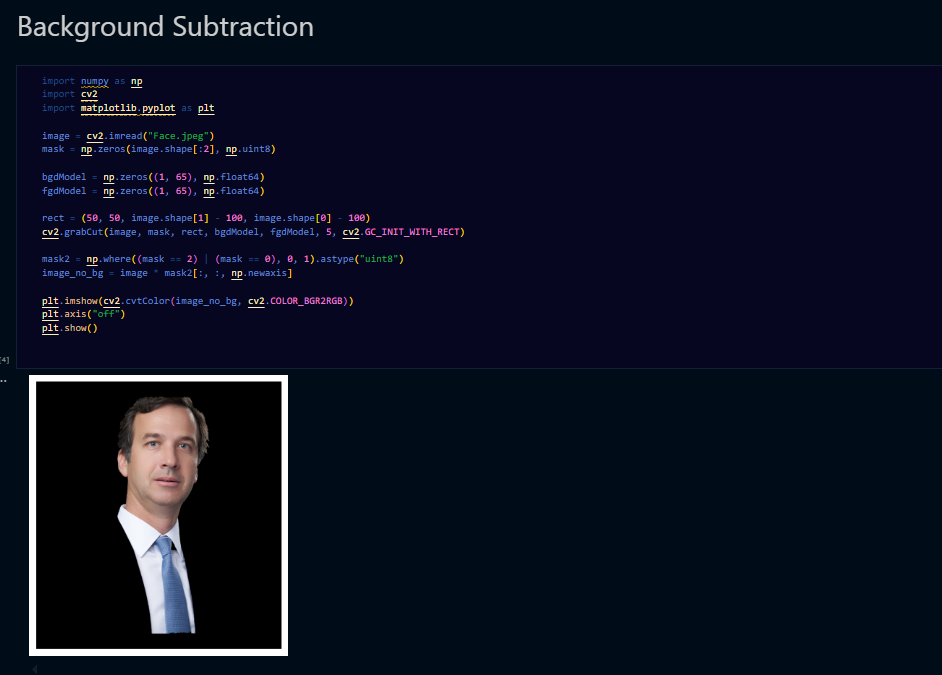
## **Face Detection**

Faces are detected using Haar Cascade Classifiers **(cv2.CascadeClassifier())**.



## **Background Subtraction**

Background subtraction is applied using **cv2.createBackgroundSubtractorMOG2()**.



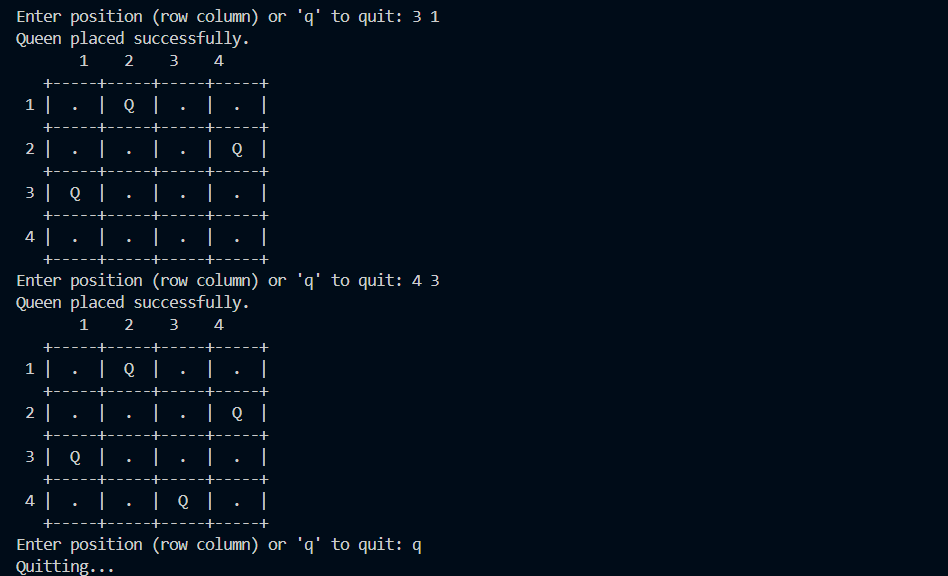
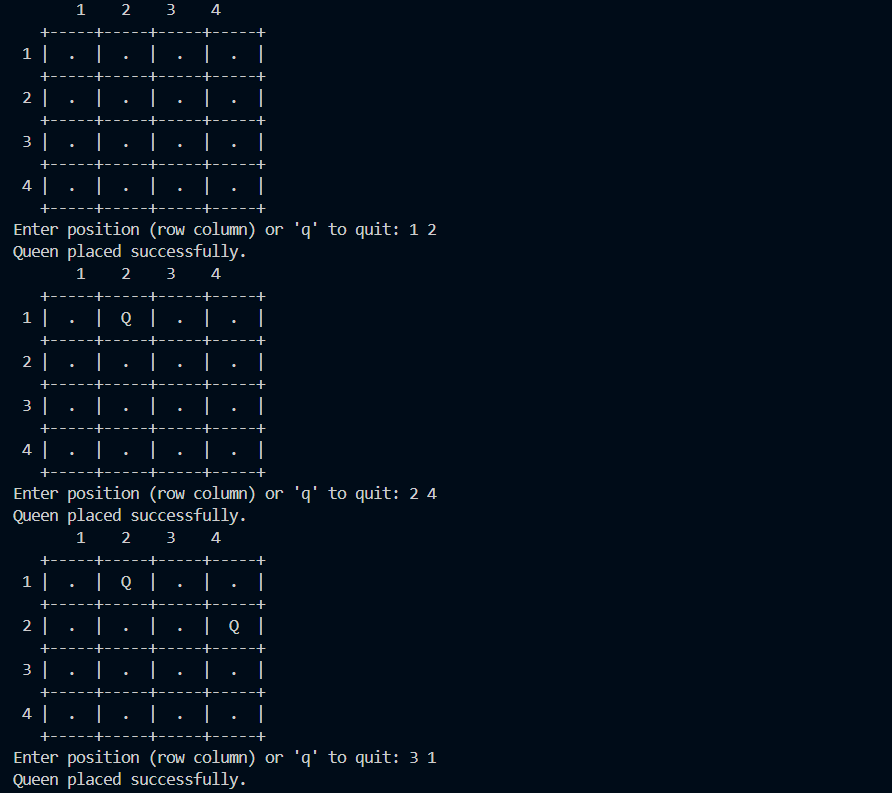
# **Implementation Details:**



Each step is executed using OpenCV functions, and results are displayed using Matplotlib for better visualization. The transformations and detections applied to the image illustrate the versatility of OpenCV in handling image-processing tasks.

1. **Conclusions:**

This lab task demonstrated various image processing techniques in OpenCV, providing hands-on experience in image transformations, feature extraction, and object detection. Future enhancements could include real-time video processing and machine learning-based image classification.



# **Conclusion:**

This interactive N-Queen solver enforces the puzzle’s constraints in real time, allowing users to explore valid placements while learning the rules. Key features include:

* Clear visualization of the board with row/column labels.
* Immediate feedback on invalid moves.
* Support for custom grid dimensions.

The program demonstrates how user-driven problem-solving can be combined with algorithmic validation to create an educational tool for constraint-based puzzles. Future enhancements could include automated solution generation or a win condition for full N-Queen placement.