****

**Superior University Lahore**

**Instructor**: **Sir Rasikh Ali**

**Subject: Programming for AI**

**Task 5**

**Open CV**   
**BS in Artificial Intelligence**

Table of Contents

[**1. Introduction** 1](#_Toc192280850)

[**2. Image Loading & Display** 2](#_Toc192280851)

[**3. Color Conversion** 2](#_Toc192280852)

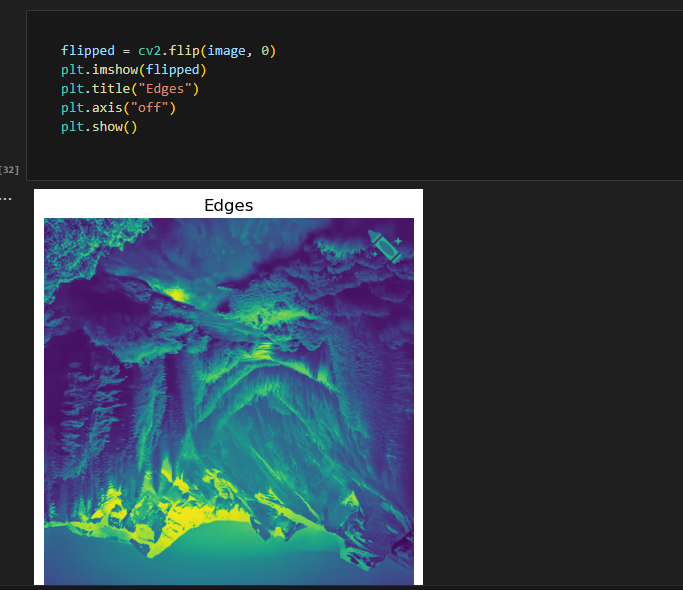
[**4. Image Processing** 4](#_Toc192280853)

[**4.1 Resizing** 4](#_Toc192280854)

[**4.2 Rotation** 5](#_Toc192280855)

[**4.3 Flipping** 6](#_Toc192280856)

[**5. Edge Detection** 6](#_Toc192280857)

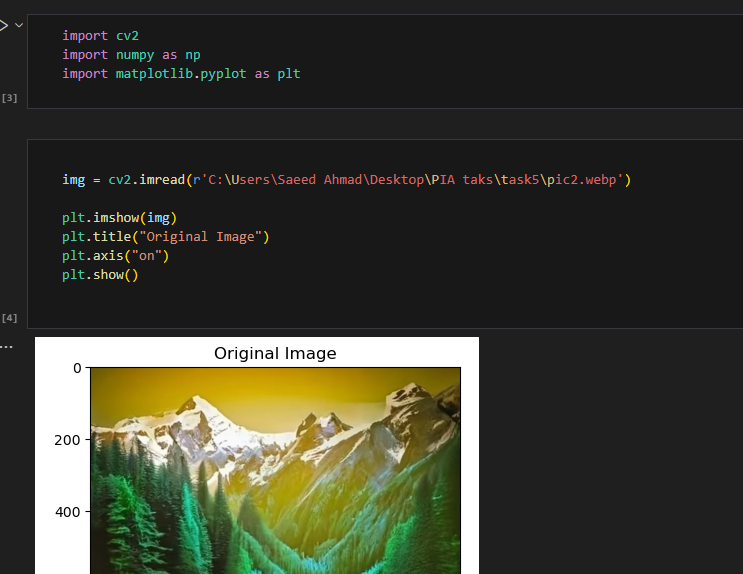
[ **Canny Edge Detection** was applied using cv2.Canny(image, 100, 200), highlighting sharp edges in the image.  **6. Contour Detection** 6](#_Toc192280858)

[**7. Conclusion** 8](#_Toc192280859)

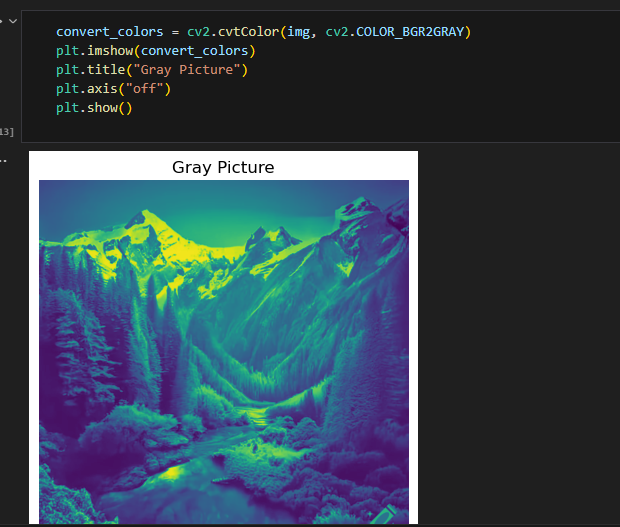
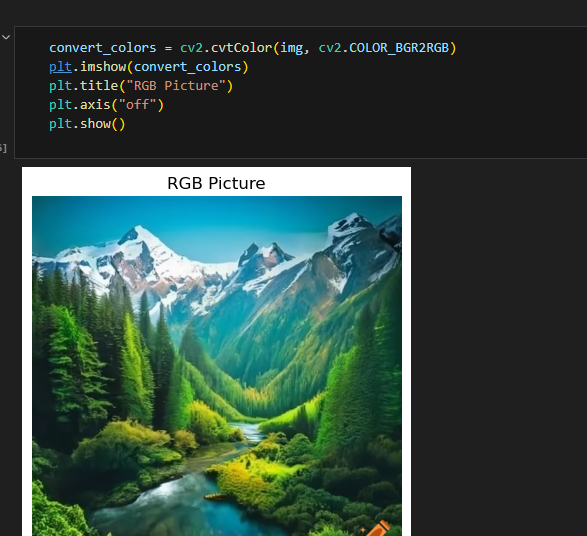
**1. Introduction**

This report provides a detailed analysis of the OpenCV-based image processing performed in the **opencv.ipynb** notebook. The objective is to demonstrate various image processing techniques such as color conversion, resizing, rotation, flipping, edge detection, and contour detection using OpenCV.

**2. Image Loading & Display**

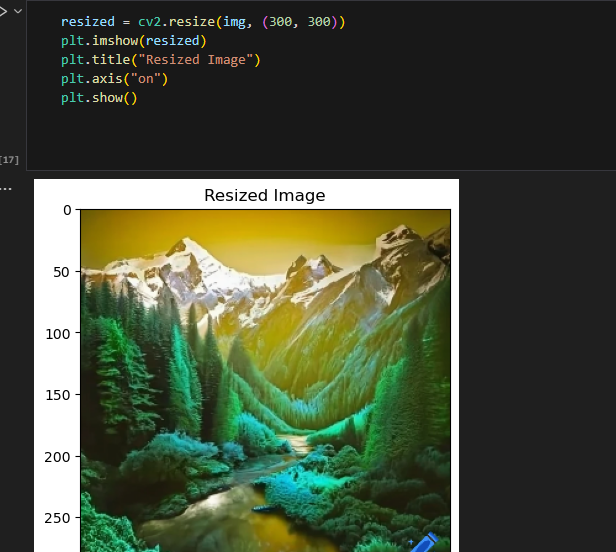
* The image file pic2.webp was loaded using OpenCV’s cv2.imread() function.
* The original image was displayed using matplotlib.pyplot.imshow().  
    
  

**3. Color Conversion**

* The image was converted to **Grayscale** using cv2.COLOR\_BGR2GRAY.  
    
  
* The image was converted to **RGB** format using cv2.COLOR\_BGR2RGB.  
    
  

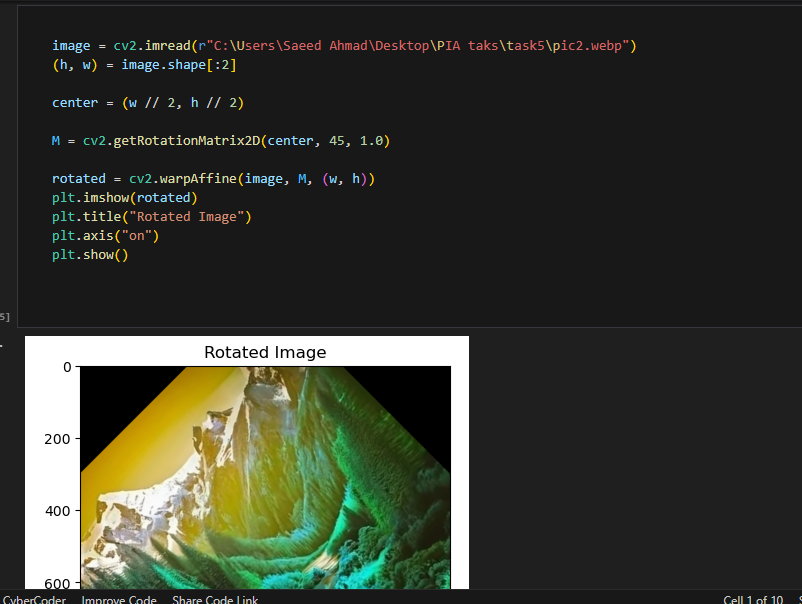
**4. Image Processing**

**4.1 Resizing**

* The image was resized to **300x300 pixels** using cv2.resize(img, (300, 300)).  
    
  

**4.2 Rotation**

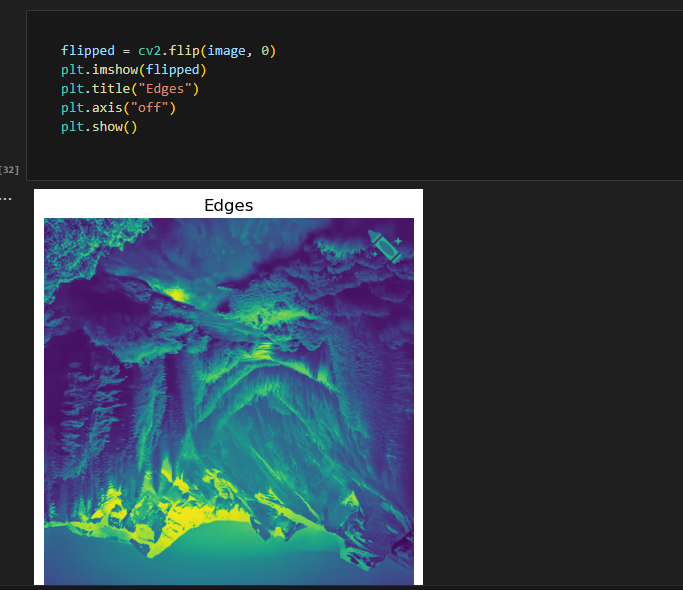
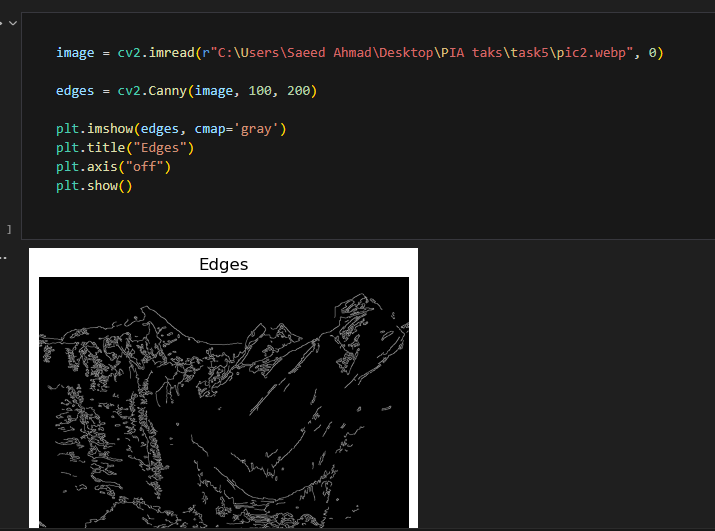
* The image was rotated **45 degrees** using cv2.getRotationMatrix2D() and cv2.warpAffine().



**4.3 Flipping**

* The image was flipped **vertically** using cv2.flip(image, 0).

**5. Edge Detection**

* **Canny Edge Detection** was applied using cv2.Canny(image, 100, 200), highlighting sharp edges in the image.   
    
    
  **6. Contour Detection**
* The image was converted to binary using cv2.threshold().
* Contours were detected using cv2.findContours().
* The detected contours were drawn on an empty image using cv2.drawContours().  
    
  

**7. Conclusion**

This report outlines the complete image processing workflow performed using OpenCV. The steps covered include image loading, color transformations, resizing, rotation, flipping, edge detection, and contour extraction. These techniques are essential for computer vision applications such as object detection, image segmentation, and feature extraction.

Future improvements can include:

* Applying **Gaussian Blur** before edge detection for noise reduction.
* Using **Adaptive Thresholding** for better contour detection.
* Implementing **Morphological Operations** for noise removal in binary images.