Image Processing Toolkit – Report

Name: Goleti Santhosh
Roll No / College ID: 23675A7309
Course: B.Tech AIML
Subject: Computer Vision
Project: GUI-based Image Processing Application using Streamlit

1. Introduction

This project demonstrates fundamental image processing techniques using Python's OpenCV library within an interactive GUI designed using Streamlit. The application allows users to upload images, apply transformations, filters, edge detection, and enhancements, and visualize results side by side. The main objective is to bridge theory and practice by allowing hands-on manipulation of image data with real-time feedback.

2. Objective

The primary objective of this project is to design and implement a GUI-based application that:

- Allows users to upload images and perform various image processing operations.
- Provides real-time visualization and interaction.
- Covers key concepts like color conversions, transformations, filtering, morphology, enhancement, edge detection, and compression.
- Helps users understand the effects of different operations on images.

3. Theory & Background

CMOS vs CCD

- **CMOS (Complementary Metal-Oxide Semiconductor):** Consumes less power, cheaper, and widely used in modern devices.
- **CCD (Charge-Coupled Device):** Provides higher-quality images but consumes more power and is costlier.
- CMOS is more suitable for consumer applications like mobile cameras, while CCD is used in scientific imaging.

Sampling & Quantization

- **Sampling:** Converts continuous images into discrete pixels by measuring intensity at regular intervals.
- **Quantization:** Assigns numerical values to pixel intensity levels, limiting the range but allowing digital storage and processing.

PSF (Point Spread Function)

PSF describes how a single point of light spreads in the imaging system due to diffraction and imperfections. It is essential in understanding blur and how image restoration techniques work.

4. Implementation Details

Tools Used

- **Python** for scripting
- **OpenCV** for image processing functions
- **NumPy** for handling matrix operations
- **Streamlit** for GUI development

Workflow

1. Upload an image.
2. Display the original image and its properties.
3. Select operations from categories: color conversion, transformation, filtering, enhancement, morphology, edge detection, and compression.
4. Visualize the processed image alongside the original.
5. Save the processed image with format options.

5. GUI Layout

- **Menu Bar:** Upload, save, and exit options.
- **Sidebar:** Operations categorized into Image Info, Color Conversions, Transformations, Filtering, Edge Detection, Morphology, Enhancement.
- **Display Area:** Two sections showing original and processed images.

- **Status Bar:** Shows image dimensions, channels, format, and file size.

6. Algorithms Explained

1. Color Conversions

- RGB → Grayscale: Combines color channels into one intensity channel.
- RGB → HSV: Converts into hue, saturation, and value components useful for color-based segmentation.

2. Transformations

- **Rotation:** Rotates the image around its center.
- **Scaling:** Enlarges or reduces the image size.
- **Translation:** Moves the image along x and y axes.

3. Filtering

- **Gaussian Filter:** Blurs the image using a normal distribution kernel.
- **Mean Filter:** Averages surrounding pixels.
- **Median Filter:** Reduces salt-and-pepper noise by using the median value.

4. Edge Detection

- **Sobel Filter:** Finds gradients in horizontal and vertical directions.
- **Laplacian Filter:** Highlights areas of rapid intensity change.
- **Canny Edge Detector:** Uses multiple steps including gradient calculation and thresholding.

5. Morphology

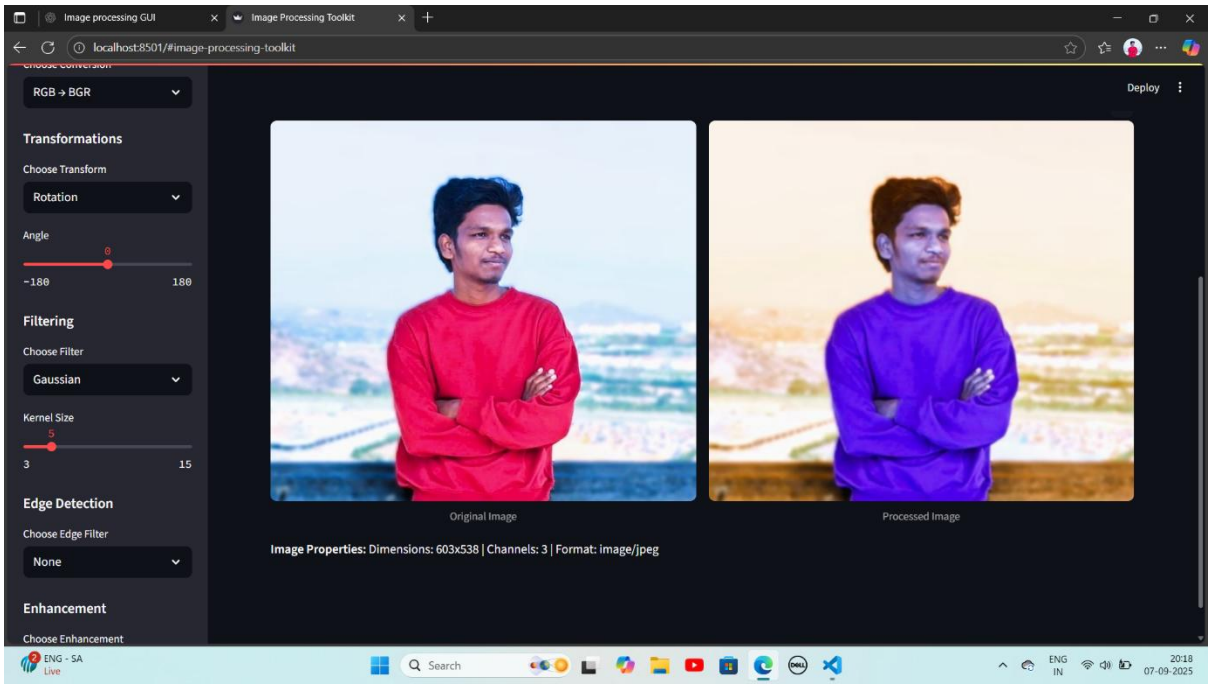
- **Dilation/Erosion:** Expands or shrinks bright regions.
- **Opening/Closing:** Combines erosion and dilation to remove noise or fill gaps.

6. Enhancement

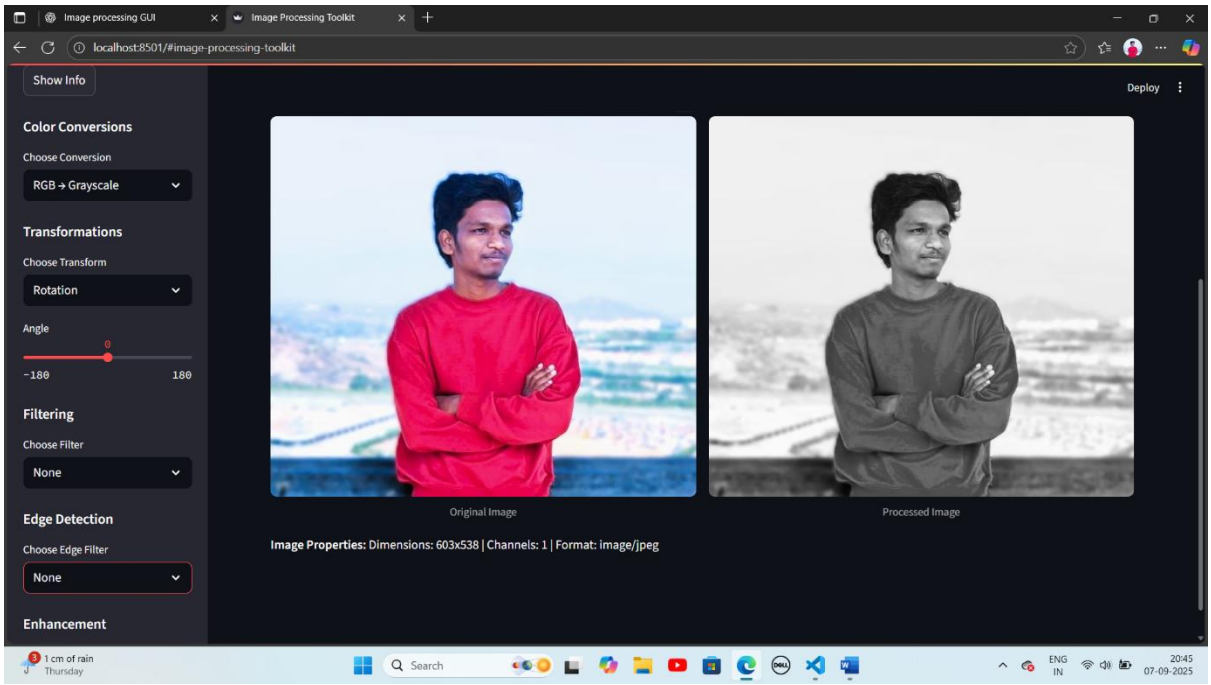
- **Histogram Equalization:** Spreads out pixel intensity distribution for contrast improvement.
- **Contrast Stretching:** Normalizes intensity values between the minimum and maximum.
- **Sharpening:** Enhances edges and details using kernel-based filtering.

7. Results & Observations

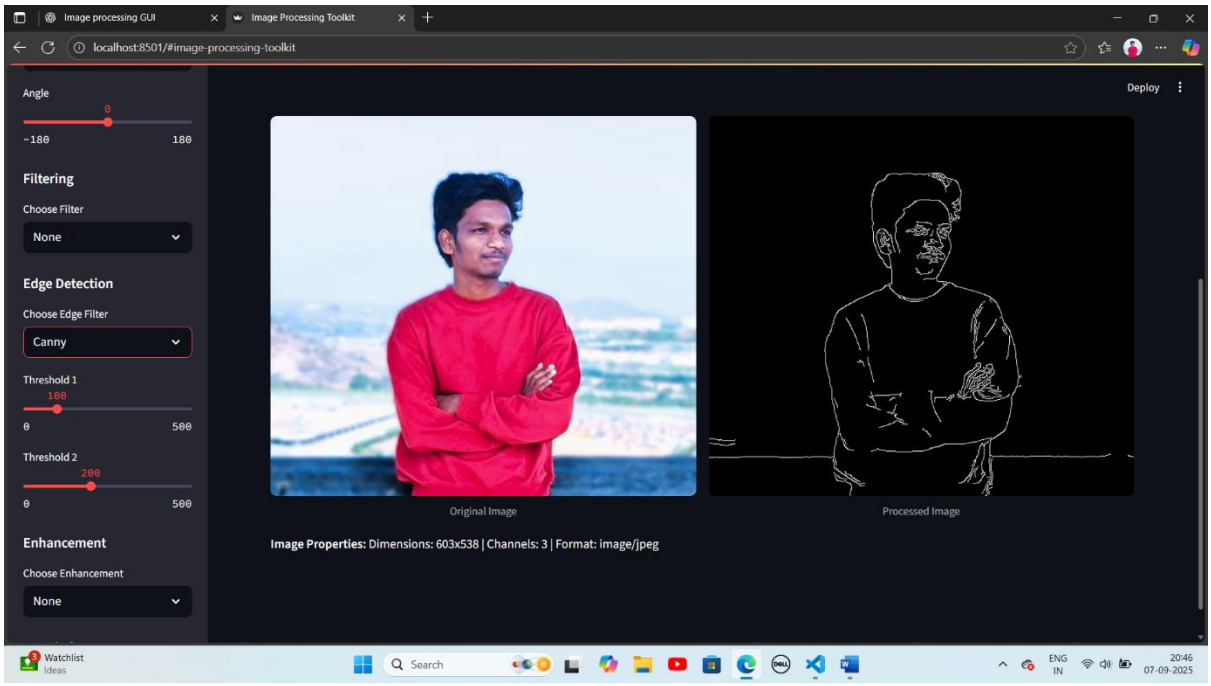
Screenshot 1 – Image Upload



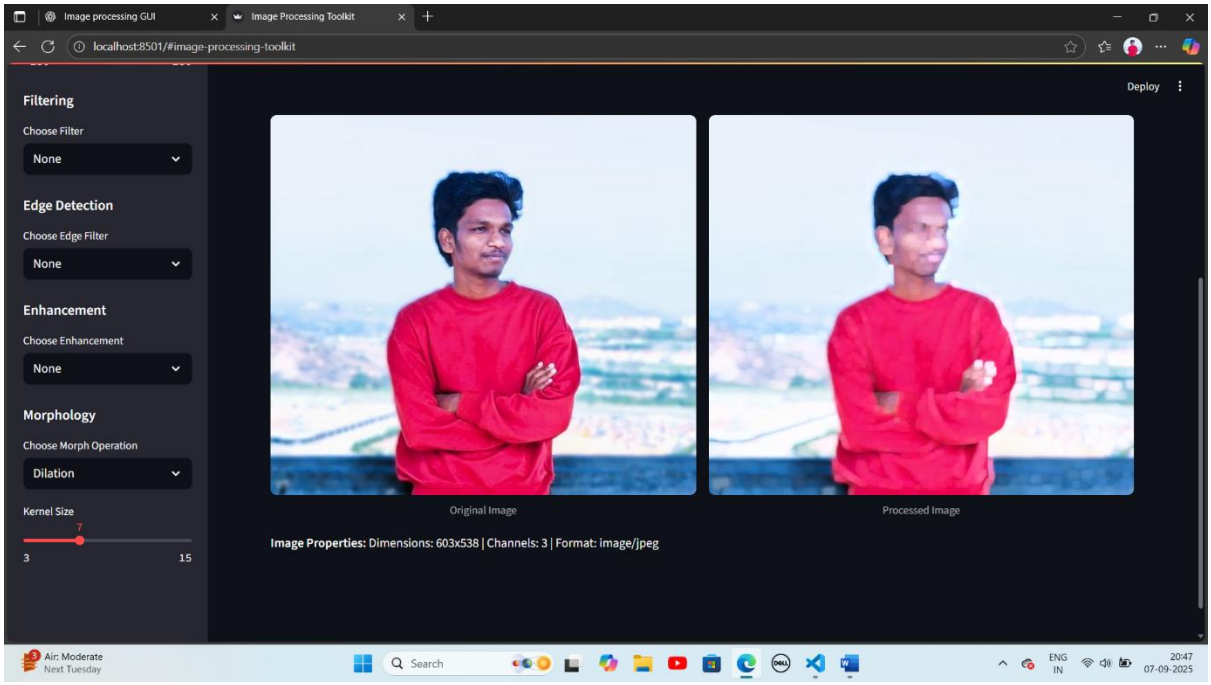
Screenshot 2 – Color Conversion (RGB to Grayscale)



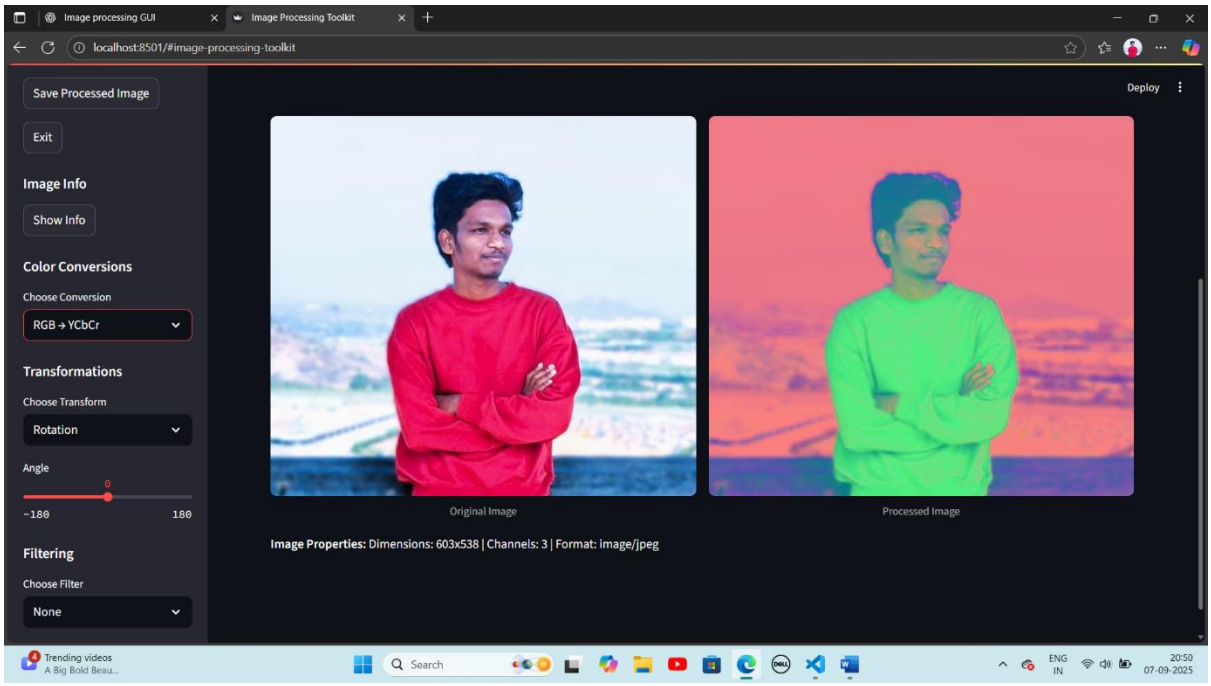
Screenshot 3 – Edge Detection using Canny



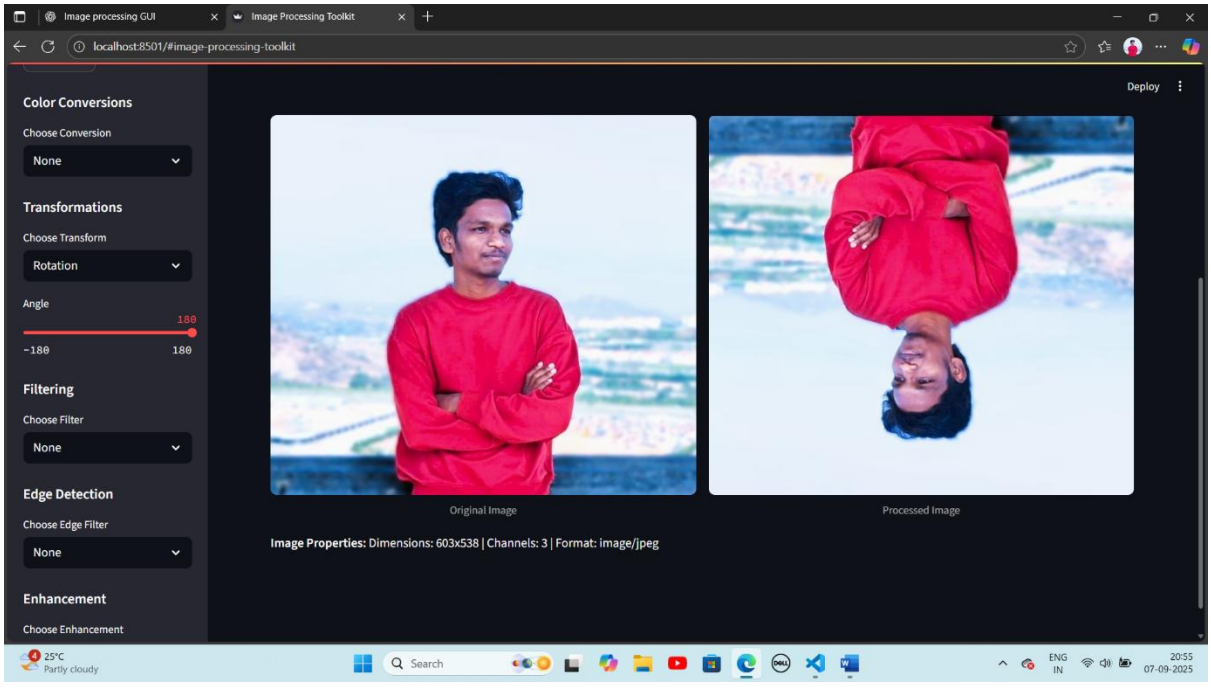
Screenshot 4 – Morphological Operation (Dilation)



Screenshot 5 – Color Conversion (RGB to YCbCr)



Screenshot 6 – Transformations (Rotation)



8. Challenges

- Handling images of different sizes and formats.
- Managing real-time updates without performance lags.
- Providing user-friendly options without overwhelming complexity.

9. Conclusion

The Image Processing Toolkit is a practical solution for learning and experimenting with key image operations. Through a simple interface, users can understand complex mathematical concepts like convolution, filtering, and transformations. This project bridges theoretical knowledge and real-world applications.