Image Processing Toolkit – Report

Submitted by: Om Prakash – 22671A7349
Course: B.Tech – AIML
Subject: Computer Vision
Project: GUI-Based Image Processing Application using OpenCV and Streamlit

1. Introduction

The aim of this project is to design and implement an interactive Image Processing Toolkit using Python libraries such as OpenCV, NumPy, and Streamlit. The application provides a GUI where users can upload images, apply various image processing operations, and visualize the results in real-time.

This toolkit covers not only basic theoretical aspects like filters, transformations, and compression but also offers hands-on experience through interactive GUI elements like sliders, dropdowns, and side-by-side image comparisons.

2. Objective

- To demonstrate fundamental image processing operations using OpenCV.
- To create an interactive GUI using Streamlit for image manipulation.
- To educate users about image properties, formats, and compression methods.
- To provide a platform for experimenting with image enhancement, filtering, and edge detection techniques.
- To compare different operations visually and through file size analysis.

3. Theory

📌 CMOS vs CCD Sensors

- **CCD (Charge-Coupled Device):**
    - High-quality imaging
    - Consumes more power
    - Better for low-light environments
- **CMOS (Complementary Metal Oxide Semiconductor):**
    - Lower cost

- Lower power consumption
- Faster processing
- Used in most modern cameras and smartphones

📌 **Sampling & Quantization**

- **Sampling:** Refers to how frequently the image is sampled in space. Higher sampling provides more detail.
- **Quantization:** Refers to mapping continuous pixel values into discrete levels. Higher quantization levels result in smoother images.

📌 **Point Spread Function (PSF)**

PSF describes how a point source of light is spread out in an image due to the imperfections in the optical system. It's essential for understanding blurring and deblurring techniques.

4. Algorithms Used

Color Conversions

- RGB ↔ BGR, HSV, YCbCr, Grayscale using OpenCV's cv2.cvtColor.

Transformations

- **Rotation:** Image is rotated about its center using affine transformation.
- **Scaling:** Enlarges or shrinks the image while maintaining proportions.
- **Translation:** Moves the image along X and Y axes.
- **Affine Transform:** Preserves parallel lines using matrix transformation.
- **Perspective Transform:** Warps the image to simulate a different viewpoint.

Filtering & Morphology

- **Gaussian Filter:** Blurs image using weighted average.
- **Mean Filter:** Smooths using average intensity.
- **Median Filter:** Removes noise while preserving edges.
- **Dilation & Erosion:** Expands or shrinks regions in an image.
- **Opening & Closing:** Combines dilation and erosion for noise removal and object highlighting.

Enhancement

- **Histogram Equalization:** Improves contrast by redistributing intensities.
- **Contrast Stretching:** Expands dynamic range of intensities.
- **Sharpening:** Enhances edges using convolution.

Edge Detection

- **Sobel & Laplacian:** Detects intensity gradients.
- **Canny:** Advanced edge detection using thresholds and non-maximum suppression.

Compression

- Converts images to formats like JPG, PNG, BMP and compares their sizes and qualities.

5. Implementation Details

Tools and Libraries

- **Python 3.x**
- **OpenCV:** For image manipulation and processing.
- **NumPy:** For efficient numerical computations.
- **Streamlit:** For creating the GUI and interactive widgets.
- **PIL (Pillow):** For handling image formats.

GUI Layout

1. **Sidebar:**
    - Upload image
    - Choose operations (filters, transformations, etc.)
    - Sliders for parameters (kernel size, angle, etc.)
    - Save and download image
2. **Main Display:**
    - Original image on left
    - Processed image on right
3. **Status Bar:**
    - Displays image resolution, format, and file size.

6. Results and Observations

📷 Image Info

After uploading an image, its resolution, channels, format, and size are dynamically displayed.

🟠 Color Conversions

- Grayscale conversion improved visualization of contours.
- HSV and YCbCr transformations were useful for color-based analysis.

🔄 Transformations

- Rotation and scaling preserved image quality.
- Affine and perspective transforms introduced creative warping effects.

🧼 Filtering & Morphology

- Gaussian filter removed noise effectively.
- Median filtering preserved edges while smoothing random disturbances.
- Morphological operations enhanced object visibility and removed artifacts.

⚙️ Enhancement

- Histogram equalization improved contrast in underexposed images.
- Sharpening made edges more prominent.

🔍 Edge Detection

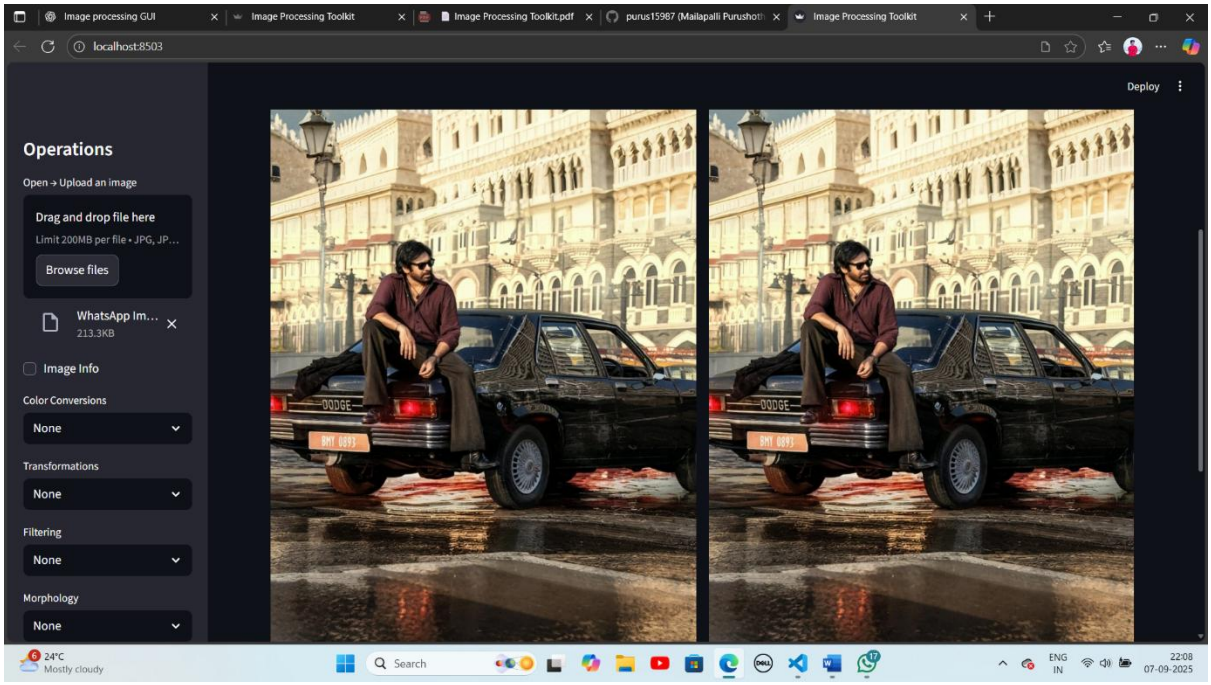
- Sobel and Laplacian highlighted boundaries clearly.
- Canny edge detection gave crisp outlines after adjusting thresholds.

📦 Compression

- JPG compression reduced file size significantly with minimal quality loss.
- PNG preserved quality but resulted in larger file sizes.

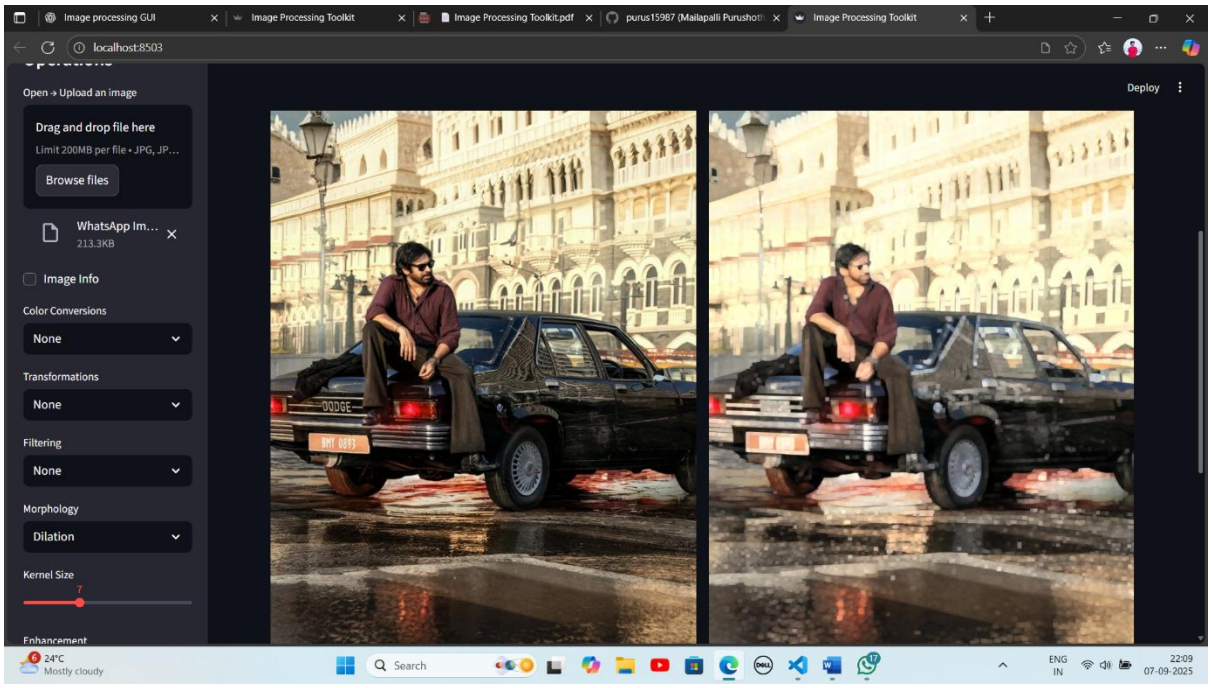
7. Screenshots

1.Upload photo:

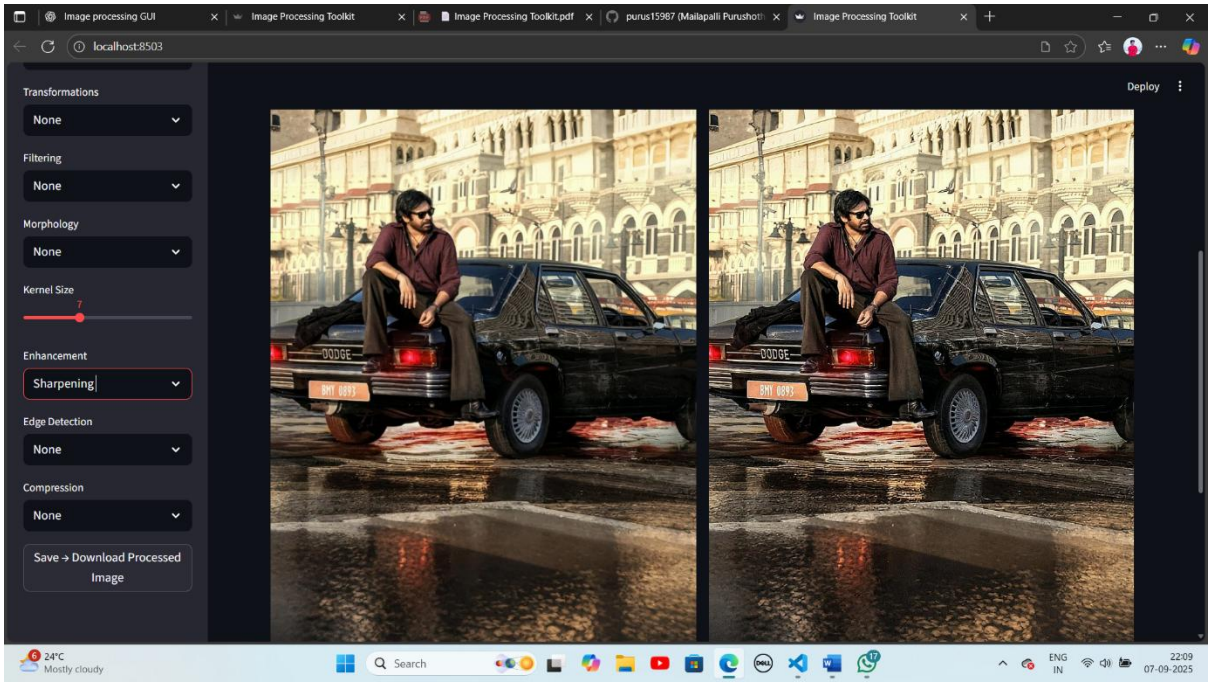


2.Color Conversion (RGB – BGR)

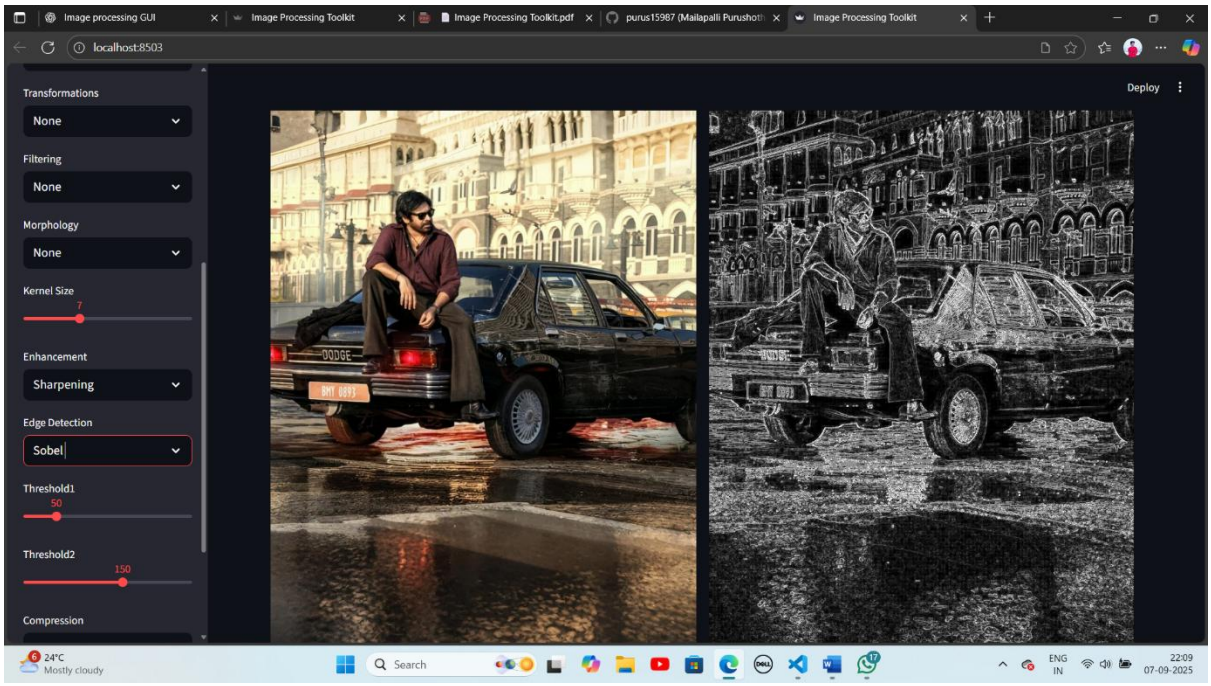
3.Morphology (Dilation)



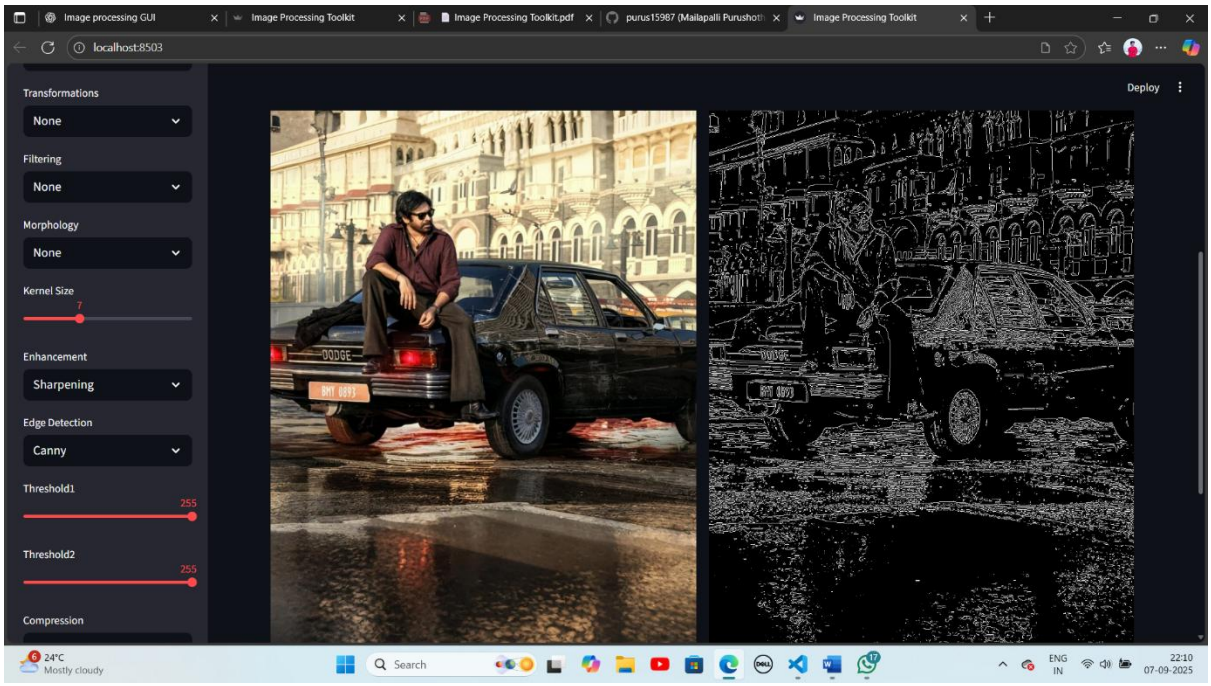
4.Enhancement (Sharpening)



5.Edge detection (Sobel)



6.Edge detection (Canny)



8. Conclusion

This Image Processing Toolkit provides an excellent platform for both beginners and advanced users to understand and experiment with image processing algorithms. The integration of OpenCV with Streamlit makes it accessible, interactive, and visually engaging.

It covers essential techniques like filtering, edge detection, transformations, and image enhancement, while also educating users on image formats and compression methods. The toolkit is scalable and can be extended with additional functionalities like video processing or deep learning-based enhancements.