Image Processing Toolkit – Project Report

Submitted by: S.Govardhan – 22671A7347
Course: B.Tech – AIML
Subject: Computer Vision
Project Title: GUI-Based Image Processing Application Using OpenCV and Streamlit

1. Introduction

Image processing is a fundamental part of computer vision and artificial intelligence applications. The purpose of this project is to create an interactive and educational toolkit where users can perform various image processing operations such as filtering, transformations, edge detection, and compression using a graphical user interface (GUI).

The toolkit is designed with Python's powerful libraries, **OpenCV**, **NumPy**, and **Streamlit**, to provide real-time visual feedback and a hands-on experience for users. It covers both theoretical aspects and practical implementations, making it suitable for beginners and advanced learners alike.

2. Objectives

- To build a GUI-based application that allows users to upload images and apply processing operations interactively.
- To demonstrate important image operations including color conversions, geometric transformations, filtering, enhancement, and compression.
- To provide insights into image properties such as resolution, format, file size, and channels.
- To visualize the effect of each operation in real-time and compare results side by side.
- To encourage experimentation with adjustable parameters like kernel size, thresholds, rotation angle, and scaling.

3. Theory Overview

📌 **CMOS vs CCD Sensors**

- **CCD (Charge-Coupled Device):** Offers better image quality and sensitivity in low-light conditions but consumes more power and is expensive.
- **CMOS (Complementary Metal-Oxide-Semiconductor):** Cheaper, more energy-efficient, and faster processing. Common in smartphones and modern cameras.

📌 **Sampling and Quantization**

- **Sampling:** Converts continuous images into discrete pixel values by selecting specific points at regular intervals.
- **Quantization:** Approximates continuous intensity values to a finite number of levels, affecting the image's appearance and smoothness.

📌 **Point Spread Function (PSF)**

The PSF models how an imaging system responds to a point source of light. It explains how light spreads, causing blur in images. Understanding PSF is crucial for image restoration and sharpening algorithms.

4. Implementation Details

Technologies Used

- **Python 3.x** – Core programming language.
- **OpenCV** – For image manipulation and processing algorithms.
- **NumPy** – For numerical operations and array manipulation.
- **Streamlit** – For building the interactive web GUI.
- **PIL (Pillow)** – For image format conversions and downloads.

GUI Design

- **Sidebar Panel:**
  - Upload and download images.
  - Choose operations like filters, transformations, and edge detection.
  - Adjust parameters with sliders (kernel size, angle, thresholds).
  - View image properties dynamically.
- **Main Display Area:**
  - Displays the original and processed images side by side.
- **Status Bar:**
  - Shows image dimensions, format, and file size.

5. Operations Implemented

📌 **Image Information**

- Resolution, color channels, format, and file size are shown for each uploaded image.

📌 **Color Conversions**

- RGB ↔ BGR
- RGB ↔ HSV
- RGB ↔ YCbCr
- RGB ↔ Grayscale

📌 **Geometric Transformations**

- Rotation, Scaling, Translation
- Affine and Perspective transformations for creative manipulations.

📌 **Filtering**

- Gaussian Blur, Mean Blur, Median Blur for noise removal.

📌 **Morphological Operations**

- Dilation, Erosion, Opening, Closing for enhancing or cleaning image structures.

📌 **Enhancement Techniques**

- Histogram Equalization to improve contrast.
- Contrast Stretching for intensity normalization.
- Sharpening using convolution kernels.

📌 **Edge Detection**

- Sobel, Laplacian, and Canny algorithms for detecting edges and contours.

📌 **Compression**

- Images can be saved in JPG, PNG, or BMP formats with real-time file size comparison.

6. Observations and Results

📷 **Image Info**

The application successfully displays important image metadata such as resolution, file format, and size, aiding users in understanding how operations affect images.

🎨 **Color Conversions**

The toolkit effectively transforms images between different color spaces, enabling color-based filtering and adjustments.

🔄 **Transformations**

- Rotation and scaling preserve image clarity.

- Affine and perspective transformations create interesting visual distortions while maintaining image structure.

🧼 Filtering and Morphology

- Gaussian and median filters remove noise while preserving edges.
- Dilation and erosion enhance or suppress certain image features effectively.

📈 Enhancement

- Histogram equalization improved contrast in dark images.
- Contrast stretching normalized intensity distribution.
- Sharpening enhanced the appearance of edges.

🔍 Edge Detection

- Sobel and Laplacian provided gradient-based edge extraction.
- Canny edge detection produced clear, noise-reduced edges with adjustable thresholds.

📦 Compression

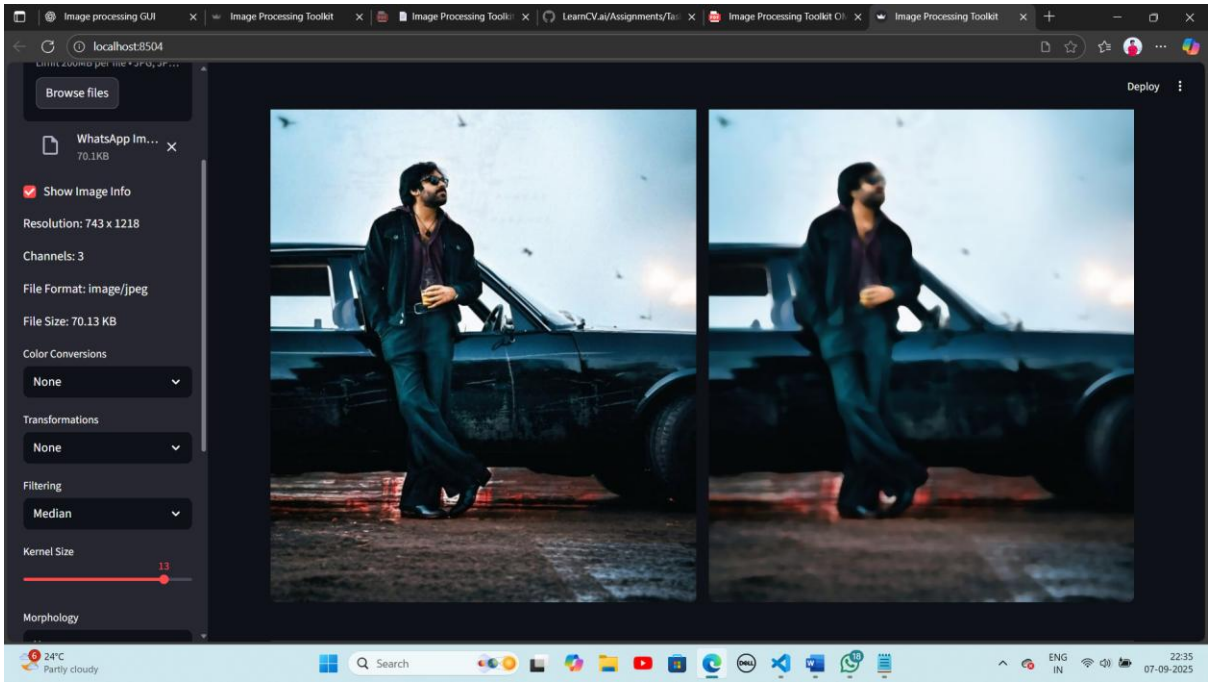
- JPG offered the best compression rate with acceptable quality.
- PNG maintained higher fidelity at larger sizes.

7. Screenshots

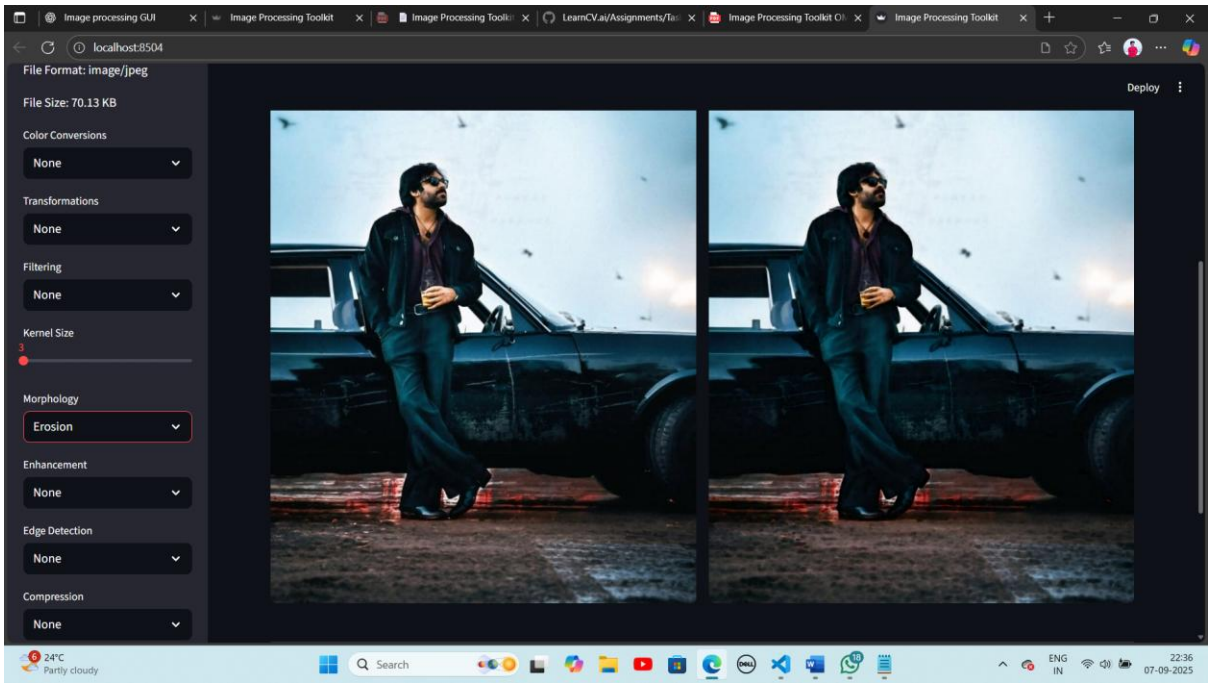
1. Upload image

2. Color conversion (RGB to BGR)

3.Filtering median (kernel size 13)

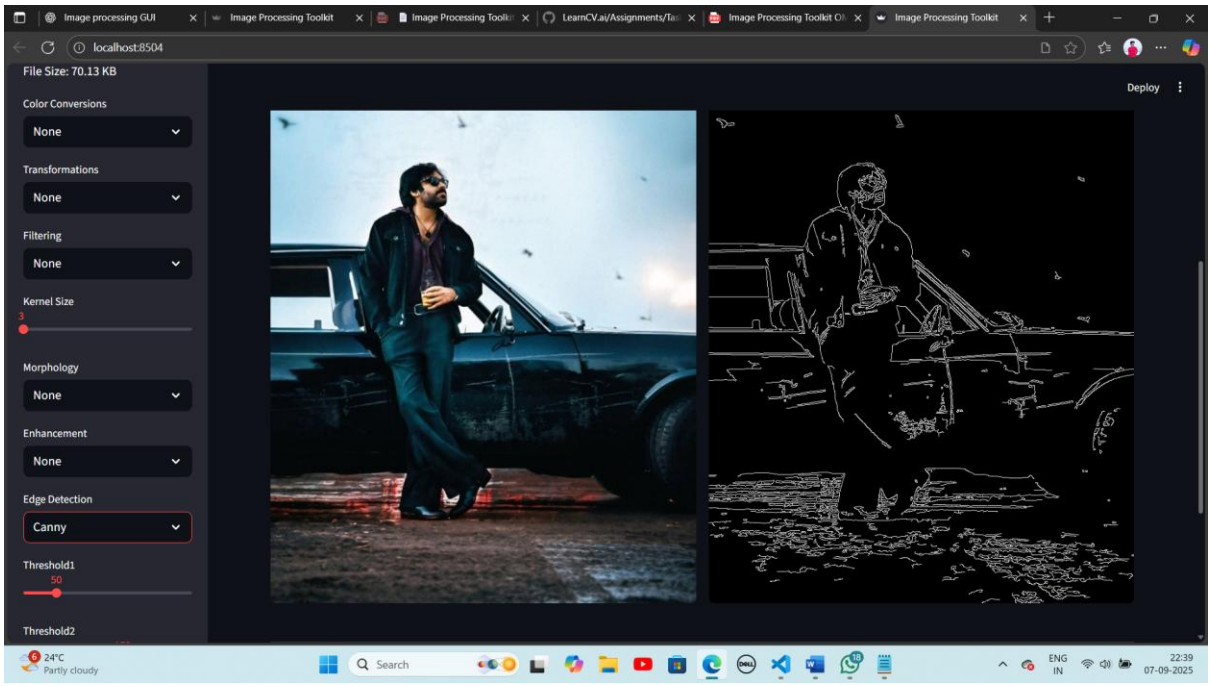


4.Morphology (erosion)



5.Enhancement (Histogram equilization)

6.Edge detection (Canny)



8. Conclusion

The Image Processing Toolkit successfully demonstrates essential image processing techniques while providing an intuitive interface for users to experiment and learn. Through interactive widgets and real-time feedback, users gain hands-on experience with image manipulation concepts, making this project a valuable educational tool.

It integrates theory with practice by allowing users to observe how each operation affects image properties such as contrast, sharpness, and file size. The modular structure of the code makes it easy to extend with future functionalities like video processing or advanced AI-based methods.