Comprehensive Report on OpenCV Image Processor GUI Application

Prepared for Academic/Professional Submission

September 7, 2025

Contents

Chapter 1

Introduction

This report provides an in-depth explanation of the GUI-based Image Processing application developed using Python, Streamlit, OpenCV, NumPy, and PIL. The project aims to demonstrate fundamental image processing operations interactively through a user-friendly graphical interface. Users can upload images, apply multiple operations, visualize outputs, and even experiment in real-time using a webcam feed.

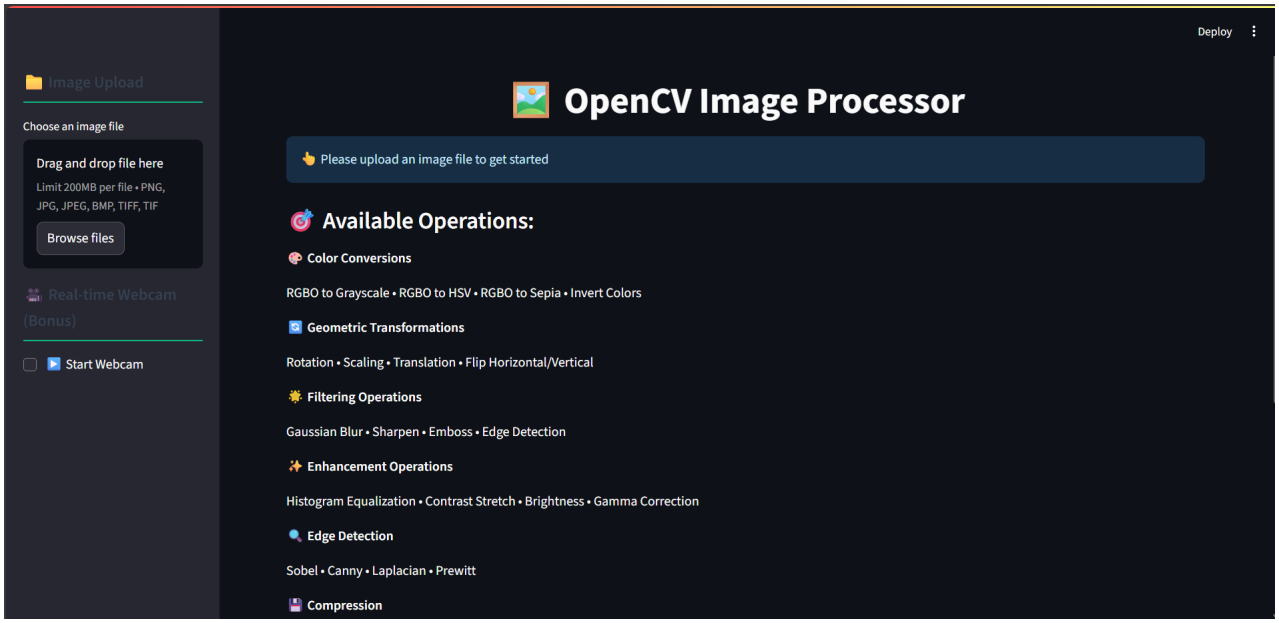

Fig 1 : Home

Chapter 2

System Design and Objectives

The main objectives of this project are:

- Provide a GUI-based platform to experiment with basic and advanced image processing concepts.
- Implement core operations like color conversions, transformations, filtering, enhancement, and edge detection.
- Support real-time webcam processing.
- Enable image export in different formats with compression options.

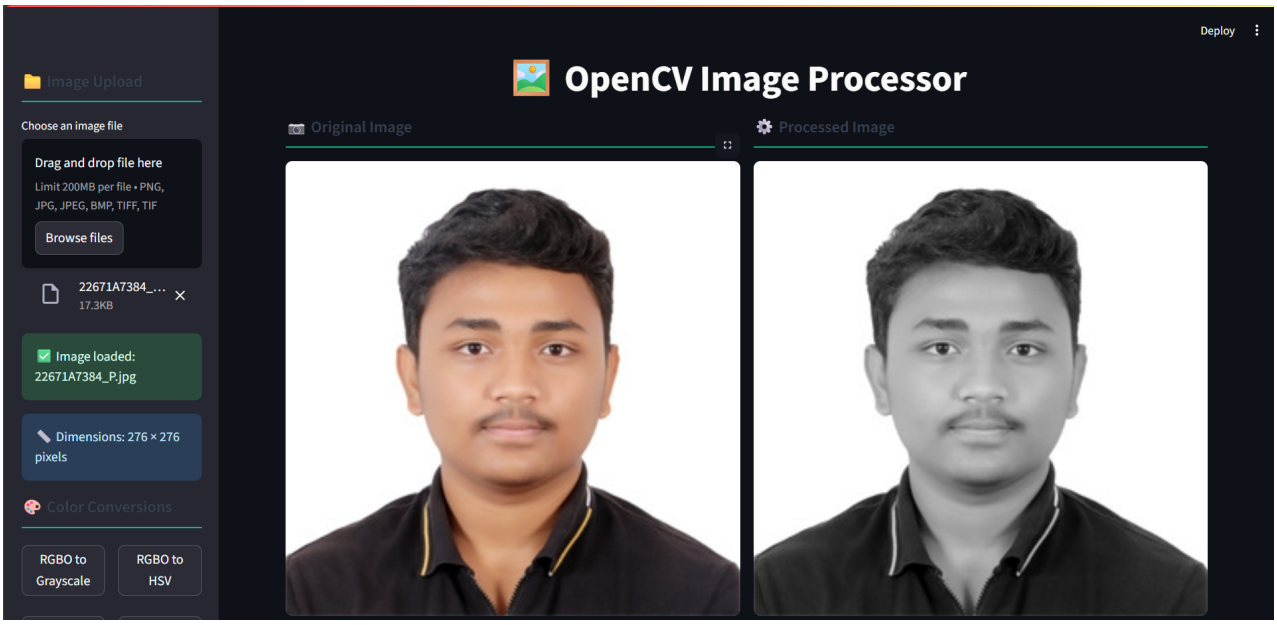

Fig 2 ; Image insertion - Grayscale

Chapter 3

Technologies Used

Python: The programming language used for development.

Streamlit: Provides the GUI framework for building interactive web applications with minimal code.

OpenCV: The core computer vision library handling all image operations.

NumPy: Used for array manipulation and image matrix processing.

PIL (Pillow): Supports loading, saving, and converting images.

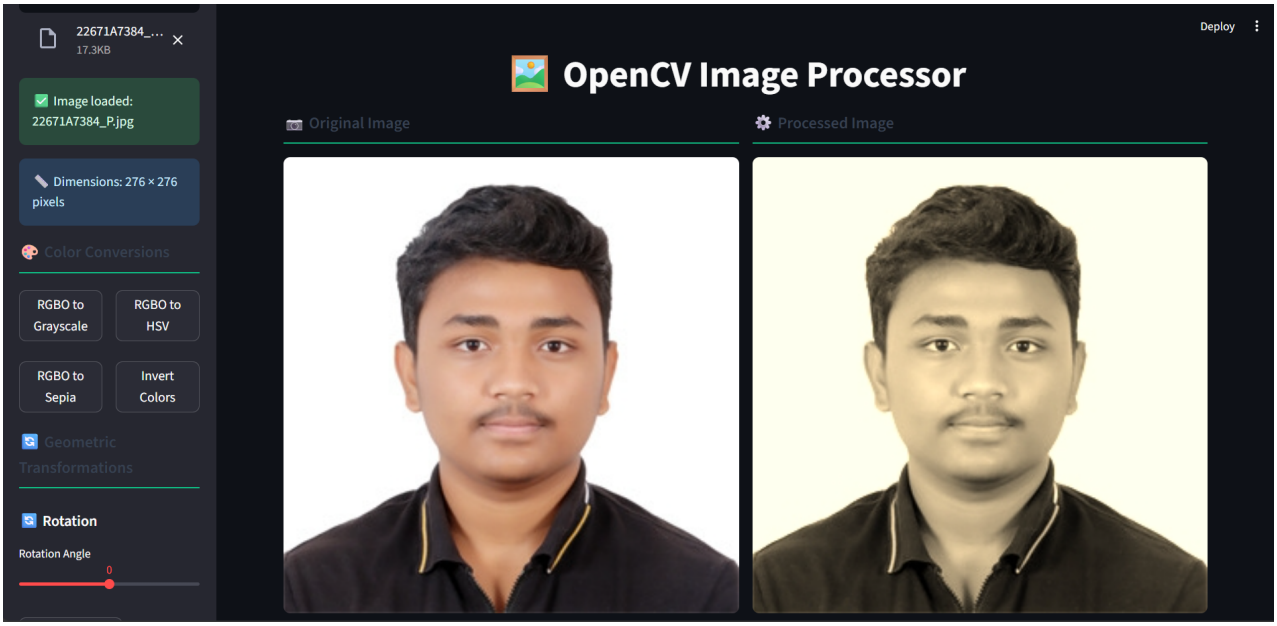

Fig 3 : Imge insertion : Sepia

Chapter 4

Application Architecture

The application is divided into the following layers:

1. **Frontend (Streamlit GUI):** Manages user interaction through sidebars, sliders, and buttons.
2. **Backend (ImageProcessor Class):** Encapsulates all image operations (color conversion, filtering, transformations, etc.).
3. **Session State:** Stores current images and operations to maintain state across user interactions.
4. **Display:** Shows original vs processed image side-by-side along with status metrics.

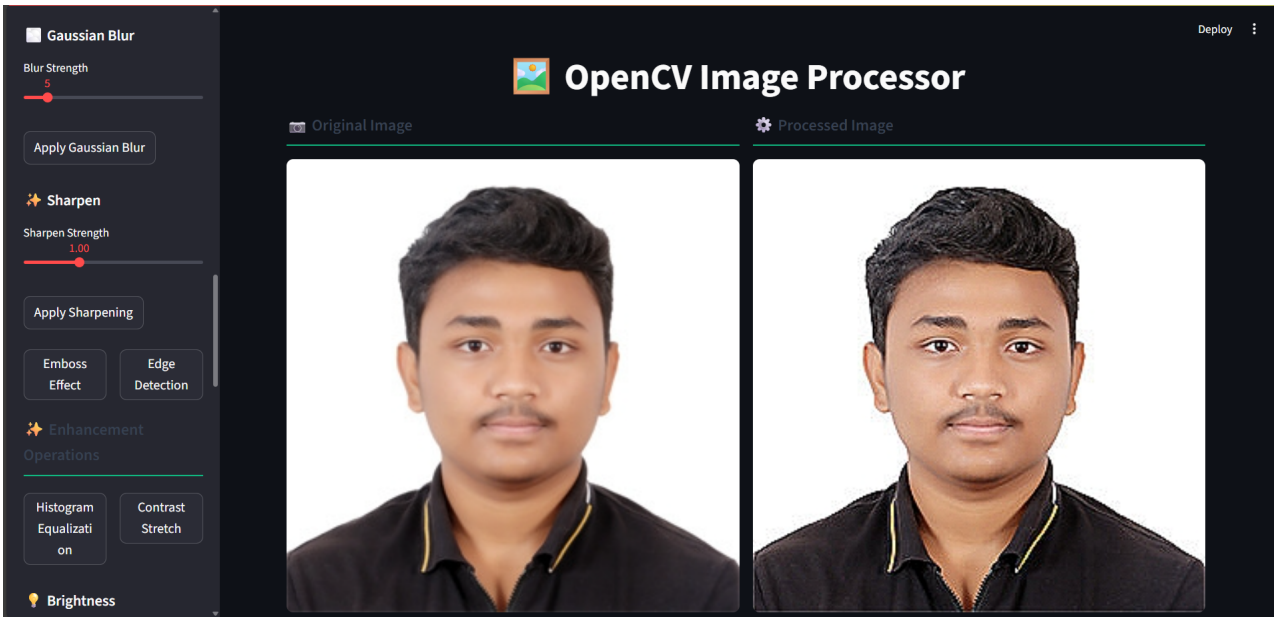

Fig 4 : Image sharpanening

Chapter 5

Code Walkthrough

5.1 Imports and Configuration

The application begins by importing required libraries:

```
import streamlit as st
import cv2
import numpy as np
from PIL import Image
import io
import time
```

Streamlit page configuration is set for title, icon, and layout.

5.2 Styling

Custom CSS is embedded to improve GUI aesthetics such as headers, buttons, and containers.

5.3 ImageProcessor Class

This class encapsulates all core functionalities.

5.3.1 Loading and Displaying Images

- `load_image()`: Converts uploaded PIL image into OpenCV format.
- `convert_for_display()`: Converts OpenCV's BGR format into RGB for Streamlit.

5.3.2 Color Conversions

- Grayscale
- HSV
- Sepia
- Invert Colors

5.3.3 Geometric Transformations

- Rotation
- Scaling
- Translation
- Flipping (Horizontal, Vertical, Both)

5.3.4 Filtering Operations

- Gaussian Blur
- Sharpening
- Emboss
- Basic Edge Detection

5.3.5 Enhancement Operations

- Histogram Equalization
- Contrast Stretching
- Brightness Adjustment
- Gamma Correction

5.3.6 Edge Detection

- Sobel
- Canny
- Laplacian
- Prewitt

5.4 Main Function

The `main()` function coordinates GUI flow:

1. Image Upload and Info Display
2. Operation Selection via Sidebar
3. Display Original vs Processed Image
4. Status Bar with metrics (operation, dimensions, format, size)
5. Webcam Support for real-time processing
6. Image Saving in PNG, JPG, BMP

Chapter 6

Features

- Intuitive GUI with clear categories for operations.
- Real-time feedback when parameters (e.g., blur kernel, gamma) are adjusted.
- Download option with compression awareness.
- Bonus webcam mode allows applying operations live.

Chapter 7

Future Enhancements

- Implement morphological operations (dilation, erosion, opening, closing).
- Add affine and perspective transformations.
- Introduce batch image processing.
- Integrate advanced models (e.g., face detection with deep learning).

Chapter 8

Conclusion

This project successfully demonstrates fundamental image processing operations using an interactive GUI. It bridges the gap between theoretical understanding and practical experimentation. The modular design makes it extensible for advanced features, making it an excellent learning and teaching toolkit.