Report – A Practical Application with a Graphical User Interface (GUI)

Name: [A. Sai Snehitha]
Roll no: 22671A7365

Abstract

This report presents the design and implementation of a **Computer Vision Toolkit** developed using **OpenCV, NumPy, Pillow, and Streamlit**. The toolkit provides a **graphical user interface (GUI)** for performing essential image processing operations, including color conversions, geometric transformations, filtering, morphological operations, enhancement, and edge detection.

The system is implemented as a Streamlit-based web application that allows users to upload images, apply operations interactively, visualize results in real time, and export processed outputs in multiple formats. Each feature is explained with its underlying functionality and relevance.

This project demonstrates how modern GUI frameworks can be combined with core computer vision libraries to create an **educational, practical, and extensible platform** for digital image analysis and multimedia applications.

---

1. Introduction

The aim of this project is to design a **user-friendly computer vision toolkit** that simplifies image processing tasks through an interactive GUI. Traditionally, image processing requires coding expertise, which can be a barrier for beginners. By integrating **Streamlit** with **OpenCV**, the toolkit enables intuitive exploration of computer vision concepts without programming.

This report details the toolkit's system architecture, core operations, and user interface, emphasizing **modularity, usability, and reusability**. It highlights how the toolkit serves both **educational purposes** and **prototype testing** for image-based applications.

---

2. System Overview

The toolkit consists of two core components:

1. **Backend (OpenCV + NumPy)** – Implements image processing operations such as filtering, transformations, and edge detection.
2. **Frontend (Streamlit GUI)** – Provides an interactive platform for uploading images, selecting operations, adjusting parameters, and visualizing results.

The modular design ensures that new features can be added with minimal changes, making the system both **flexible and extensible**.

---

3. Utility and Helper Functions

3.1 Image Loading and Metadata Extraction

- **load_image**: Converts uploaded images into NumPy arrays for processing.
- **show_info**: Displays metadata such as resolution, format, number of channels, and file size.

3.2 Image Saving and Downloading

- **save_image**: Saves processed images in PNG, JPG, or BMP formats.
- **download_link**: Generates a Base64-encoded link for downloading processed files.

These utilities enhance the usability of the toolkit by ensuring seamless data handling and export.

---

4. Core Operations

4.1 Color Conversions

The toolkit supports:

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

These conversions are crucial for pre-processing tasks in computer vision and machine learning pipelines.

4.2 Geometric Transformations

- **Rotate** – Rotates an image by a specified angle.
- **Scale** – Resizes the image proportionally.
- **Translate** – Shifts the image along X and Y axes.
- **Affine Transformation** – Applies linear transformations using control points.
- **Perspective Transformation** – Warps the image to simulate different viewpoints.

These operations illustrate spatial manipulation of images, essential in tasks such as image alignment and viewpoint correction.

4.3 Filtering and Morphology

- **Filters**: Gaussian, Mean, and Median filters for smoothing and noise reduction.
- **Morphology**: Dilation, Erosion, Opening, and Closing for shape-based processing.

4.4 Enhancement

- **Histogram Equalization** – Improves global contrast.
- **Contrast Stretching** – Normalizes intensity ranges.
- **Sharpening** – Enhances details using convolution kernels.

4.5 Edge Detection

- **Sobel Operator** – Highlights gradient changes.

- **Laplacian Operator** – Captures rapid intensity variations.
- **Canny Detector** – A robust multi-stage edge detection technique.

---

5. Streamlit Application Structure

The application is organized into **tabs**, each dedicated to a specific set of operations:

1. **Info** – Displays metadata and the original image.
2. **Colors** – Provides interactive color space conversions.
3. **Transform** – Supports geometric transformations with parameter sliders.
4. **Filters** – Includes smoothing and morphological tools.
5. **Enhance** – Provides contrast, equalization, and sharpening.
6. **Edges** – Performs edge detection with adjustable thresholds.
7. **Save** – Allows exporting processed images.

The **sidebar** provides file upload and parameter controls, ensuring smooth user interaction.

---

6. Detailed Operations by Category

Each tab is implemented with interactive widgets such as **sliders, radio buttons, and dropdowns**, enabling users to explore and visualize image processing techniques dynamically. This design makes the toolkit ideal for both **self-learning** and **classroom demonstrations**.

---

7. Save and Download Feature

The **Save tab** enables users to export results in multiple formats (PNG, JPG, BMP). It also displays the compressed file size, helping users understand the trade-off between quality and storage.

---

8. Conclusion

The **Computer Vision Toolkit** integrates **OpenCV** with **Streamlit** to provide an interactive and modular GUI for image processing. It enables users to perform a wide range of operations—color conversions, transformations, filtering, enhancement, and edge detection—without writing code.

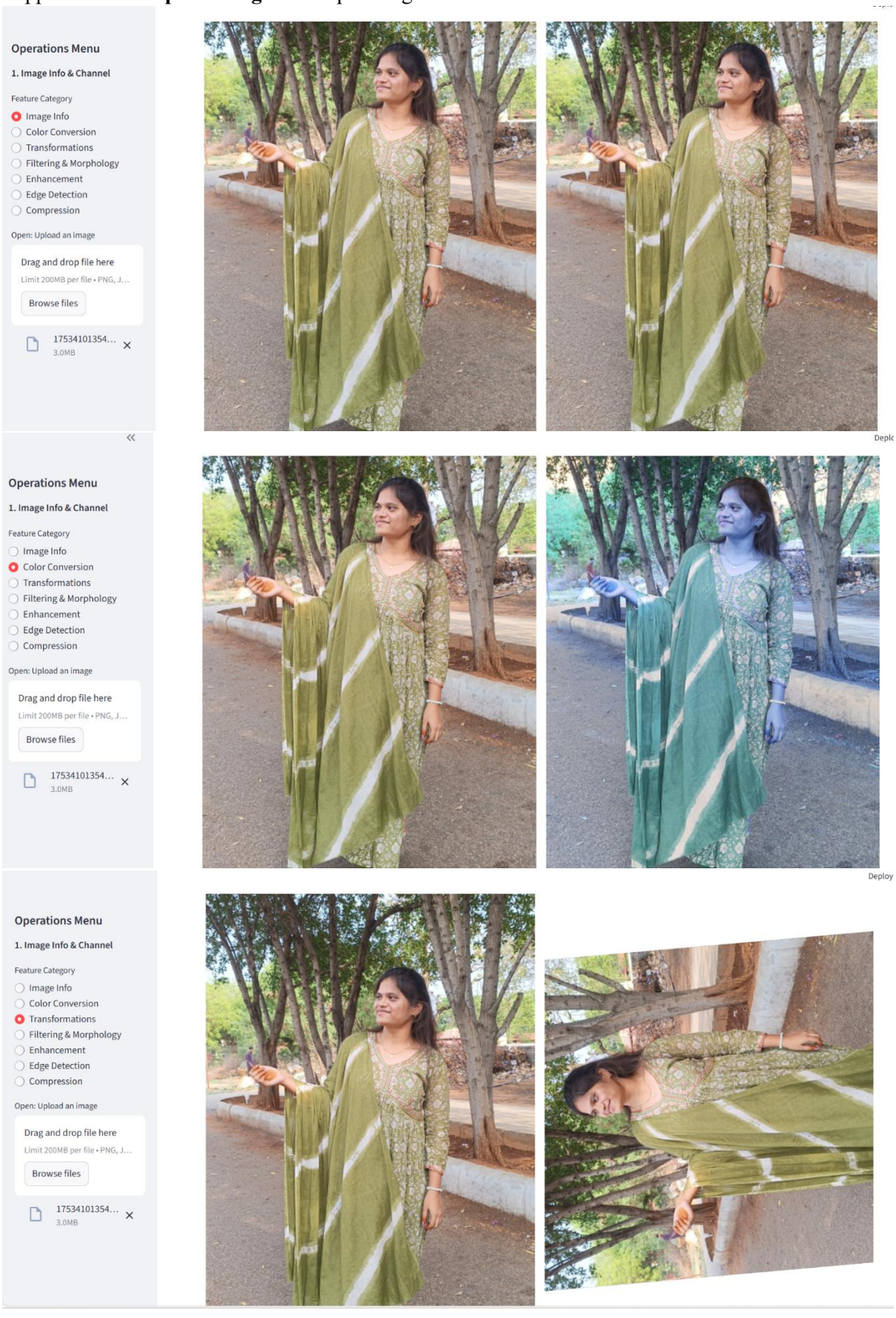
This toolkit is valuable as:

- An **educational platform** for teaching image processing concepts.
- A **testing environment** for rapid prototyping in computer vision.
- A **practical tool** for multimedia and graphics research.

Future enhancements may include:

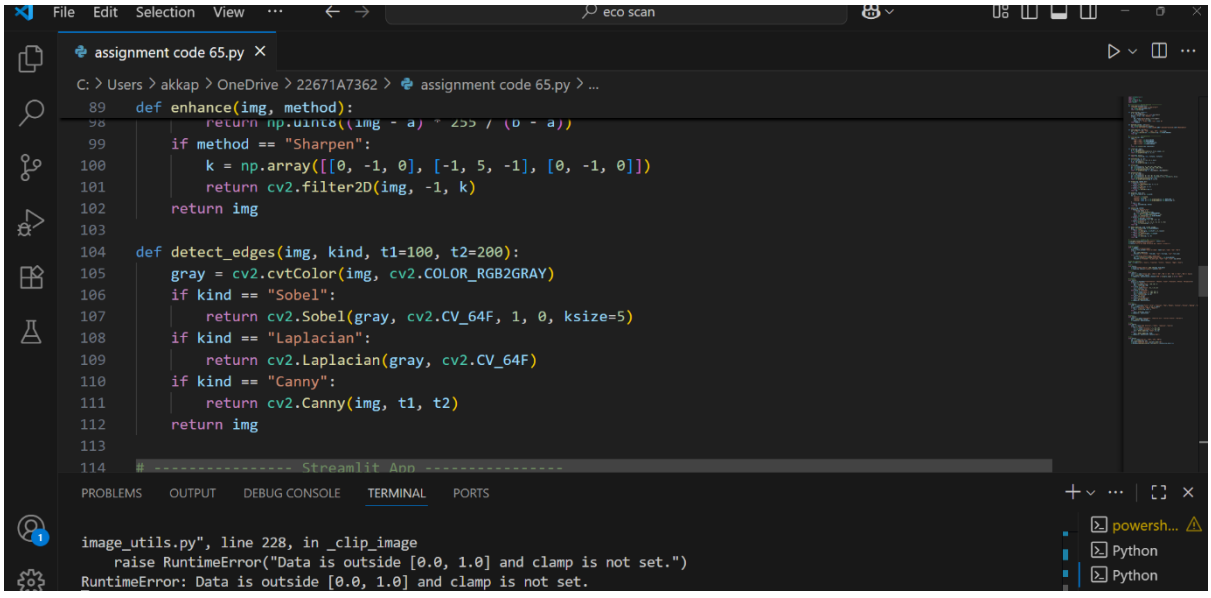
- Real-time **video processing**.

- Integration of **deep learning models** for object detection and recognition.
- Support for **batch processing** of multiple images.




Operations Menu
1. Image Info & Channel
Feature Category
Image Info
Color Conversion
Transformations
Filtering & Morphology
Enhancement
Edge Detection
Compression
Open: Upload an image
Drag and drop file here
Limit 200MB per file • PNG, J...
Browse files
17534101354...
3.0MB


Operations Menu
1. Image Info & Channel
Feature Category
Image Info
Color Conversion
Transformations
Filtering & Morphology
Enhancement
Edge Detection
Compression
Open: Upload an image
Drag and drop file here
Limit 200MB per file • PNG, J...
Browse files
17534101354...
3.0MB


File Edit Selection View
eco scan
assignment code 65.py
C: > Users > akkap > OneDrive > 22671A7362 > assignment code 65.py > ...
def enhance(img, method):
return np.uint8((img - a) * 255 / (b - a))
if method == "Sharpen":
k = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
return cv2.filter2D(img, -1, k)
return img
def detect_edges(img, kind, t1=100, t2=200):
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
if kind == "Sobel":
return cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=5)
if kind == "Laplacian":
return cv2.Laplacian(gray, cv2.CV_64F)
if kind == "Canny":
return cv2.Canny(img, t1, t2)
return img
---------------- Streamlit App ----------------
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
image_utils.py", line 228, in _clip_image
raise RuntimeError("Data is outside [0.0, 1.0] and clamp is not set.")
RuntimeError: Data is outside [0.0, 1.0] and clamp is not set.
powersh...
Python
Python

9. References

- OpenCV Documentation: https://docs.opencv.org/
- Streamlit Documentation: https://docs.streamlit.io/
- Pillow Documentation: https://pillow.readthedocs.io/
- NumPy Documentation: https://numpy.org/doc/