📄 Report: Image Processing & Analysis Toolkit

1. Problem Statement

The aim of this project is to design and implement a **GUI-based application** that demonstrates fundamental **image processing operations** in a simple and interactive way. The toolkit enables users to upload an image, apply various transformations, filters, and enhancements, and visualize results instantly.

This eliminates the need for manually writing code, making image processing accessible to students, beginners, and researchers.

---

2. Introduction

Image processing plays a vital role in **computer vision** and **AI-based applications** such as medical imaging, object recognition, and image enhancement.

This project introduces a **Streamlit-based graphical toolkit** integrated with **OpenCV** and **NumPy**. Users can perform real-time operations like:

- Color conversion (RGB ↔ BGR, HSV, YCbCr, Grayscale)
- Geometric transformations (rotation, scaling, translation, affine, perspective)
- Filtering (Gaussian, Median, Mean)
- Morphological operations (erosion, dilation, opening, closing)
- Image enhancement (histogram equalization, contrast stretching, sharpening)
- Edge detection (Sobel, Canny, Laplacian)
- Image compression and saving results

The system displays **Original vs. Processed Images** side by side, making analysis easier.

---

3. Requirements

Software Requirements

- Python 3.10 or above
- Libraries:
    - **OpenCV (cv2)** – image processing
    - **NumPy** – matrix operations
    - **Pillow (PIL)** – image handling
    - **Streamlit** – GUI framework
    - **Base64** – download functionality

Hardware Requirements

- A PC/Laptop with minimum 4GB RAM
- Browser (Chrome/Edge/Firefox)

---

4. Methodology

The workflow is divided into the following steps:

1. **Image Upload / Default Image**
    - Users can upload an image using Streamlit's file uploader.
    - If no image is uploaded, a **default photo** (e.g., KTM demo image) is displayed.
2. **Operation Selection**
    - Sidebar menu provides categories like **Color Conversion, Transformations, Filtering, Enhancement, Edge Detection, Compression**.
3. **Processing**
    - Based on user input, OpenCV functions (e.g., cv2.cvtColor, cv2.GaussianBlur, cv2.Canny) are applied.
    - NumPy handles array manipulation for stretching and scaling.
4. **Display Results**
    - Original and processed images are displayed side-by-side.
    - Image details (resolution, channels, size, format) are also shown.
5. **Save & Download**
    - Processed image can be downloaded in **PNG, JPG, or BMP** format.

---

5. Results

📌 GUI Preview

The application provides an **intuitive interface** where users can interactively test image operations.

- Example: Edge Detection using **Canny filter**
- Image info
- Colour conversion
- Edge detection
- Compression
- Transformation…..

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

sam.jpg 1.0MB ×

Display: Original vs Processed

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

sam.jpg 1.0MB ×

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...

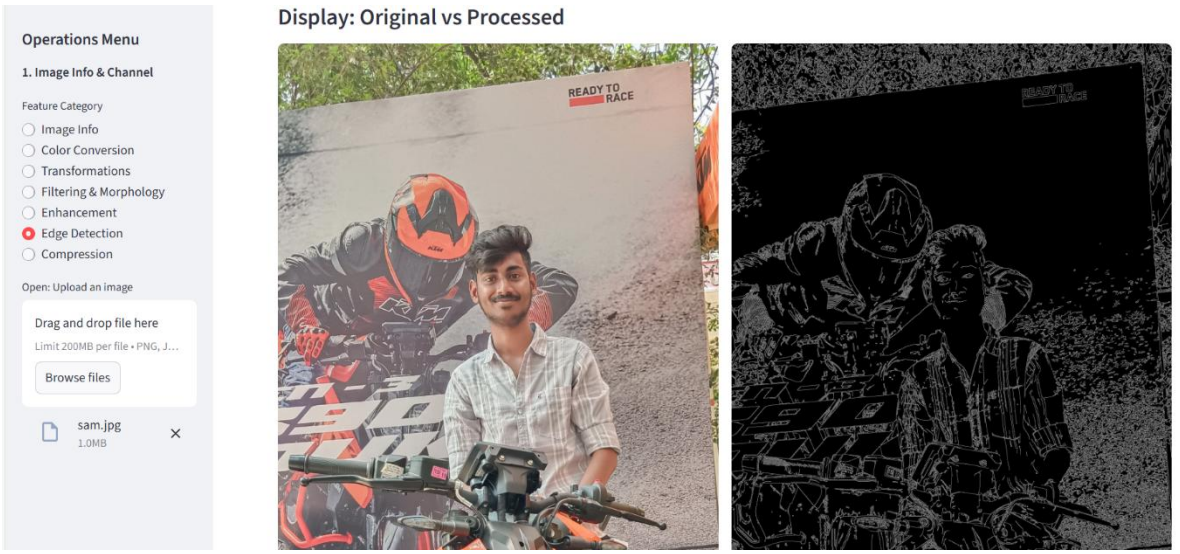
Browse files

sam.jpg 1.0MB ×

Rotation angle

20

Display: Original vs Processed



assignment.py | assign.py

assignment.py > ...

```python
def get_image_array(uploaded_file):
    img = Image.open(uploaded_file)
    return np.array(img)

def convert_img(img, code):
    return cv2.cvtColor(img, code)

def apply_rotation(img, angle):
    H, W = img.shape[:2]
    center = (W // 2, H // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    return cv2.warpAffine(img, M, (W, H))

def apply_scaling(img, factor):
    return cv2.resize(img, None, fx=factor, fy=factor, interpolation=cv2.INTER_LINEAR)

def apply_translation(img, x, y):
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
    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.
  Stopping...
(.venv) PS C:\Users\akkap\OneDrive\Desktop\eco_scan>
```

powersh... Python

6. Conclusion

The **Image Processing & Analysis Toolkit** demonstrates how advanced image operations can be simplified into a **click-based GUI application**.

- ✅ Makes learning easier for students.
- ✅ Reduces programming effort.
- ✅ Useful for **educational demonstrations, research, and prototyping**.

By combining **OpenCV + Streamlit**, this project bridges the gap between **theory and hands-on practice** in image processing.

A. Sampath
22671A7362
AIML-B