# Project Report: Image Processing Toolkit using Streamlit & OpenCV

Course: Image Processing & Analysis

Date: 07-09-2025

## 1. Problem Statement / Objective

The objective of this project is to design and implement an interactive image processing toolkit that allows users to: - Upload images in multiple formats. - Apply different image transformations, enhancements, and filters interactively. - Visualize both original and processed images side-by-side. - Save or download processed results in different formats.

## 2. Introduction / Assignment Tasks

Image processing is widely used in fields like computer vision, medical imaging, biometrics, and multimedia applications. This project integrates Streamlit (for UI) with OpenCV (for core image processing) to build a web-based interactive platform. Assignment tasks covered include: 1. Image upload, display, and format handling. 2. Basic operations: color space conversions, transformations. 3. Bitwise operations on multiple images. 4. Filtering, morphological operations, and edge detection. 5. Image enhancement techniques (histogram equalization, sharpening). 6. Exporting processed images in different formats.

## 3. Requirements Gathering / Libraries Used

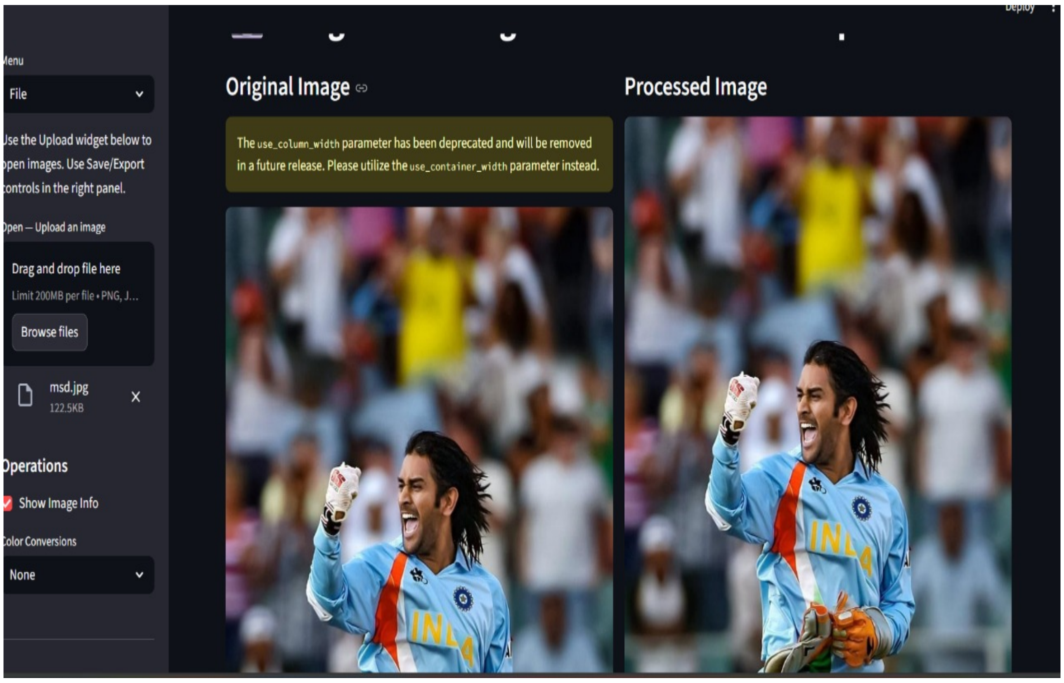
Requirements: - Simple GUI for non-programmers to experiment with images. - Fast, interactive feedback while applying transformations. - Support for multiple formats (PNG, JPG, BMP, TIFF). - Ability to download/save the processed results. Libraries Used: - Streamlit: for building the web-based GUI. - OpenCV (cv2): core image processing functions. - NumPy: numerical operations and image array handling. - PIL (Pillow): image conversion and export. - io, os, datetime: file handling and downloads.

## 4. Methodology / Code Explanation

The project is divided into functional blocks: a. Image Loading & Utilities: - Handles uploads using st.file_uploader(). - Converts uploaded file bytes into OpenCV-readable NumPy arrays. - Utilities for PIL ↔ OpenCV conversion, image metadata extraction, and byte export. b. Image Processing Operations: 1. Color Conversions: BGR ↔ RGB, HSV, YCrCb, Grayscale. 2. Transformations: Rotate, Scale, Translate, Affine, Perspective. 3. Bitwise Operations: AND, OR, XOR, NOT (with optional second image). 4. Filtering & Morphology: Gaussian, Median, Mean, Sobel, Laplacian, Dilation, Erosion, Opening, Closing. 5. Enhancement & Edge Detection: Histogram Equalization, Sharpening, Canny Edge. c. User Interface (Streamlit): - Sidebar menus for selecting operations and parameters. - Dual display columns: Original vs. Processed image. - Image info panel (dimensions, channels, format, size). - Save/export options with adjustable JPEG quality.

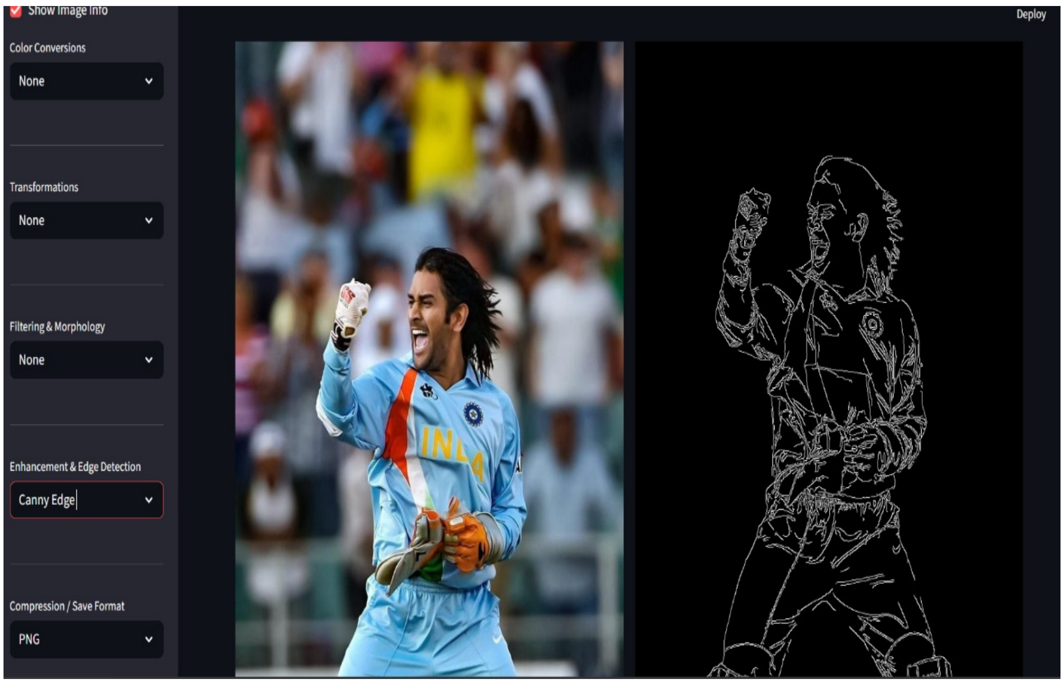
## 5. Results & Screenshots

The following screenshots showcase the implemented image processing toolkit. The system allows users to upload images, apply transformations, and view the original and processed outputs side-by-side.

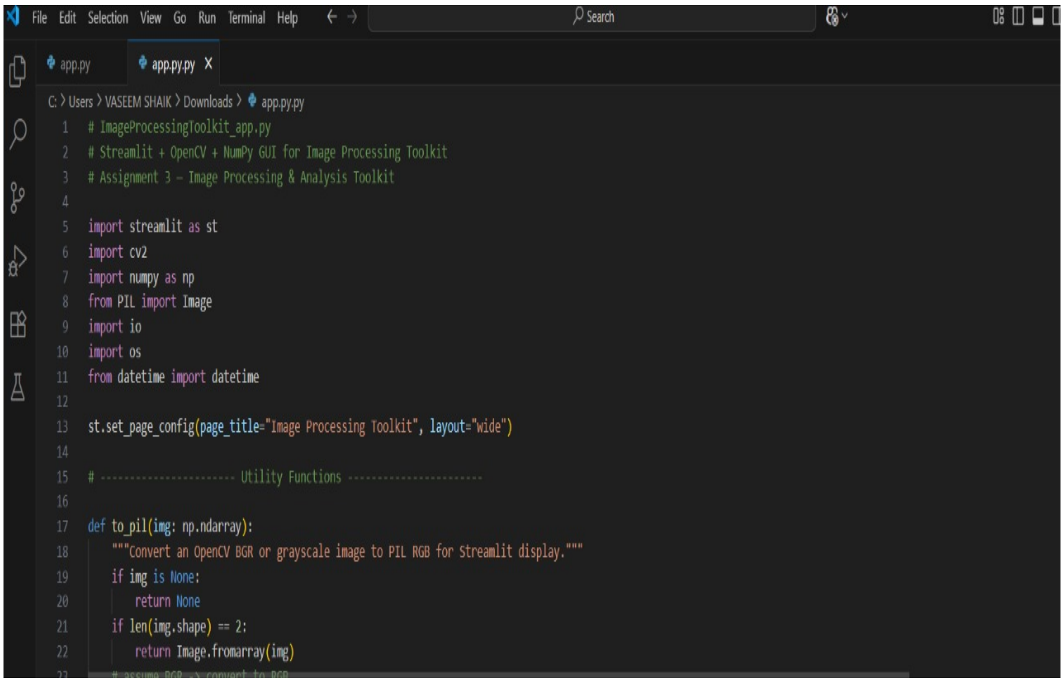


The use_column_width parameter has been deprecated and will be removed in a future release. Please utilize the use_container_width parameter instead.

*Original vs Processed Image*



*Canny Edge Detection Example*



File Edit Selection View Go Run Terminal Help
Search
app.py
app.py.py
C: > Users > VASEEM SHAIK > Downloads > app.py.py

```python
# ImageProcessingToolkit_app.py
# Streamlit + OpenCV + NumPy GUI for Image Processing Toolkit
# Assignment 3 - Image Processing & Analysis Toolkit

import streamlit as st
import cv2
import numpy as np
from PIL import Image
import io
import os
from datetime import datetime

st.set_page_config(page_title="Image Processing Toolkit", layout="wide")

# --------------------- Utility Functions ---------------------

def to_pil(img: np.ndarray):
    """Convert an OpenCV BGR or grayscale image to PIL RGB for Streamlit display."""
    if img is None:
        return None
    if len(img.shape) == 2:
        return Image.fromarray(img)
```

*Code Implementation Screenshot*

## 6. Conclusion

Knowledge Gained: - Practical use of OpenCV functions for image processing. - Building interactive web apps with Streamlit. - File handling, format conversion, and real-time UI updates. Difficulties Overcome: - Handling images with different channels (RGB, Grayscale, HSV). - Ensuring consistent BGR/RGB conversions between OpenCV and PIL. - Making kernel size adjustments for filters. - Preserving interactivity without performance lag. Future Improvements: - Add advanced filters (bilateral, non-local means denoising). - Support for video processing in real-time. - Region-of-interest (ROI) based editing. - Deep learning■based enhancements (super-resolution, segmentation).