A Report

on

Image Processing Using Streamlit-Based GUI

carried out as part of the course CSE CS3242 Submitted by

Ishaan Dhawan

229301676 &

Taanya Tapke

229309005

VI-CSE-A

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

MANIPAL UNIVERSITY
JAIPUR
(University under Section 2(f) of the UGC Act)

Under the Guidance of : Dr. Ajay Kumar

Guide Signature(with date) : ……

Table of Contents:

1. Introduction

Image Processing Using Streamlit-Based GUI

Image processing is a vital component of computer vision, where digital images are analyzed and transformed using various algorithms to extract information or enhance visual features. This project introduces an advanced **Streamlit-based web application** that empowers users to apply a wide range of image processing operations in a visually interactive environment.

The primary aim of this project is to bridge the gap between image processing theory and real-world usability by providing a GUI that allows real-time transformation and comparison of processed images. Unlike traditional command-line-based systems, this app offers intuitive controls like sliders and dropdowns to perform transformations such as **sharpening, blurring, enhancement, edge detection, rotation, interpolation, and noise addition.**

Problem Statement

Traditional image processing tools often demand programming proficiency, lack interactivity, or fail to offer real-time feedback. Users require platforms that not only visualize transformations but also allow customization, side-by-side comparison, and direct download capabilities.

This project solves the following challenges:

- Lack of interactive visualization during image processing
- Difficulty in comparing original vs processed outputs
- Complexity in applying compound transformations
- Limited access to downloadable results post-processing

2. Litrature Review

2.1 Historical Perspective

Classical image processing began with basic spatial filtering, histogram equalization, and morphological operations. As computational power increased, libraries like **OpenCV** (Open Source Computer Vision Library) became standard tools for implementing real-time processing. Similarly, the **Python Imaging Library (PIL)** and its successor **Pillow** made it easier to handle image enhancement, color correction, and format conversion in Python.

- **Key Processing Techniques**

- **Sharpening & Smoothing**: Used to enhance or reduce image details. Sharpening emphasizes edges and fine details, while smoothing (via filters like Gaussian or Median blur) reduces noise and smoothens transitions.

- **Edge Detection**: The **Canny Edge Detection** algorithm is one of the most widely used methods for detecting meaningful structural edges in an image. It is crucial in tasks such as object detection, contour extraction, and scene understanding.

- **Enhancement Techniques**: Enhancements like brightness, contrast, color balance, and sharpness are typically used in digital photography and medical imaging to improve interpretability.

- **Noise Addition**: Synthetic noise (e.g., Gaussian, salt & pepper) is often used in testing the robustness of filters or simulating real-world degradation.

- **Image Interpolation**: When scaling images, interpolation methods like **nearest-neighbor**, **bilinear**, and **bicubic** affect the smoothness and visual quality of the result. These are widely used in upscaling, zooming, and rendering applications.

2.2 Role of Streamlit Visualization

Streamlit has revolutionized data application development by offering an intuitive interface for deploying Python applications as web apps without requiring full-stack development knowledge. It allows developers to focus on logic and visuals, making it ideal for image processing demonstrations and experimentation.

2.3 Modern Trends

Recent advancements involve AI-based methods like neural style transfer, deep learning-based denoising, and super-resolution. However, classical methods remain highly relevant due to their simplicity, efficiency, and interpretability—especially for real-time and embedded applications.

3. Algorithm Description

This section outlines the core image processing operations implemented in the project using a combination of OpenCV, PIL (Pillow), and scikit-image. Each operation is performed interactively using sliders and selection widgets provided by the Streamlit interface.

3.1 Sharpening

Purpose: To enhance edges and fine details by emphasizing high-frequency components in the Image.

How it works: A custom sharpening kernel is applied via 2D convolution using OpenCV's filter2D() function. The kernel used is adjustable based on an intensity parameter, giving users control over the sharpness level.

Sharpened Image = Original Image * Kernel

3.2 Smoothing/Blurring

Purpose: To reduce noise and smooth transitions in images using various filters.

Techniques Implemented:

- **Gausian Blur:** Uses a Gaussian Kernel to reduce image detail and noise.
- **Median Blur:** Replaces each pixel's value with the median of the neighboring pixels; effective for salt and pepper noise.
- **Bilateral Filter:** Preserves the edges while blurring non-edge regions using color and spatial distance.

3.3 Image Enhancement

Purpose: To adjust various visual aspects of the image using PIL's ImageEnhance module.

Types of Enhancement

- **Contrast**
- **Brightness**
- **Color**
- **Sharpness**

Each type uses a user defined factor (typically ranging from 0.1 to 3.0) to control intensity.

3.4 Noise Addition

Purpose: To simulate noisy environments and test image robustness.

Noise Types:

- **Gaussian Noise:** Normally distributed noise with adjustable variance.
- **Salt & Pepper Noise:** Randomly replaces pixels with black or white.
- **Poisson Noise:** Adds noise based on the statistical variation in photon detection.

Implemented using random_noise() from skimage.util.

3.5 Rotation

Purpose : To rotate the image around its center without cropping or changing the resolution.

Method : Uses cv2.getRotationMatrix2D() followed by cv2.warpAffine() for transformation.

3.6 Canny Edge Detection

Purpose : To detect edges based on intensity gradients.

Steps:

- Convert image to grayscale.
- Apply Gaussian Blur (implicitly through thresholds).
- Apply hysteresis thresholding with two user-controlled thresholds.

3.7 Interpolation (Resizing)

Purpose: To scale the image up or down while maintaining quality using interpolation techniques.

Interpolation Methods:

- Nearest Neighbor
- Bilinear
- Bicubic

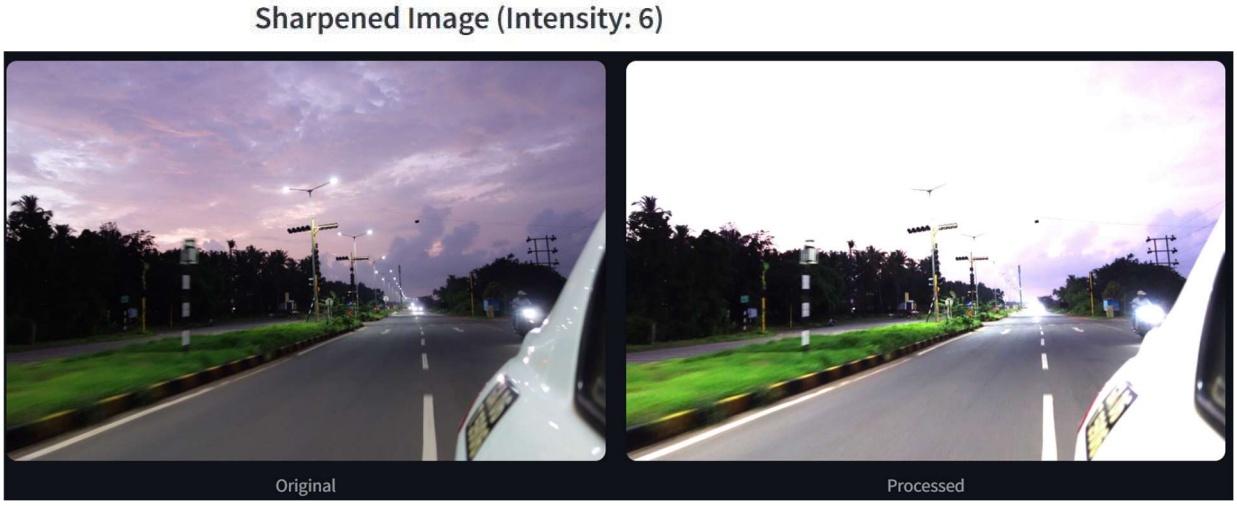
Image size is scaled based on a user-selected factor from 0.1x to 3.0x.

4. Experimental result

The image processing application was tested across various real-world images with diverse lighting, texture, and resolution conditions. The outcomes demonstrate the effectiveness and Flexibility of the implemented operations in transforming images with real-time control.

Sharpening:

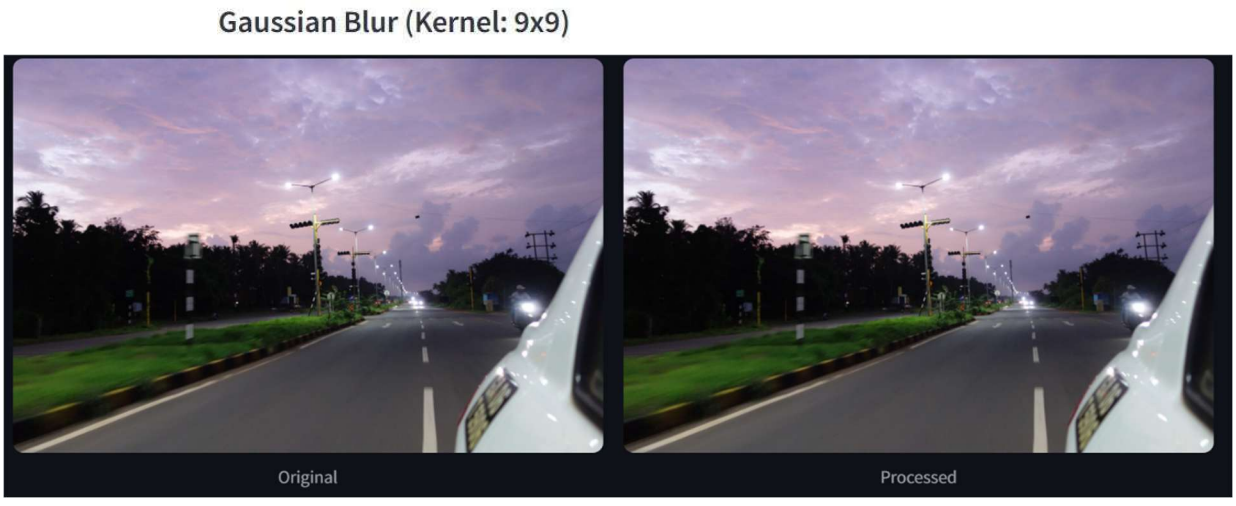
Enhanced edge clarity and boosted fine details, particularly in textured or architectural images. The sharpening intensity slider allowed for customizable enhancement strength.



Smoothing/Blurring :

Gaussian and median filters effectively reduced image noise. The bilateral filter preserved edges while softening flat regions, making it ideal for portrait smoothing.

Gaussian:



Median:

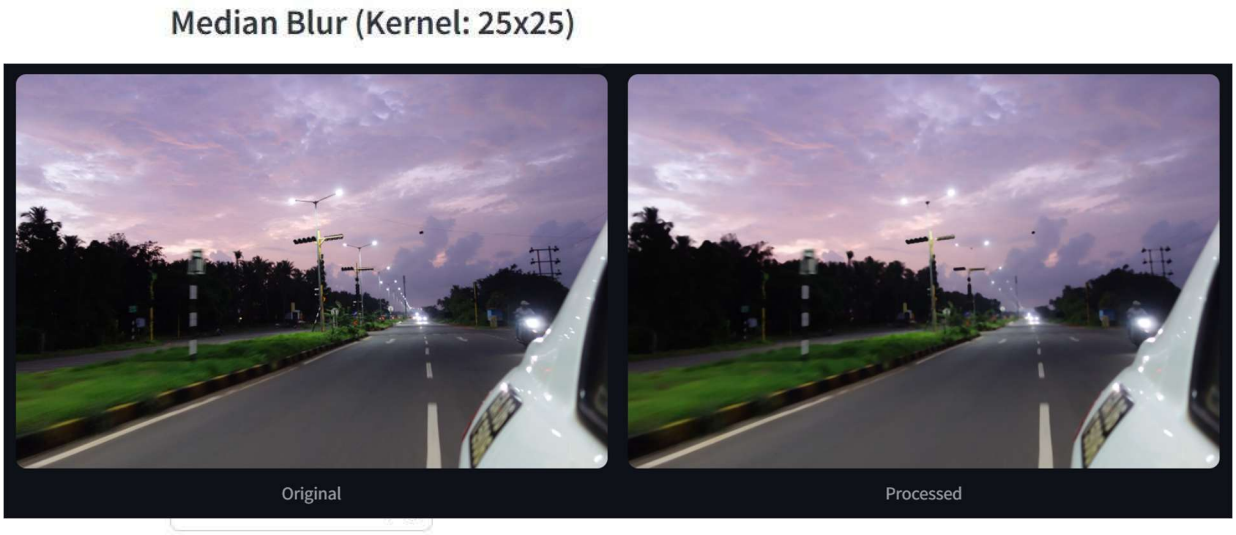
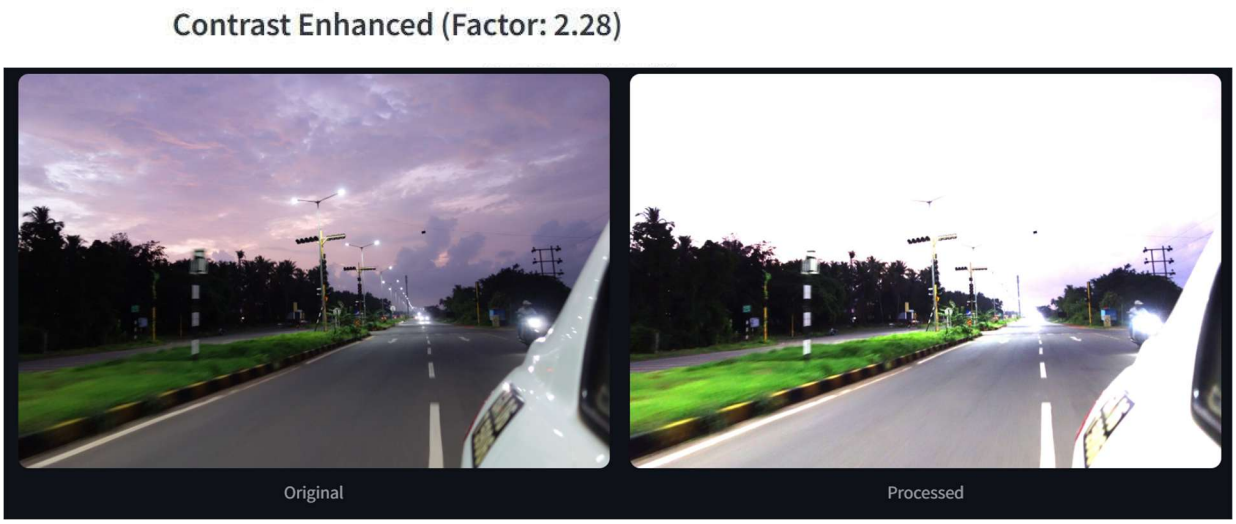


Image Enhancement:

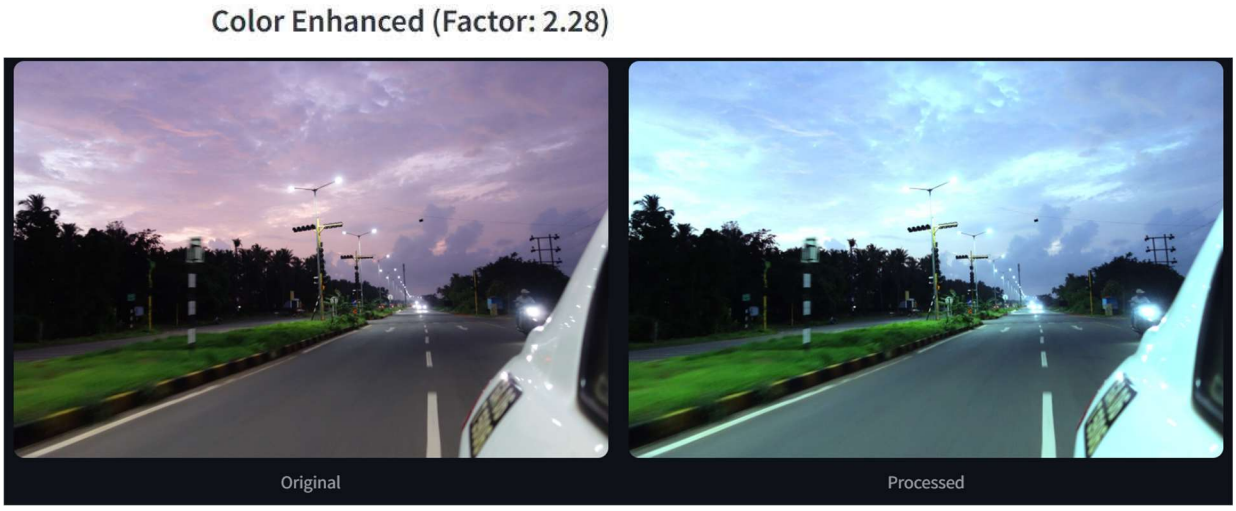
Adjustments to brightness, contrast, color saturation, and sharpness allowed for aesthetic improvements and restoration of underexposed images.

Contrast:

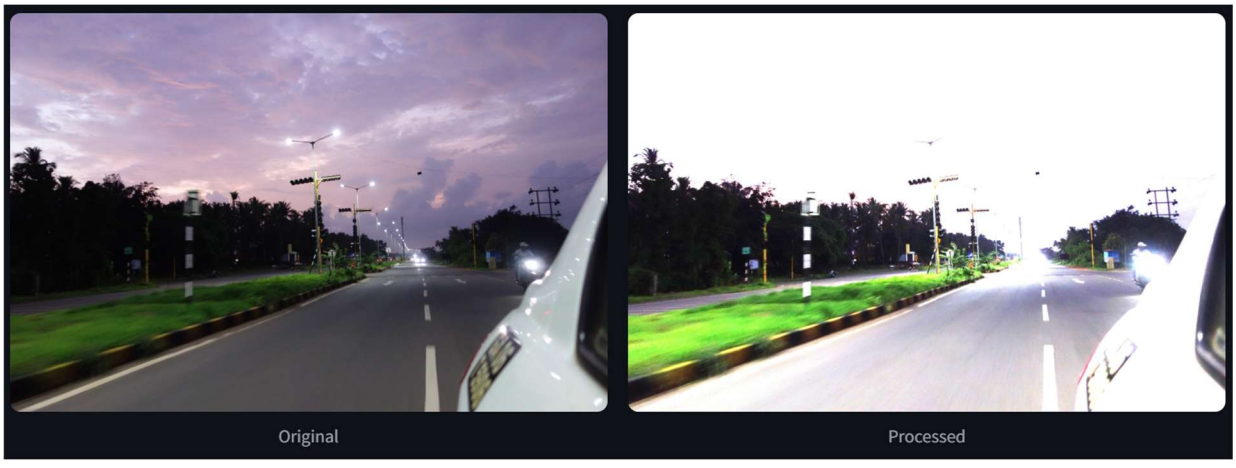


Brightness Enhanced:

Color Enhancement:



Sharpness:

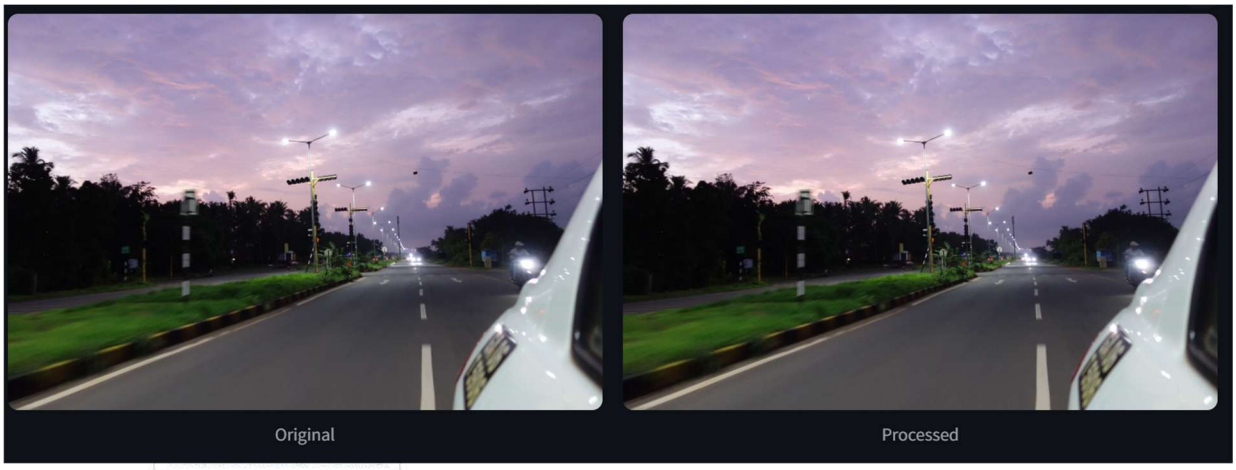


Noise Addition:

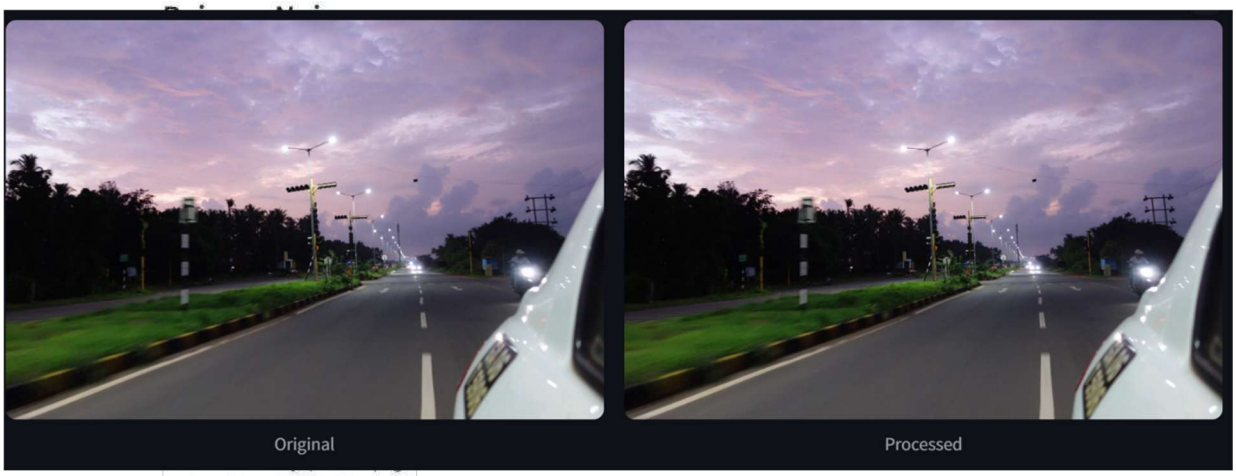
Controlled noise addition (Gaussian, Salt & Pepper, Poisson) helped in testing robustness and simulating real-world degradation for image augmentation experiments.

Gaussian:

Salt and Pepper Noise:

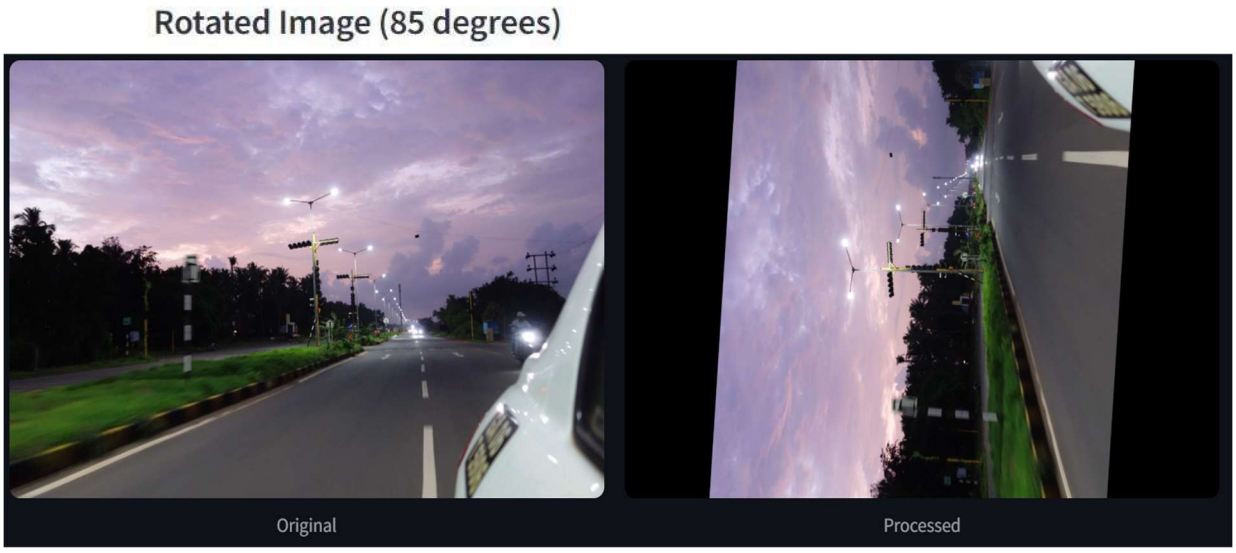


High Pass:



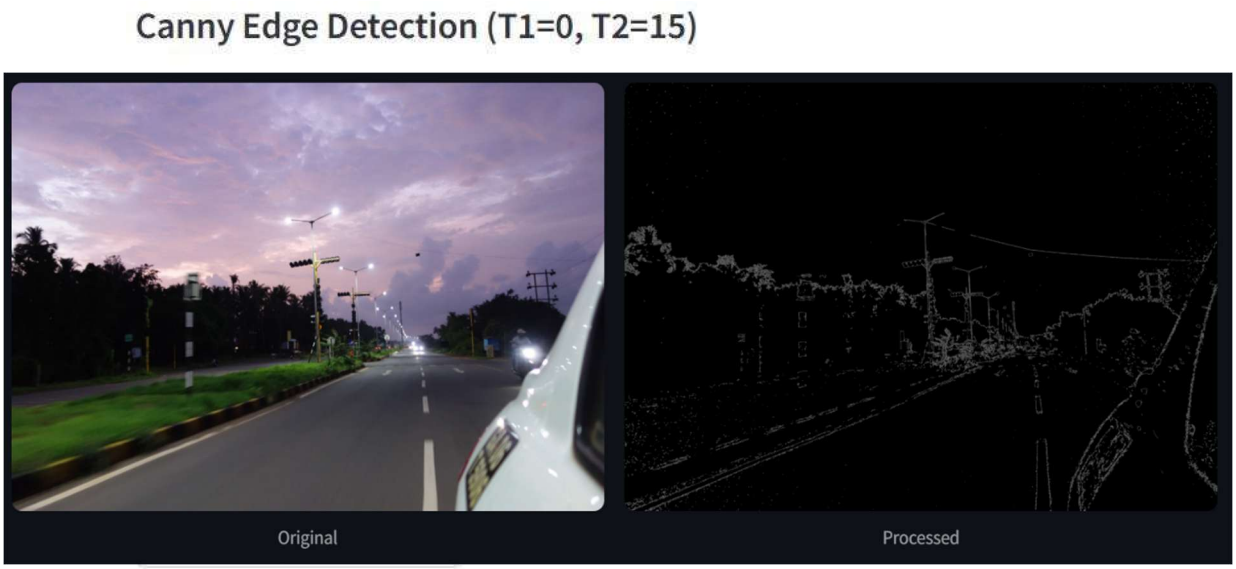
Rotate:

Provided seamless and distortion-free image rotation up to ±180 degrees without cropping or loss of quality.



Canny Edge Detection:

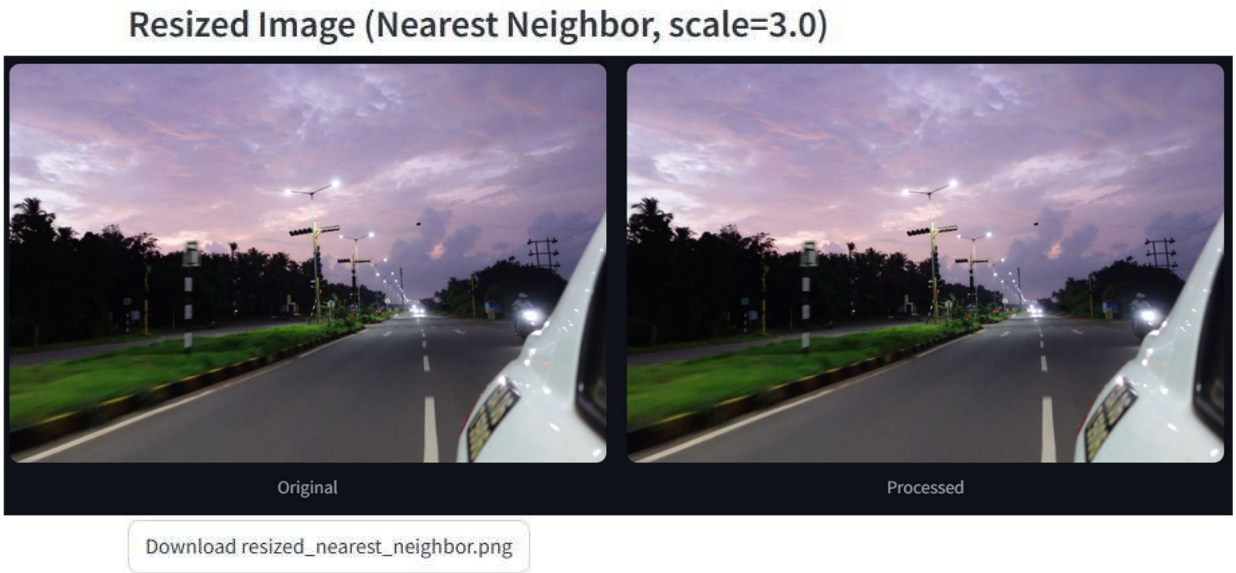
Delivered accurate contour mapping and edge highlighting, especially effective in high-contrast or grayscale images.



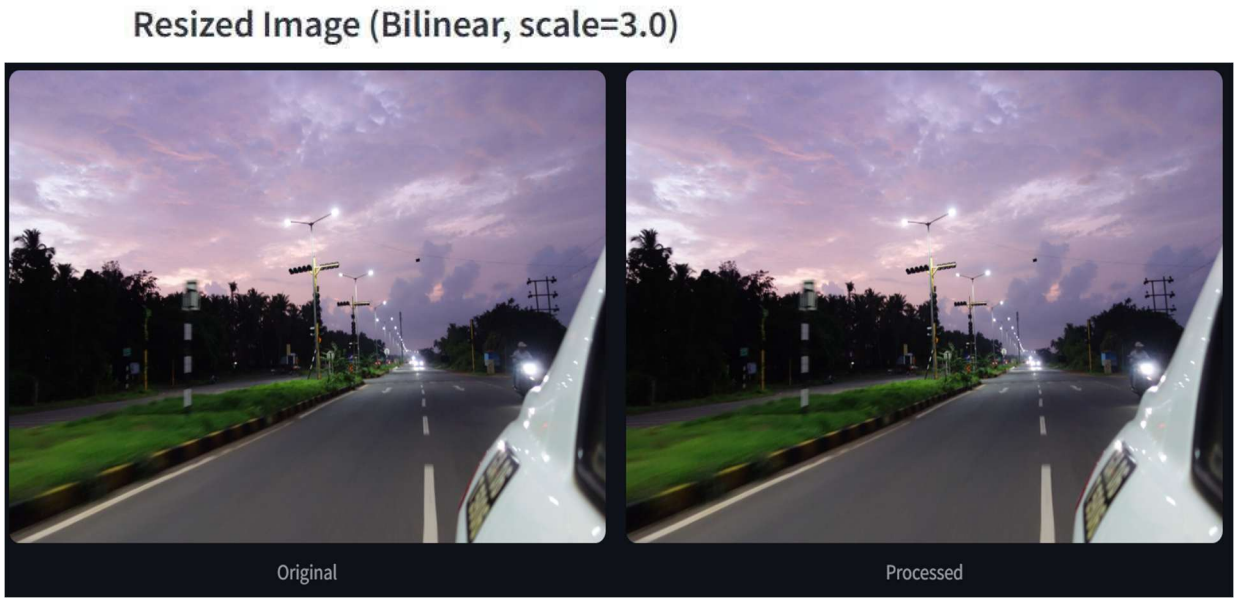
Interpolation (Resizing):

Enabled image upscaling and downscaling with user-selected interpolation methods. Bicubic and bilinear methods maintained smooth transitions with minimal artifacts.

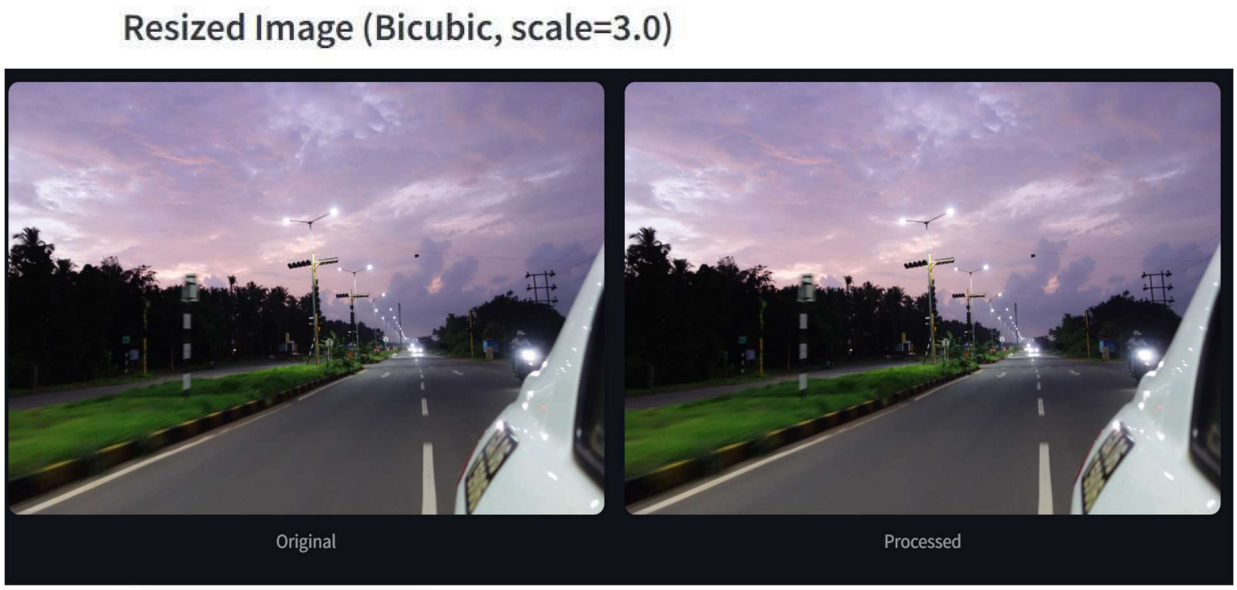
Nearest Neighbour:



Bilinear:



Bicubic:



Summary and Observation:

The results from each transformation validate the functionality and versatility of the application. Each operation produced expected visual outcomes, and the interactive controls provided an intuitive environment for real-time experimentation. The ability to compare original and processed images side-by-side significantly enhanced usability and made it easy to interpret the effects of parameter changes.
Overall, the application succeeded in delivering a responsive and effective image processing experience suitable for educational, experimental, and practical use cases.

5. Conclusion

This project demonstrates the practical application of classical image processing techniques within an intuitive, interactive, and accessible web interface using **Streamlit**. By combining the strengths of OpenCV, PIL, and Matplotlib with user-friendly GUI components, the application allows users to perform a wide variety of transformations—ranging from sharpening and smoothing to enhancement, noise simulation, edge detection, and interpolation.

The key achievements of the project include:

- Real-time parameter tuning with immediate visual feedback
- Seamless comparison between original and processed images
- Download functionality for all transformed outputs
- Elimination of the need for deep technical expertise to perform image processing tasks

Through this tool, the gap between technical implementation and user accessibility has been effectively bridged. It can serve as a valuable resource for education, experimentation, and quick prototyping in the fields of computer vision, digital art, and user-interface design.

Future Scope:

- Integration of advanced AI-powered features like denoising using deep learning
- Real-time video feed processing
- Batch processing and cloud storage integration
- Deployment on mobile platforms or as a progressive web app (PWA)

6. References:

- Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th Edition). Pearson Education.*(A foundational textbook widely used in image processing and computer vision courses.)*

- Jain, A. K. (1989). *Fundamentals of Digital Image Processing*. Prentice-Hall, Inc. *(Covers core concepts such as filtering, enhancement, and restoration techniques.)*

- Pratt, W. K. (2007). *Digital Image Processing: PIKS Scientific Inside* (4th Edition). Wiley-Interscience.*(Discusses practical applications and algorithms behind modern image transformations.)*

- Canny, J. (1986). *A Computational Approach to Edge Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), 679–698. https://doi.org/10.1109/TPAMI.1986.4767851

- Tomasi, C., & Manduchi, R. (1998). *Bilateral Filtering for Gray and Color Images*. Proceedings of the 1998 IEEE International Conference on Computer Vision. https://doi.org/10.1109/ICCV.1998.710815