COMPUTER VISION
MINI PROJECT
IMAGE PROCESSING PLAYGROUND
REPORT

ARUN SANKAR M

2023510033

AI AND DS

SEMESTER:4

INTRODUCTION

This mini project involves building an **image processing playground** where users can upload their own images and experiment with various basic image processing techniques. These include **adding noise**, **applying filters**, and **performing edge detection**. Users can also try out different combinations of these techniques for more customized results.

The frontend is built using **HTML, CSS, and JavaScript**, while the backend is powered by **Python** and connected via the **Flask** framework.

Website URL : https://arunsankar18.pythonanywhere.com

Why This Project ?

The main goal of this project is to provide a user-friendly **interface for experimenting with image processing techniques**, making it easier to understand how these techniques affect different types of images. Users can try various combinations of operations—for example, **adding noise to an image**, then applying different filters to see which one best removes the noise. You can also perform **edge detection** on both noisy and clean images for comparison.

Additionally, users can upload their own noisy images, apply filters, and view the results in real-time. This platform is especially useful for **students or anyone interested in learning about image processing**, rather than for general audiences.

About The Project

I will begin by explaining how to use the website and what features it offers. After that, I'll give a glance of the technical aspects behind the project and how I deployed the website online.

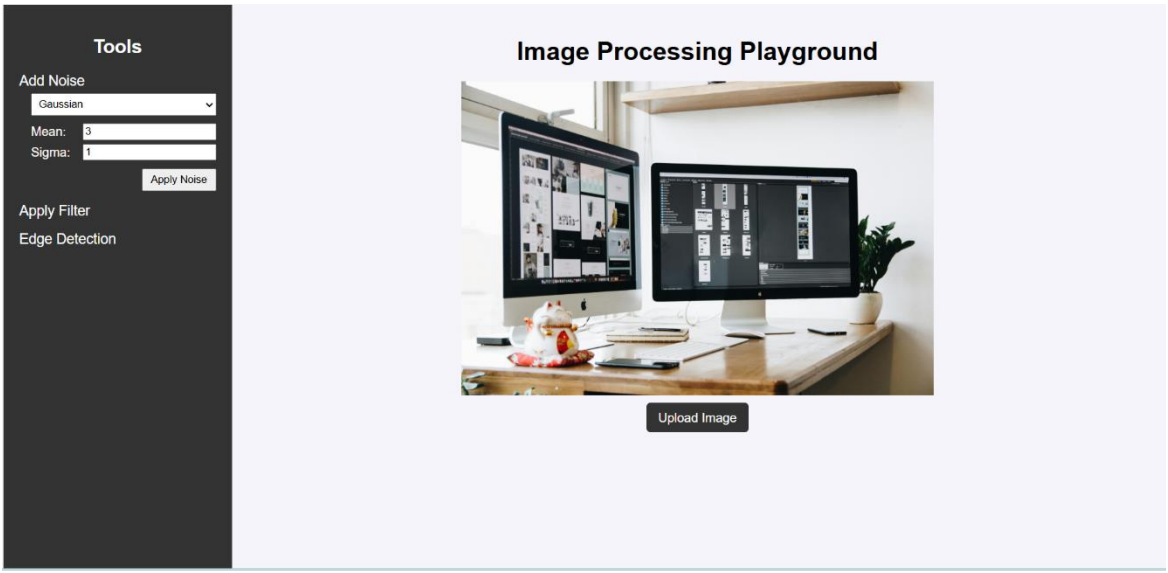
You can access the website using the URL provided on the previous page. By default, a sample image is available for experimentation, but you can also upload your own image if you prefer.

On the left side of the interface, you'll find a set of tools, including:

- **Add Noise**
- **Apply Filter**
- **Edge Detection**

When you click on **Add Noise**, you'll be able to select the type of noise you want to apply. Each option comes with adjustable parameters. If you're unsure, default values are provided and can be used as-is.

Here's a preview of how the interface looks:



Project Overview – Image Processing Playground

This project includes a total of **six noise types**, which users can apply to their images:

- **Gaussian Noise**: Parameters – Mean, Sigma
- **Rayleigh Noise**: Parameter – Scale
- **Gamma Noise**: Parameters – Shape, Scale
- **Salt and Pepper Noise**: No additional parameters
- **Uniform Noise**: Parameters – Low and High values
- **Periodic Noise**: Parameter – Frequency

Each noise type comes with adjustable parameters (when applicable). Default values are provided, but you're free to customize them. If you're interested in how these parameters affect the image, you may refer to standard textbooks or online resources. We've left those explanations to the reader to explore further.

Filters Available

The **Apply Filter** section includes the following **nine filters**:

- Mean Filter
- Harmonic Mean Filter
- Contra-Harmonic Mean Filter *(requires kernel size and Q value)*
- Median Filter
- Min Filter
- Max Filter
- Adaptive Mean Filter
- Adaptive Median Filter
- Box Filter *(requires box width and height)*

Except for the Contra-Harmonic and Box filters, all filters only require **kernel size** as a parameter to keep things simple.

Edge Detection Techniques

Five edge detection methods are included:

- **Sobel**: Parameter – Kernel size
- **Roberts**: No parameters
- **LoG (Laplacian of Gaussian)**: Parameters – Kernel size, Sigma
- **DoG (Difference of Gaussians)**: Parameters – Sigma1, Sigma2
- **Canny**: Parameters – Low and High Threshold

These techniques allow you to explore how edge detection behaves under various image conditions.

Workflow and Image Handling

As mentioned earlier, the **image shown in the preview** acts as the current working image. For example, if you apply Salt and Pepper noise to the image, that altered version becomes the input for the next operation—say, a Median filter.

If you want to apply different techniques starting from the original image again, you will need to **re-upload the image** or use the **Reset** option.

In case of any issues, simply **refresh the page**. A "Reset to Initial Image" feature is planned for future updates.

Once you upload an image, just select a technique and apply it. A **loading animation** will appear while the backend processes your request, and the result will be displayed in the UI once ready.

Other Features

- **Upload**: Upload your own image
- **Reset**: Reset to the default image
- **Download**: Save the processed image to your device

Project Demo

You can watch a demonstration of this project on GitHub using the following link:

Arun-Sankar-lab/WEBSITE-CV-PROJECT

Glance At Technical Aspects

I have used **basic HTML** and **CSS** for styling, with **JavaScript** handling the frontend logic. The backend is powered by **Python**, connected to the frontend using **Flask**. For image processing tasks, I utilized **OpenCV**.

To publish the website online, I took advantage of the **three-month free trial** on **PythonAnywhere**.

Here's how the system works:

- When a user selects a technique (e.g., adding noise, applying filters, edge detection), it triggers a request to the backend.
- The backend processes the image and stores the processed file in a folder.
- The backend sends the filename of the processed image to the frontend.
- The frontend retrieves the file using the filename and displays the processed image in the preview section.

Currently, the deletion process for images is **manual**, but I plan to automate it in the future.

If you're interested in the source code, you can find it on **GitHub** link above.