

Web-Based Image Processing Application

Computer Vision Project Report

Amira Yasser Mohamed
Alhussien Ayman Hanafy
Ahmed Ahmed Mokhtar
Osama Magdy Ali

February 26, 2026

Contents

1	Project Overview	3
2	System Architecture	3
2.1	Backend (FastAPI)	3
2.2	Frontend (JavaScript)	3
3	Application Interface	4
3.1	Home Page	4
3.2	Operations UI	5
4	Task 1: Noise Addition	5
4.1	Observations	5
5	Task 2: Low Pass Filtering	6
5.1	Observations	6
6	Task 3: Edge Detection	6
6.1	Observations	6
7	Histogram and Distribution Curve	7
7.1	Observations	7
8	Histogram Equalization	7
8.1	Observations	7
9	Normalization	7
9.1	Observations	7
10	Frequency Domain Filtering	7
10.1	Observations	8
11	Hybrid Images	8
11.1	Observations	8
12	Conclusion	8

1 Project Overview

This project implements a web-based image processing application using:

- **Backend:** FastAPI (Python + OpenCV)
- **Frontend:** Vanilla JavaScript (HTML, CSS)

The system allows users to:

- Upload images
- Apply spatial domain filters
- Perform edge detection
- Apply histogram operations
- Perform frequency domain filtering
- Generate hybrid images

The application provides an interactive UI to test different parameters and observe their effect in real-time.

2 System Architecture

2.1 Backend (FastAPI)

- Image upload handling
- OpenCV-based image processing functions
- REST API endpoints for each operation
- Parameter validation

2.2 Frontend (JavaScript)

- Image preview
- Parameter input forms
- Asynchronous API calls
- Result visualization

3 Application Interface

3.1 Home Page

The screenshot shows the main landing page of the Image Processing Lab. At the top, there's a navigation bar with links for Home, Operations Lab, Demos, and Documentation. Below the navigation is a large banner with the text "Image Processing Lab" and "Unlock the Power of Pixels". A blue button labeled "Launch demo" is centered in the banner. Below the banner, there are four numbered callouts: "3 noise types: Gauss, uniform, salt&pepper", "11 max kernel size · average / Gaussian / median", "4 edge detectors: Sobel, Roberts, Prewitt, Canny", and "3 hybrid images: low · high · final composition". The main content area features two images: one of a person's eye with a futuristic interface overlay, and another of a person sitting on a couch with large speakers. To the right of these images is a section titled "architecture" with the heading "original image · working copy · undo without reload". It contains a bulleted list of operations and a "explore all operations" button. Below this is a section titled "The playground of Operations" which lists six core modules: Noise addition, Low-pass filters, Edge detection, Histogram & CDF, Frequency domain, and Hybrid image, each with a brief description and icon.

architecture

original image · working copy · undo without reload

- Image read once – stored as original (unchanged) and working copy.
- All operations affect only the working copy. Undo reverts to original without re-reading.
- Noise addition: Gaussian (mean, variance clipped), uniform (low/high), salt & pepper (ratio). Parameters restricted to reasonable intervals. Applied only once – no stacking.
- Spatial filtering from scratch: average, Gaussian, median — kernel size odd 3-11, user controlled.
- Edge detection (manual): Sobel, Roberts, Prewitt – gradient magnitude. Canny via OpenCV allowed.
- Histogram + distribution curve + CDF (cumulative max = 1 after normalization). Per-channel RGB histograms.
- Frequency domain: FFT → low/high-pass → IFFT. Hybrid Image = low-freq (filtered) + high-freq (original – low) and displayed.

[explore all operations](#)

The playground of Operations

Noise addition
Gaussian (μ, σ^2), uniform [a,b], salt & pepper (ratio). Parameters bounded. Applied once, undo available.

Low-pass filters
Average, Gaussian, median — all from scratch. Kernel size: odd 3-11 (user choice). No OpenCV filtering.

Edge detection
Sobel, Roberts, Prewitt (gradient magnitude, scratch). Canny via OpenCV. Full gradient display.

Histogram & CDF
Draw histogram + distribution curve + cumulative function. After normalization CDF reaches 1. RGB channel plots.

Frequency domain
FFT → low-pass / high-pass filters → inverse FFT. Visualise filtered image.

Hybrid image
Combine low-frequency (strong LPF) and high-frequency (original – low) into one hybrid. Display all three.

core modules

- Noise (Gauss/uniform/s&p)
- Spatial low pass filters
- Edge (Sobel,Roberts,Prewitt)
- Canny (OpenCV)

more

- histogram + CDF / equalization
- frequency domain filters
- hybrid image (low+high)
- per-channel RGB histograms

parameter ranges

- noise mean: 0-100, variance >0
- kernel size: odd 3-11

All operations preserve original, modify working copy.

Figure 1: Application Home Page

3.2 Operations UI

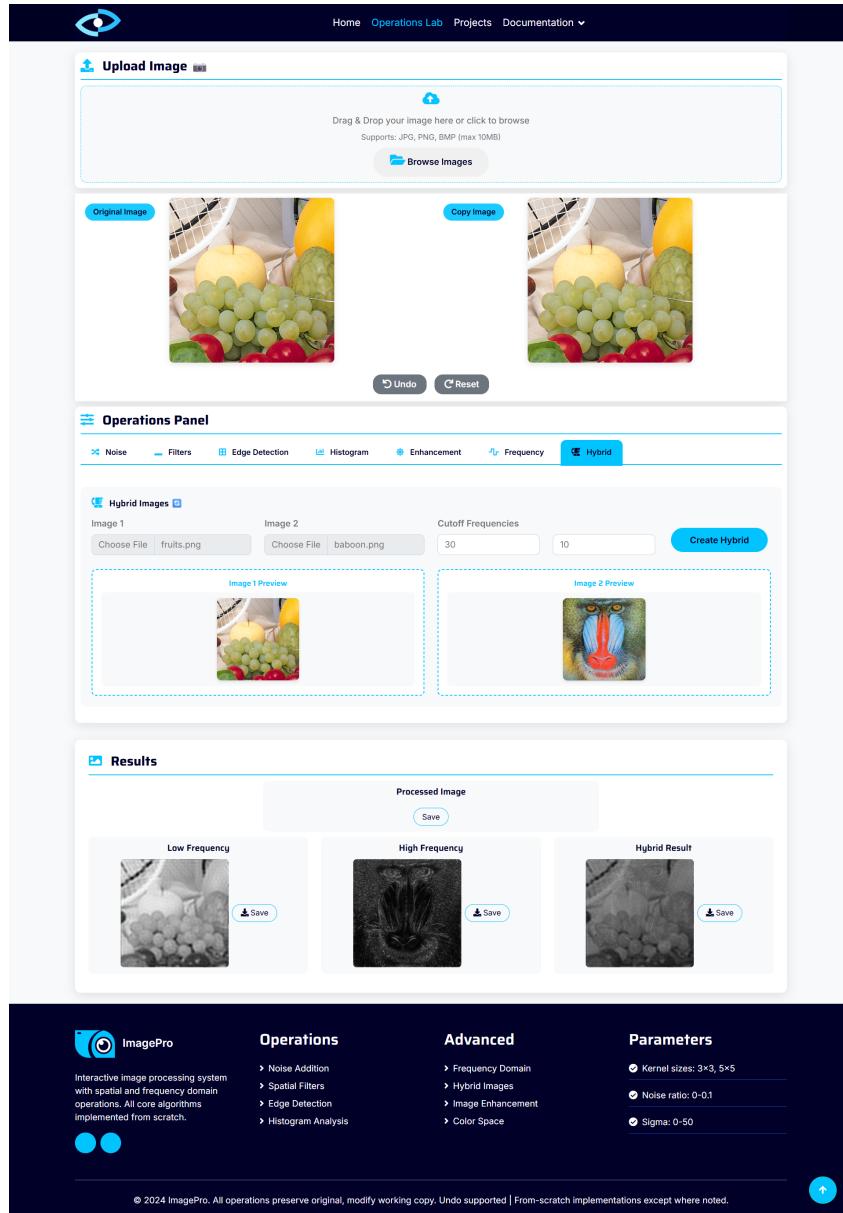


Figure 2: Operations User Interface

4 Task 1: Noise Addition

We implemented additive noise using OpenCV:

- Uniform Noise
- Gaussian Noise
- Salt & Pepper Noise

4.1 Observations

- Gaussian noise produces smooth intensity variation.

- Salt & Pepper noise creates sharp black and white pixels.
- Increasing variance increases distortion level.

Noise Type	Parameter	Effect
Gaussian	Variance \uparrow	Stronger distortion
Salt & Pepper	Density \uparrow	More impulse pixels
Uniform	Range \uparrow	Wider intensity spread

Table 1: Noise Parameter Effects

5 Task 2: Low Pass Filtering

Implemented filters:

- Average Filter (3x3, 5x5)
- Gaussian Filter (3x3, 5x5)
- Median Filter (3x3, 5x5)

5.1 Observations

- Median filter performs best for Salt & Pepper noise.
- Gaussian filter preserves edges better than average filter.
- Larger kernel size increases smoothing but blurs edges.

6 Task 3: Edge Detection

Implemented:

- Sobel (X and Y)
- Roberts
- Prewitt
- Canny (OpenCV built-in)

6.1 Observations

- Sobel provides directional edge information.
- Roberts is sensitive to noise.
- Prewitt is similar to Sobel but slightly less accurate.
- Canny produces the cleanest and thinnest edges.

Method	Noise Sensitivity	Edge Quality
Sobel	Medium	Good
Roberts	High	Weak
Prewitt	Medium	Moderate
Canny	Low	Excellent

Table 2: Edge Detection Comparison

7 Histogram and Distribution Curve

We plotted:

- Intensity histogram
- Cumulative distribution function (CDF)

7.1 Observations

- Low contrast images show narrow histogram distribution.
- Equalization spreads histogram across full intensity range.

8 Histogram Equalization

8.1 Observations

- Improves contrast in low contrast images.
- May over-enhance already high contrast images.

9 Normalization

Image intensities were scaled to [0, 255].

9.1 Observations

- Useful before thresholding.
- Improves dynamic range usage.

10 Frequency Domain Filtering

Implemented:

- Low Pass Filter (Gaussian)
- High Pass Filter

10.1 Observations

- Low pass removes noise but blurs image.
- High pass enhances edges.
- Cutoff frequency controls level of detail.

11 Hybrid Images

Hybrid images were created by:

- Applying low-pass filter to one image
- Applying high-pass filter to another image
- Adding both results

11.1 Observations

- Seen up close → high frequency image dominates.
- Seen from far → low frequency image dominates.

12 Conclusion

This project successfully implemented a complete image processing pipeline in a web-based interactive application.

Key achievements:

- Modular FastAPI backend
- Interactive frontend UI
- Parameter-based experimentation
- Spatial and frequency domain processing

The system allows testing different algorithms and parameters dynamically, which helps understand the behavior of image processing techniques practically.