Computer Vision – Task 1

Image Processing App

Team 20

Name	Section	Bench Number
Abdullah Mohamed	1	39
Mohamed Gamal	2	12
Mohamed Ali	2	22

1. Introduction:

The Image Processing Tool is a PyQt-based application designed to provide users with a range of image processing functionalities. It offers a user-friendly interface to manipulate images, apply filters, perform edge detection, and more.

2. Features:

Image Loading: Users can load images from their local file system using the "Open" button. Supported image formats include PNG, JPG, BMP, JPEG, TIFF, etc.

Image Display:

Loaded images are displayed in the main window using QGraphicsView. The application supports the display of multiple images simultaneously.

Image Processing

Noise Addition: Users can add different types of noise to images, including Gaussian noise, uniform noise, and salt-and-pepper noise.

Blur Filters: Various blur filters such as median blur, average blur, and Gaussian blur can be applied to images.

Edge Detection: Edge detection algorithms like Sobel, Prewitt, and Roberts can be used to highlight edges in images.

Thresholding: Global and local thresholding techniques are available for segmenting images based on pixel intensity.

Hybrid Image Generation: Users can create hybrid images by combining two input images with specified low-pass and high-pass cutoff frequencies.

Histogram Analysis

Users can visualize the histogram of loaded images to analyze pixel intensity distributions.

Histogram equalization can be performed to enhance image contrast.

3. Usage

Image Processing Actions: Users can perform various image processing actions by interacting with buttons and sliders provided in the application interface.

Thresholding: Users can adjust threshold values for global and local thresholding operations using input fields.

Hybrid Image Generation: Users can select two images and set low-pass and high-pass cutoff frequencies using sliders to generate hybrid images.

4. Backend Integration

The application is integrated with a backend API client (APIClient) responsible for communicating with the server to perform image processing tasks. It sends requests to the server and receives processed images in response.

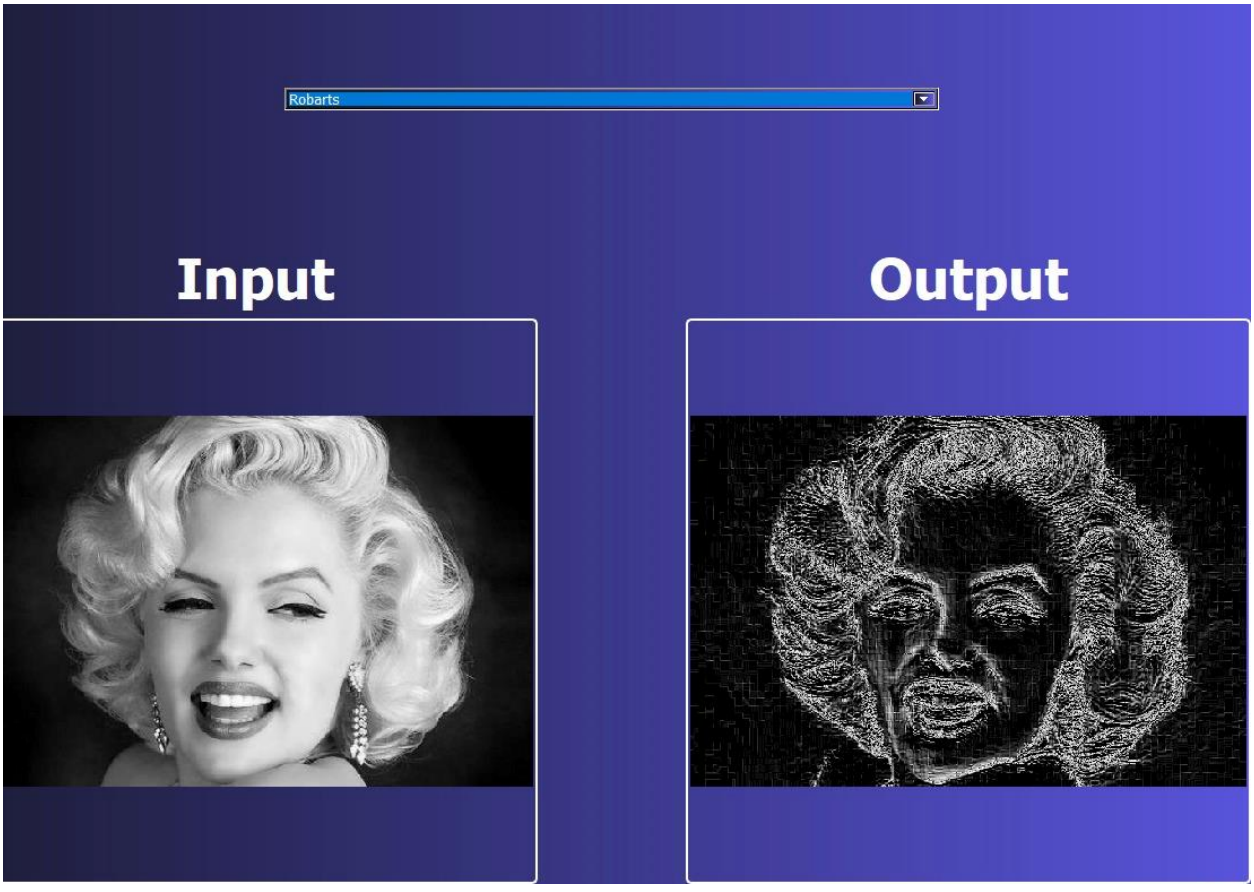
5. User Authentication

The application requires user authentication before accessing backend services. Users need to provide valid credentials (username and password) to log in to the system.

6. Discussion of Findings

1. Edge Detection Using the Roberts Operator

Edge detection using the Roberts operator is a common technique in image processing for detecting edges without the need for preprocessing steps such as blurring or noise reduction. The Roberts operator is a simple and computationally efficient edge detection algorithm that operates directly on the raw image data. It works by computing the gradient magnitude of the image, which highlights regions of rapid intensity change, typically indicative of edges.



The Roberts operator applies two 2x2 convolution kernels to the image, one for detecting vertical edges and the other for horizontal edges. These kernels are specifically designed to approximate the gradients of the image along the x and y directions. By convolving these kernels with the image,

the Roberts operator calculates the gradient magnitude at each pixel, which represents the rate of change of intensity in the image.

The resulting gradient magnitude image reveals regions with significant intensity changes, effectively highlighting edges in the original image. However, it's important to note that the Roberts operator is sensitive to noise and can produce spurious edge detections, especially in noisy images. Without preprocessing steps such as noise reduction, these artifacts may be more pronounced.

Despite its simplicity, the Roberts operator is often used as a basic edge detection technique in scenarios where computational efficiency is paramount and where the input images are relatively clean or noise-free. However, in practical applications involving noisy images or where more precise edge detection is required, more sophisticated techniques such as the Sobel or Canny edge detectors, which incorporate additional preprocessing steps, may be preferred.

2. Gray Image Transformation

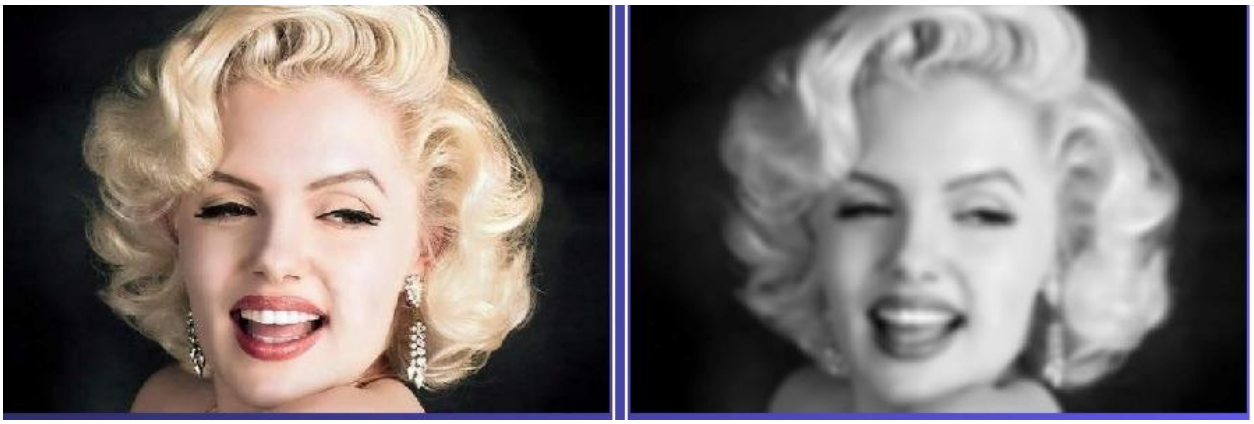
Once the original image is converted to grayscale, each pixel is represented by a single intensity value ranging from 0 (black) to 255 (white), where intermediate values represent shades of gray. Converting an image to grayscale is a common preprocessing step in image processing tasks, including edge detection, as it simplifies the image while preserving important structural information.

The grayscale representation removes color information from the image, focusing solely on the luminance or brightness of each pixel. This simplification is often beneficial for edge detection algorithms, as they primarily rely on variations in intensity rather than color.

After converting to grayscale, the image maintains its spatial dimensions but consists of only one channel instead of three (as in the case of a color image). This simplifies the computational complexity of subsequent processing steps while retaining essential features for edge detection algorithms to operate effectively.

3. Gaussian Filter Kernal Size = 7 and STD = 10

When applying a Gaussian filter with a kernel size of 7x7 and a standard deviation (std) of 10 to an image, the resulting filter will have a larger spatial extent and a higher degree of smoothing compared to filters with smaller kernel sizes or lower standard deviations.



The Gaussian filter is a popular choice for image smoothing because it effectively blurs the image while preserving important structural details. The kernel size determines the spatial extent of the filter, with larger kernels encompassing more neighboring pixels in the convolution operation.

In this case, with a kernel size of 7x7, the Gaussian filter will consider a 7x7 neighborhood around each pixel during convolution. This larger neighborhood allows for a more extensive blending of pixel intensities, resulting in smoother transitions between adjacent regions in the image.

4. Gaussian Noise with Mean = 20 and STD = 30

Gaussian noise with a mean of 20 and a standard deviation of 30 represents a type of random noise that can be added to an image. This noise follows a Gaussian (normal) distribution, where most values are centered around the mean of 20, with a spread determined by the standard deviation of 30.

When applied to an image, Gaussian noise introduces random variations in pixel intensity, simulating the effects of noise present in real-world imaging systems or during image acquisition processes. The mean value of 20 represents the average amount of noise added to each pixel, while the standard deviation of 30 controls the spread or magnitude of this noise.

A higher standard deviation results in a broader distribution of noise values, leading to more significant variations in pixel intensity throughout the image. This means that with a standard deviation of 30, the magnitude of noise added to each pixel can range widely, with some pixels having values much higher or lower than the mean of 20.

7. Conclusion

The Image Processing Tool offers a comprehensive set of functionalities for image manipulation and analysis. It provides an intuitive interface for users to perform various image processing tasks effectively.

8. Future Enhancements

Support for additional image processing algorithms and filters.

Integration of more advanced edge detection techniques.

Improved user interface with enhanced visualization and interaction capabilities.