Computer Vision

VisualFX Studio: Your Ultimate Image Enhancement Suite

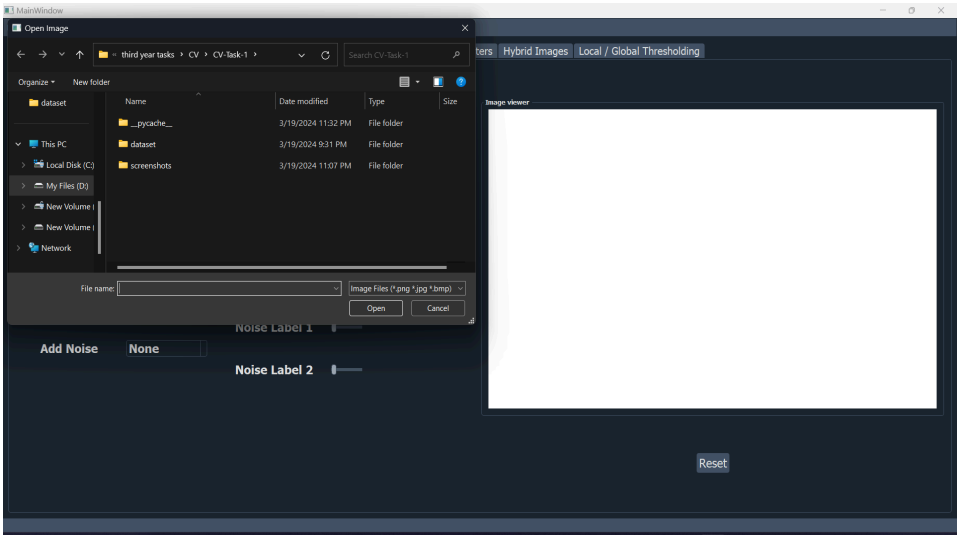
Task 1 - Team 11

Introduction

This project is an image processing application developed using PyQt5 in Python. The application provides various functionalities for image manipulation, including adding noise, filtering, edge detection, histogram analysis, equalization, normalization, thresholding, frequency domain filtering, and creating hybrid images.

Features

Load Images



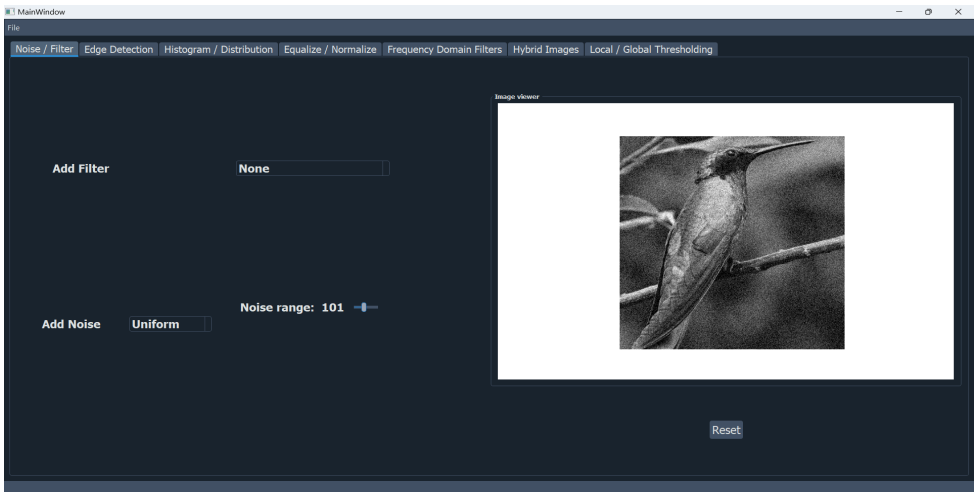
The `read_image` function prompts the user to select an image file, reads the selected image using OpenCV, and displays it in the application.

Additive Noise

Add various types of noise (Uniform, Gaussian, Salt & Pepper) to the images.

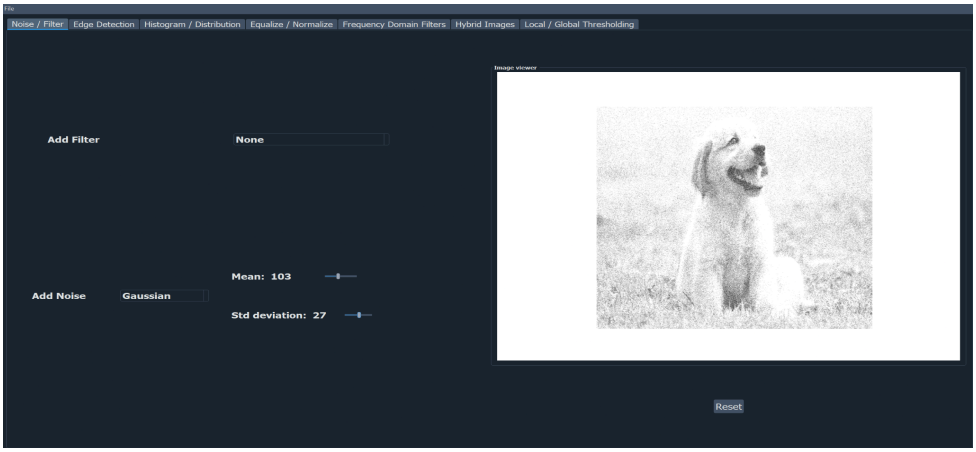
Uniform Noise

The `add_uniform_noise` function generates uniform noise within a specified range and adds it to a copy of the original image. Finally, it clips the resulting noisy image to the range [0, 255] and returns it.



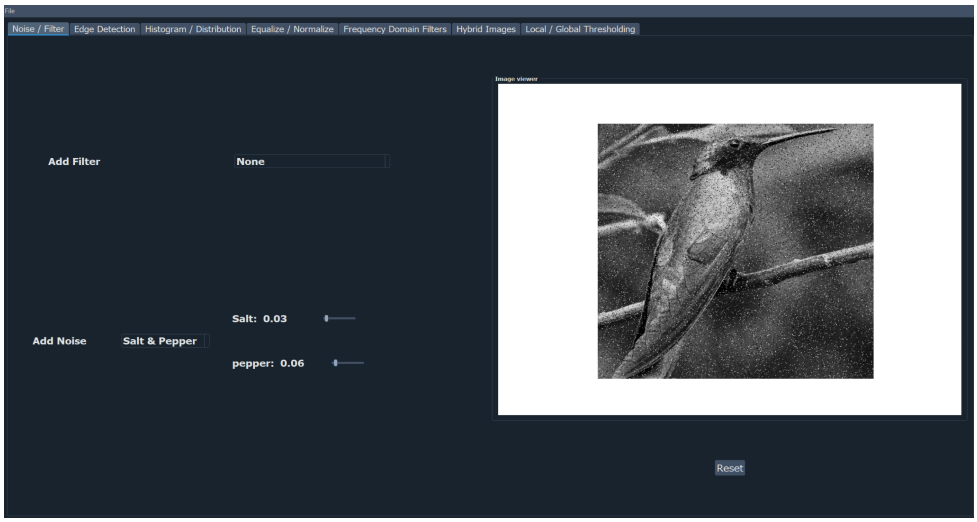
Gaussian Noise

The `add_gaussian_noise` function generates Gaussian noise with a specified mean and standard deviation, then adds it to a copy of the original image. It clips the resulting noisy image to the range [0, 255] and returns it.



Salt & Pepper Noise

The `add_salt_and_pepper_noise` function creates salt and pepper noise by randomly assigning white (salt) and black (pepper) pixels to a copy of the original image based on specified probabilities. It returns the resulting noisy image.

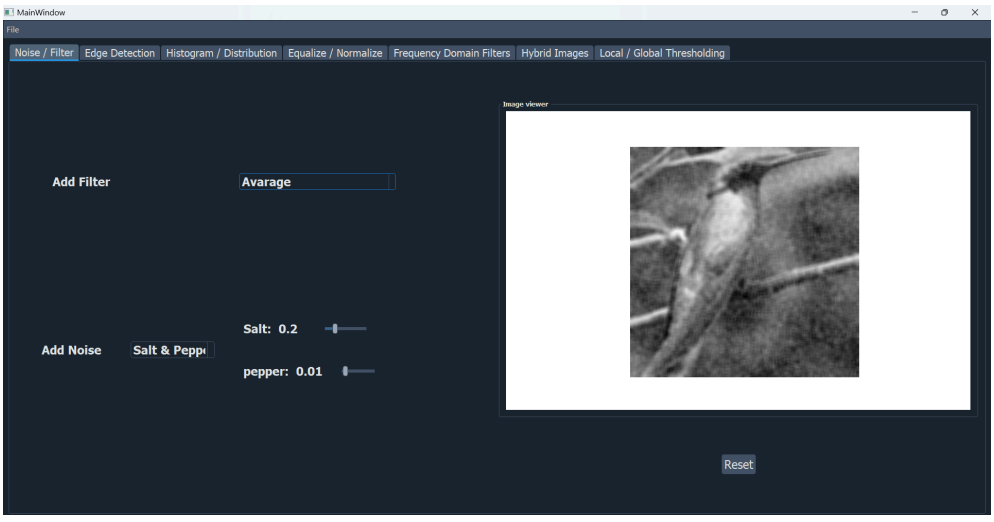


Filtering Noisy Images

Apply low-pass filters (Average, Gaussian, Median) to filter noisy images.

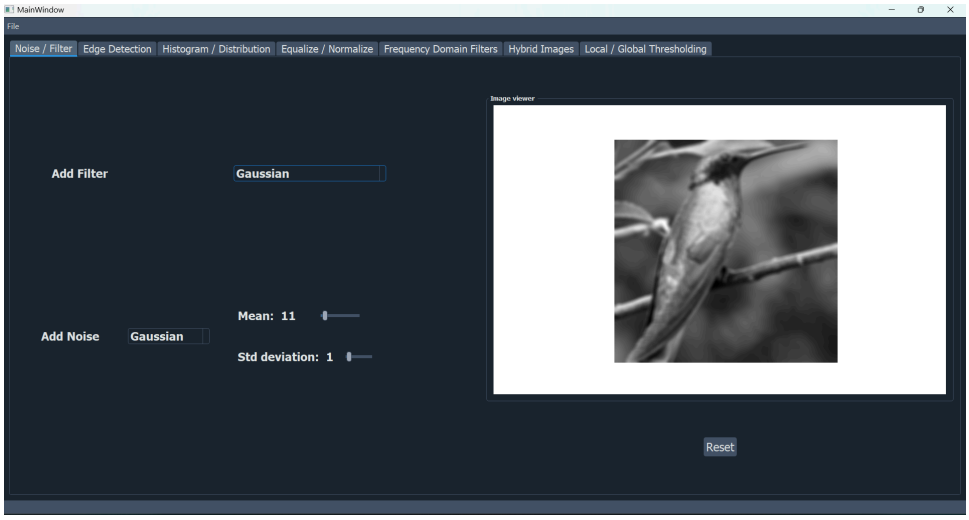
Average Filter

The `remove_average_noise` function employs an average filtering technique using an 11x11 kernel matrix to mitigate noise in the image. By convolving the kernel with the noisy image, it calculates the average intensity value within the local neighborhood of each pixel. The resulting smoothed image is returned, reducing noise and enhancing image clarity.



Gaussian Filter

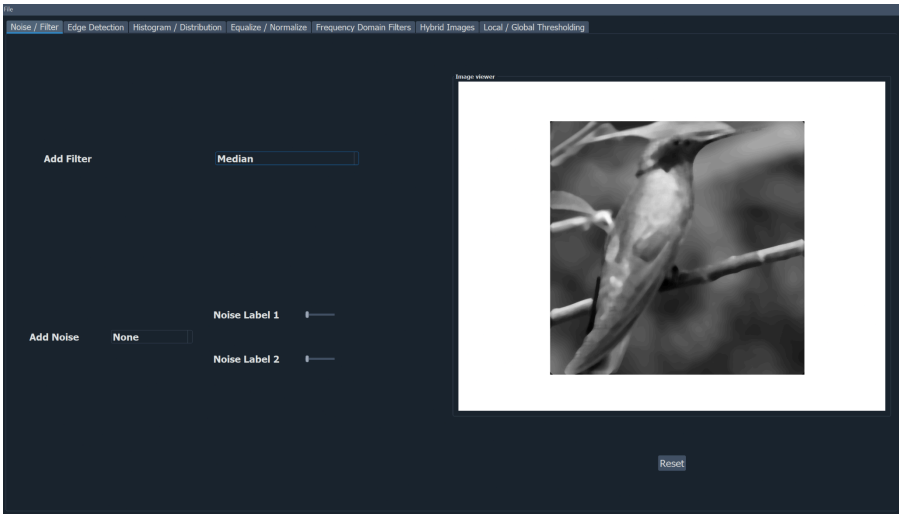
The `remove_gaussian_noise` function constructs an 11x11 Gaussian filter kernel matrix to reduce Gaussian noise in the image. This matrix is generated based on specified Gaussian parameters, and it is then applied to the noisy image using a convolution operation. The resulting filtered image exhibits reduced Gaussian noise, enhancing image quality.



Median Filter

The `remove_median_noise` function aims to remove noise from the image using a median filter. Here's a concise summary of the algorithm:

1. Padding: The noisy image is padded with zeros to handle edge cases, ensuring that the kernel can be applied to all pixels.
2. Kernel Application: For each pixel in the original image, a kernel of size 11x11 centered around the pixel is extracted from the padded image.
3. Median Calculation: The median value of the pixel intensities within the kernel is computed, and this value becomes the new intensity value of the corresponding pixel in the filtered image.
4. Output Image: The resulting filtered image with reduced noise is returned as the output of the function.

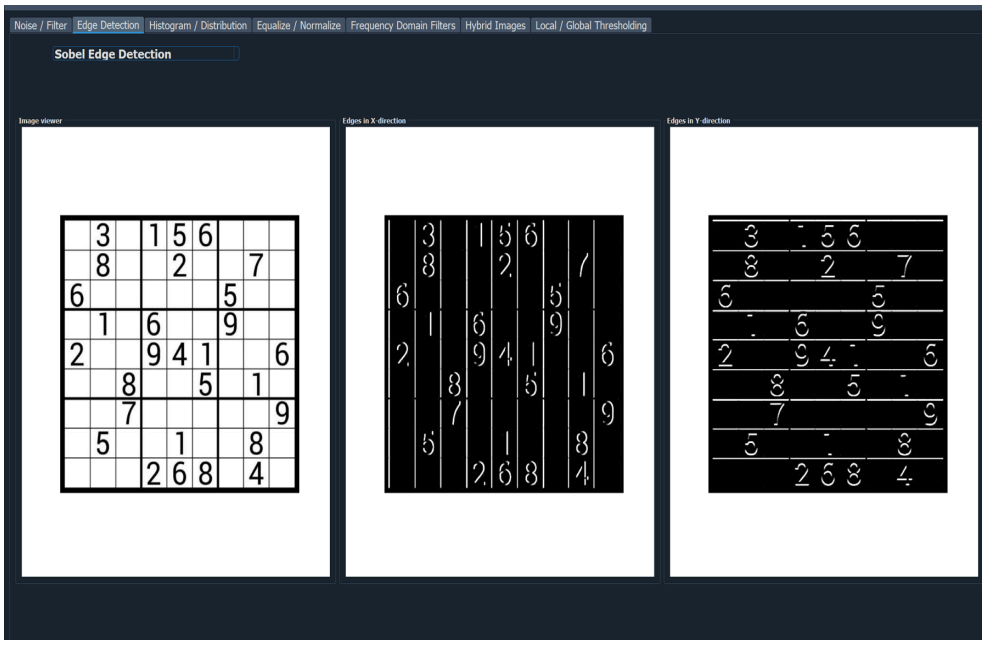


Edge Detection

Detects edges using different masks (Sobel, Roberts, Prewitt, Canny).

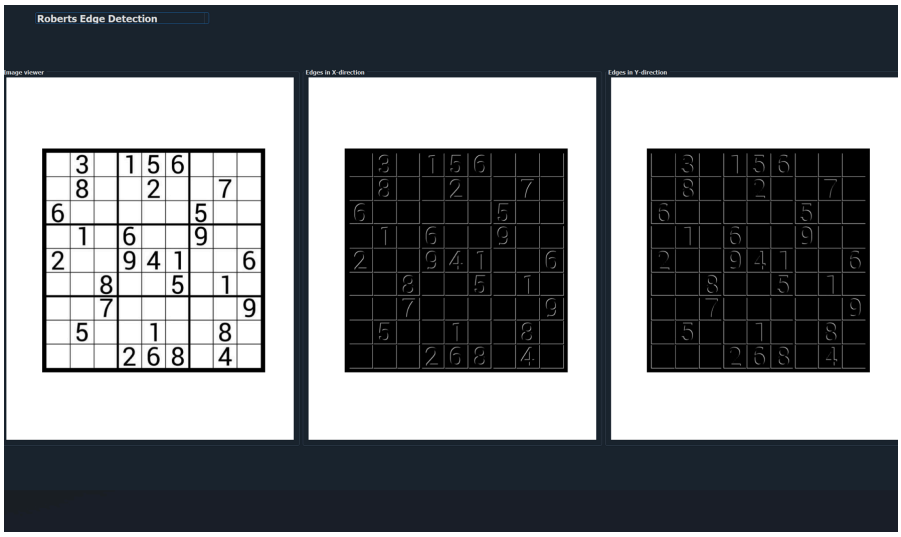
Sobel edge detection

The Sobel edge detection method is a widely used technique for detecting edges in digital images. It operates by convolving the image with a pair of 3x3 kernel matrices to calculate the gradient magnitude and direction at each pixel. The gradient magnitude represents the rate of change of intensity, while the gradient direction indicates the direction of the steepest ascent in intensity. By thresholding the gradient magnitude, edges can be detected effectively.

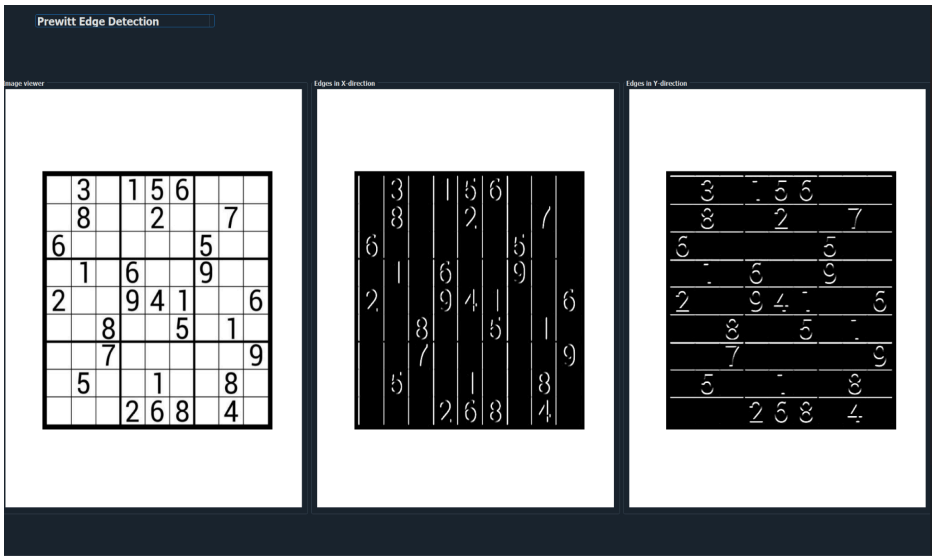


Robert edge detection

The Roberts edge detection method is one of the simplest edge detection techniques. It employs a pair of 2x2 kernel matrices to compute the gradient approximation in the horizontal and vertical directions. The gradient magnitude is obtained by combining the absolute values of these gradients. While Roberts edge detection is computationally efficient, it may produce less accurate results compared to other methods.



Prewitt edge detectionSimilar to Sobel edge detection, the Prewitt edge detection method calculates the gradient magnitude and direction using convolution with 3x3 kernel matrices. However, Prewitt kernels are designed to approximate the gradient better along the diagonal directions. This makes Prewitt edge detection more robust to diagonal edges compared to Sobel.



Canny edge detection

The Canny edge detection algorithm is a multi-stage process that aims to detect edges accurately while minimizing noise and false positives. It involves steps such as Gaussian blurring to reduce noise, gradient calculation using Sobel or Prewitt operators, non-maximum suppression to thin edges, and hysteresis thresholding to detect and link edges. Canny edge detection is widely used in computer vision applications due to its high accuracy and reliability.

Histogram Analysis

Draw histograms and distribution curves of images

Histogram analysis is a fundamental technique in image processing used to visualize the distribution of pixel intensities within an image. By plotting a histogram, which represents the frequency of occurrence of each intensity level, we gain insights into the overall brightness and contrast characteristics of the image. Additionally, distribution curves can be overlaid on histograms to provide further information about the distribution shape and to aid in image enhancement and analysis.

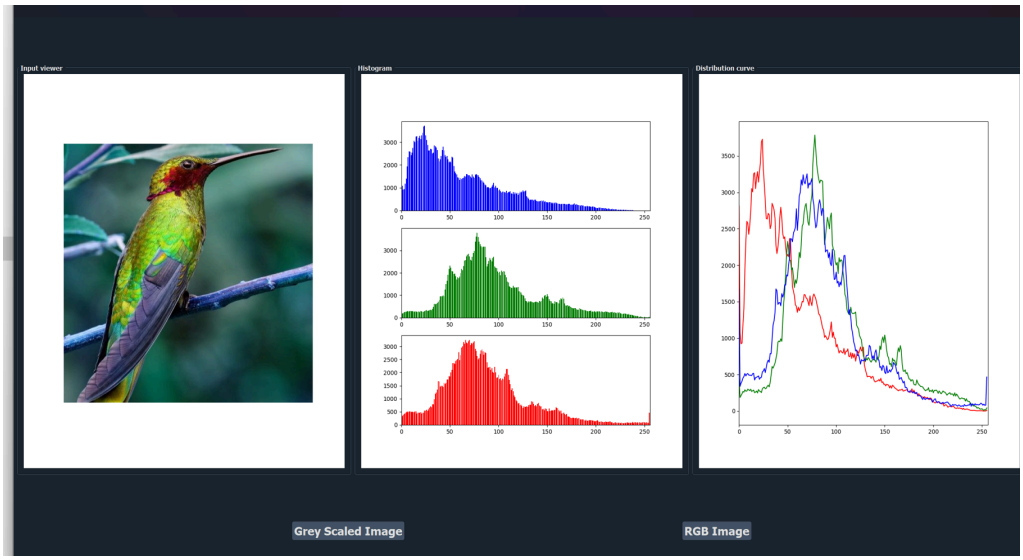


Image Equalization and Normalization

Equalization Algorithm

Histogram equalization is a technique that spreads out the pixel intensities in an image, resulting in a more evenly distributed histogram. This process enhances the contrast of the image by effectively redistributing the intensity values. The algorithm for histogram equalization involves the following steps:

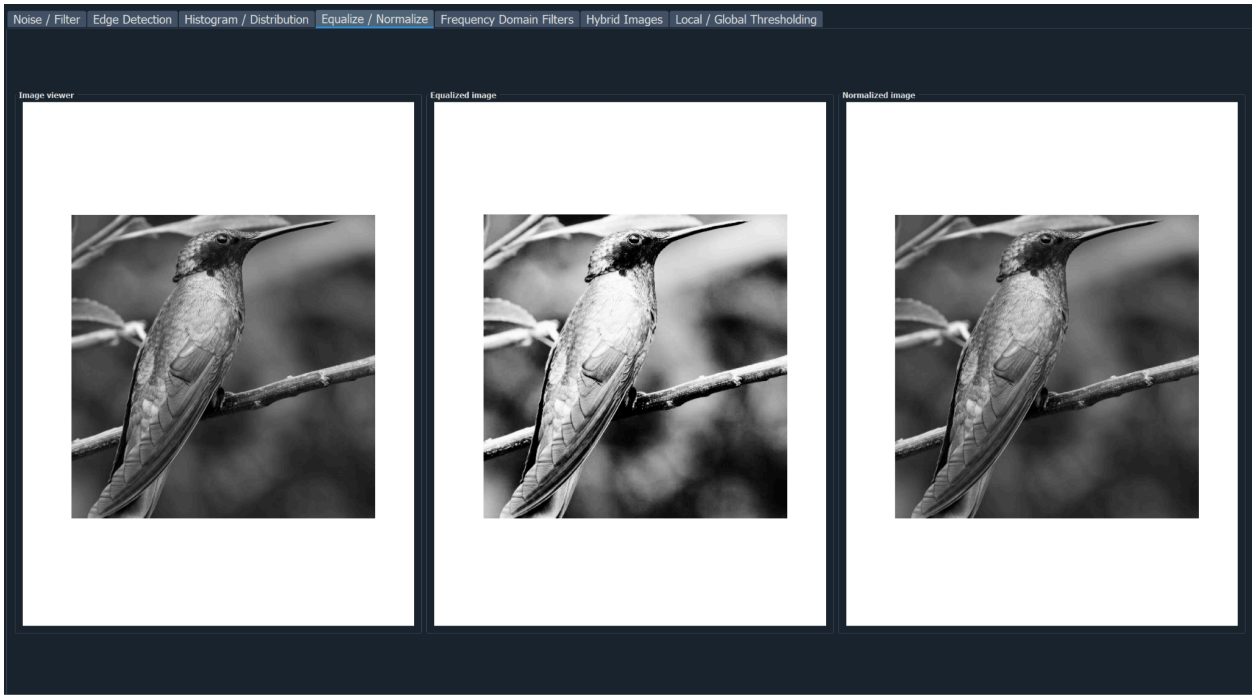
1. Compute Histogram: Calculate the histogram of the input image to obtain the distribution of pixel intensities.
2. Compute Cumulative Distribution Function (CDF): Compute the cumulative sum of the histogram values to obtain the CDF.
3. Normalize CDF: Normalize the CDF to the desired range, typically [0, 255].
4. Map Pixel Intensities: Map the original pixel intensities to their corresponding values in the normalized CDF, effectively equalizing the histogram.

5. Output Equalized Image: Generate the equalized image with enhanced contrast.

Normalization Algorithm

Normalization is the process of scaling the pixel values of an image to a specific range, typically [0, 255] for 8-bit images. This technique ensures that the pixel intensities are scaled proportionally within the desired range, facilitating better visualization and analysis. The algorithm for normalization involves the following steps:

1. Find Minimum and Maximum Values: Determine the minimum and maximum pixel intensities in the input image.
2. Scale Pixel Values: Scale the pixel values linearly to the desired range
3. Output Normalized Image: Generate the normalized image with pixel values scaled to the desired range.



Local & Global Thresholding:

Thresholding is a fundamental technique in image processing used to segment an image into regions based on pixel intensity. Local and global thresholding methods are commonly employed to adaptively determine the threshold value for each pixel based on its local neighborhood or the entire image.

Global Thresholding:

Global thresholding is a simple technique where a single threshold value is applied to the entire image. Pixels with intensities higher than the threshold value are assigned to one class (e.g., foreground), while pixels with intensities lower than the threshold value are assigned to another class (e.g., background). The algorithm for global thresholding involves the following steps:

1. Select Threshold Value: Choose a threshold value based on the desired segmentation outcome or by analyzing the histogram of the image.
2. Thresholding Operation: Iterate over each pixel in the image and compare its intensity value with the chosen threshold. Assign pixels to either the foreground or background class based on the threshold value.
3. Output Result: Generate a binary image where pixels above the threshold are set to a predefined maximum value (e.g., 255) and pixels below the threshold are set to zero.

Local Thresholding:

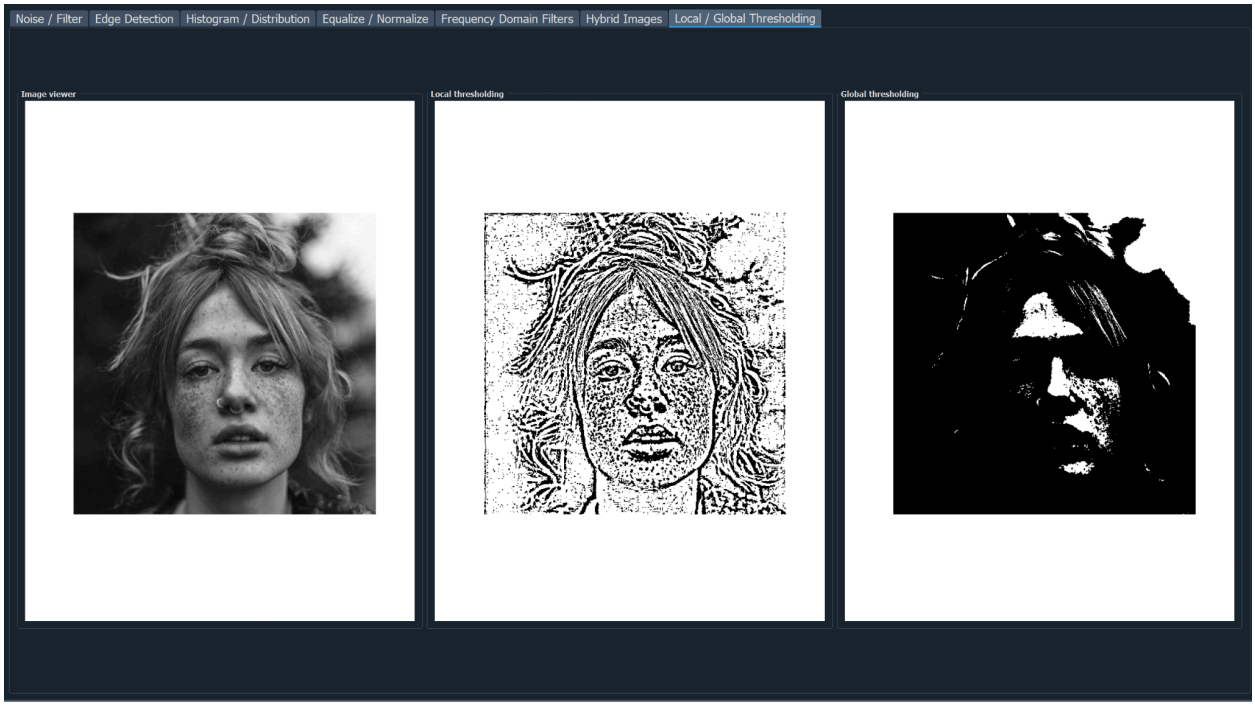
Local thresholding, also known as adaptive thresholding, addresses the limitations of global thresholding by considering local neighborhoods around each pixel. This approach allows for more robust segmentation, especially in images with varying illumination conditions or uneven backgrounds. The algorithm for local thresholding involves the following steps:

1. Define Local Neighborhood: Specify a window or kernel size to define the local region around each pixel.

2. Compute Local Threshold: For each pixel, calculate a local threshold value based on the statistics (e.g., mean, median) of the pixel intensities within the local neighborhood.
3. Apply Thresholding: Compare the intensity of each pixel with its corresponding local threshold value. Assign pixels to the foreground or background class based on this local threshold.
4. Output Result: Generate a binary image where pixels above the local threshold are set to a predefined maximum value, and pixels below the threshold are set to zero.

Parameters:

1. Global Thresholding: The main parameter in global thresholding is the threshold value itself, which determines the separation between the foreground and background regions. This value can be chosen manually based on prior knowledge of the image characteristics or automatically determined using histogram analysis techniques.
2. Local Thresholding: In addition to the threshold value, local thresholding methods typically involve parameters such as the size of the local neighborhood (block size) and a constant factor (c) used in computing the local threshold. These parameters influence the sensitivity of the thresholding operation and may require tuning based on the image content and application requirements.



Frequency Domain Filters:

Frequency domain filtering is a powerful technique in image processing that operates on the frequency components of an image rather than directly on the spatial domain pixel values. By transforming an image into the frequency domain using techniques like the Discrete Fourier Transform (DFT), we can manipulate its frequency components to achieve various filtering operations such as low-pass and high-pass filtering.

Discrete Fourier Transform (DFT):

The Discrete Fourier Transform (DFT) is a mathematical operation that transforms an image from the spatial domain to the frequency domain. It represents the image as a combination of sinusoidal waves of different frequencies and amplitudes. In the frequency domain, high-frequency components correspond to rapid changes in pixel values, while low-frequency components correspond to smooth transitions.

Low-pass Filtering

Low-pass filtering is a frequency domain filtering technique that attenuates high-frequency components of an image while preserving low-frequency components. This filter is commonly used for smoothing or blurring an image and removing noise. The algorithm for low-pass filtering involves the following steps:

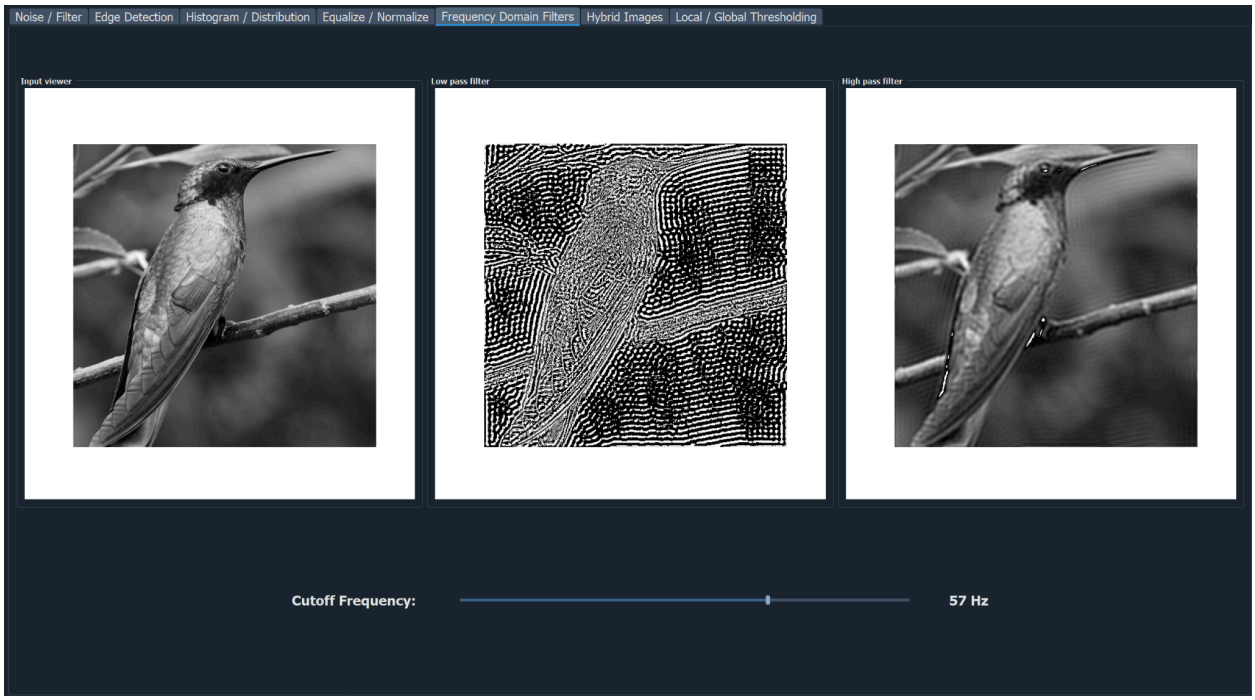
1. Compute DFT: Apply the DFT to the input image to obtain its frequency spectrum.
2. Design Filter Mask: Create a filter mask that attenuates high-frequency components while preserving low-frequency components. This mask is typically defined by a cutoff frequency that separates the high and low-frequency regions.
3. Apply Filter: Multiply the frequency spectrum of the image by the filter mask to perform low-pass filtering.
4. Inverse DFT: Transform the filtered frequency spectrum back to the spatial domain using the inverse DFT to obtain the filtered image.

High-pass Filtering:

High-pass filtering is the opposite of low-pass filtering; it attenuates low-frequency components while preserving high-frequency components. This filter is useful for enhancing edges and features in an image. The algorithm for high-pass filtering involves similar steps to low-pass filtering but with a filter mask designed to preserve high-frequency components.

Parameters:

Cutoff Frequency: The cutoff frequency is a critical parameter in frequency domain filtering that determines the separation between high and low-frequency components. It influences the degree of smoothing or sharpening applied to the image and should be chosen based on the desired filtering effect and image characteristics.



Hybrid Images

Hybrid images are created by combining the low-frequency content of one image with the high-frequency content of another image. This process exploits the characteristics of the human visual system, which perceives low-frequency information at a distance and high-frequency information up close.

Construction Algorithm

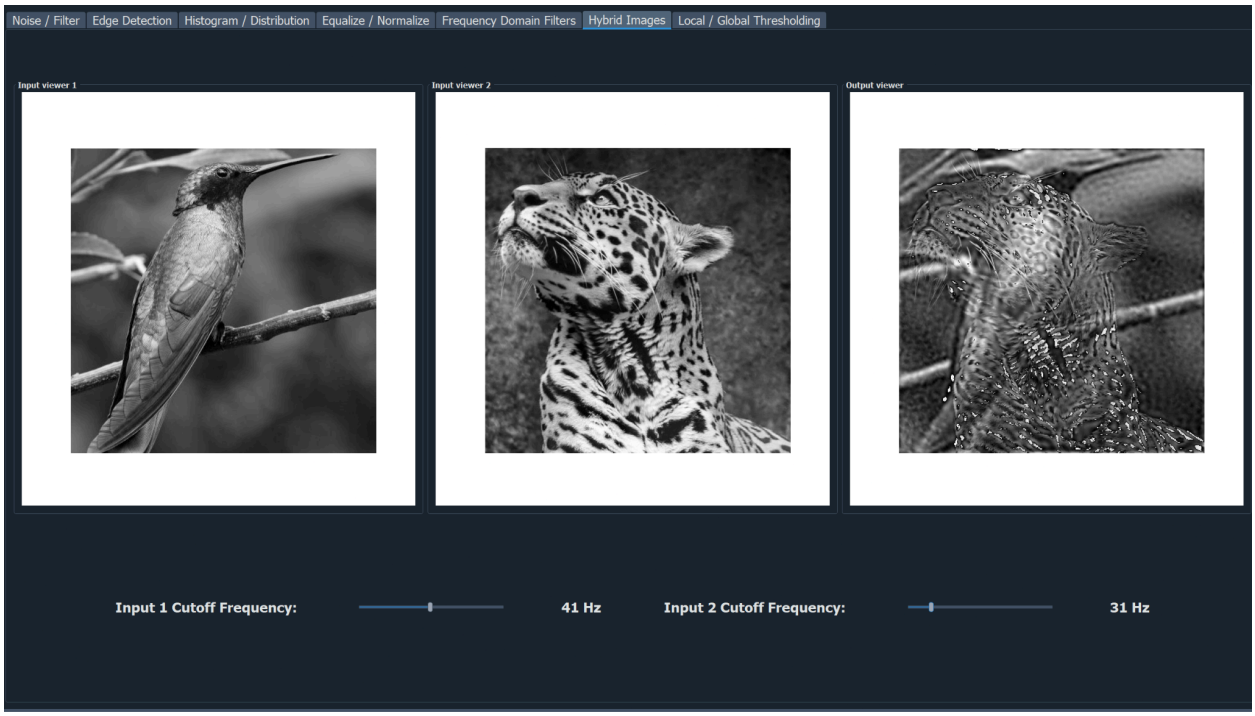
The construction of hybrid images involves the following steps:

1. Select Input Images: Choose two input images with distinct spatial frequency characteristics. Typically, one image contains fine details and high-frequency content (e.g., edges, textures), while the other image contains smooth gradients and low-frequency content (e.g., background).

2. Frequency Filtering: Apply frequency domain filtering to each input image. Use a low-pass filter to extract the low-frequency components from one image and a high-pass filter to extract the high-frequency components from the other image.
3. Combination: Combine the filtered images by adding or blending them together. This creates a new image where the low-frequency components are derived from one image and the high-frequency components from the other.
4. Display Result: Display the resulting hybrid image. When viewed from a distance, the low-frequency content dominates, producing one interpretation of the image. However, when viewed up close, the high-frequency details become more apparent, revealing a different interpretation.

Parameters:

1. Cutoff Frequencies: The cutoff frequencies determine the separation between low and high-frequency components in each input image. These frequencies control the balance between the low and high-frequency content in the hybrid image. Higher cutoff frequencies result in more emphasis on high-frequency details, while lower cutoff frequencies prioritize low-frequency features.
2. Filter Types: The choice of filter types, such as low-pass and high-pass filters, influences the frequency content extracted from each input image. Low-pass filters preserve low-frequency information, while high-pass filters highlight high-frequency details. Adjusting the parameters of these filters can alter the visual appearance and perceptual effects of the hybrid image.



Team Members:

1. - Mohamed Sayed Mosilhe
2. - Mina Adel
3. - Mariam Magdy
4. - Ali Maged
5. - Abdallah Ahmed