Computer Vision

Edge Detector & Active Contour

Task 2 - Team 11

Introduction

This report delves into the implementation of edge and boundary detection techniques, focusing on the utilization of the Canny edge detector, the Hough transform, and Active Contour Models (ACM), commonly known as snakes. The primary objectives encompass detecting edges, lines, circles, and ellipses within given images and evolving Active Contour Models to delineate and quantify objects of interest.

It involves detecting edges using the Canny edge detector and identifying shapes such as lines, circles, and ellipses using the Hough transform. These techniques will be applied to images, with the detected shapes superimposed onto the images, facilitating visual inspection and analysis.

In Snakes, We initialize contours around given objects and employ the Active Contour Model (snake) algorithm to iteratively evolve these contours. Using the greedy algorithm, we track the contour evolution, representing the output as chain code. Additionally, we compute essential metrics such as perimeter and area inside these contours, providing quantitative insights into the segmented objects.

Through the execution of these tasks, this report aims to showcase the effectiveness and versatility of edge and boundary detection techniques in computer vision. The analysis will not only demonstrate the practical utility of these methods but also highlight their limitations and potential areas for improvement. Ultimately, the findings presented herein contribute to the broader understanding of image processing and its applications in various domains.

Features

Canny Edge Detection:

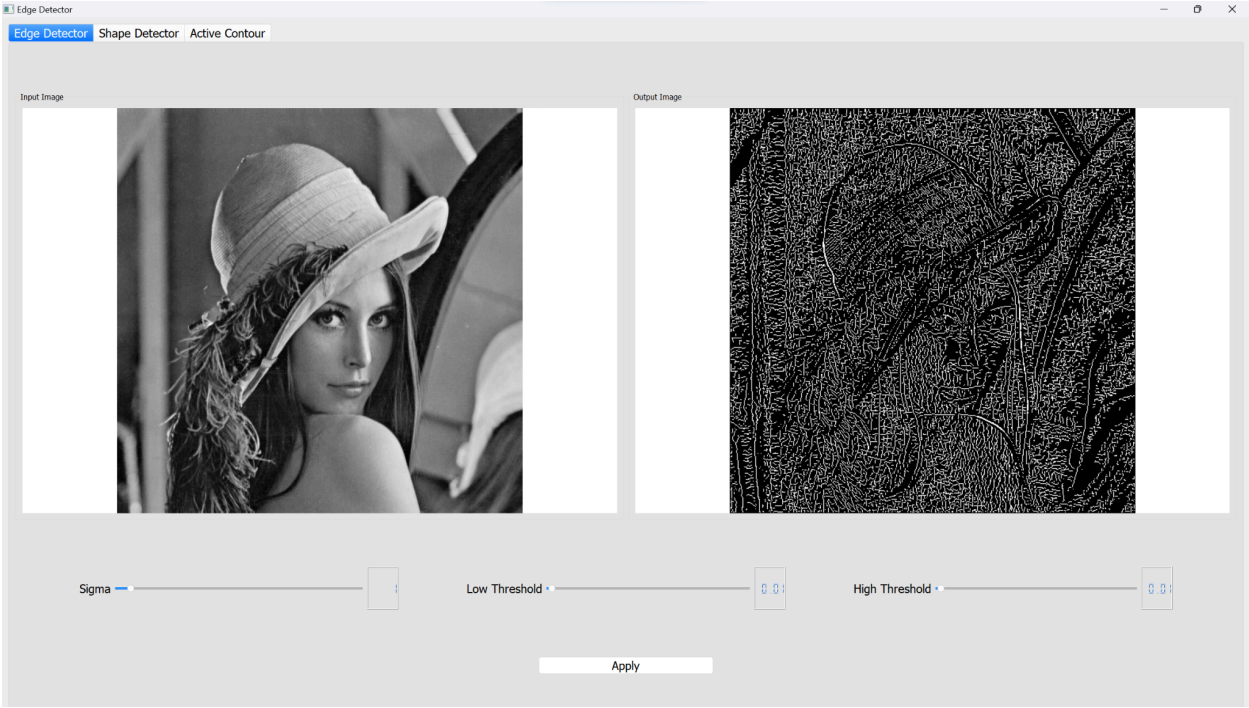
Parameters

Low threshold: The low threshold is used to identify weak edges. Weak edges are edges that are not as strong as strong edges, but they are still significant enough to be included in the final image. The low threshold is typically set to a value that is a fraction of the high threshold.

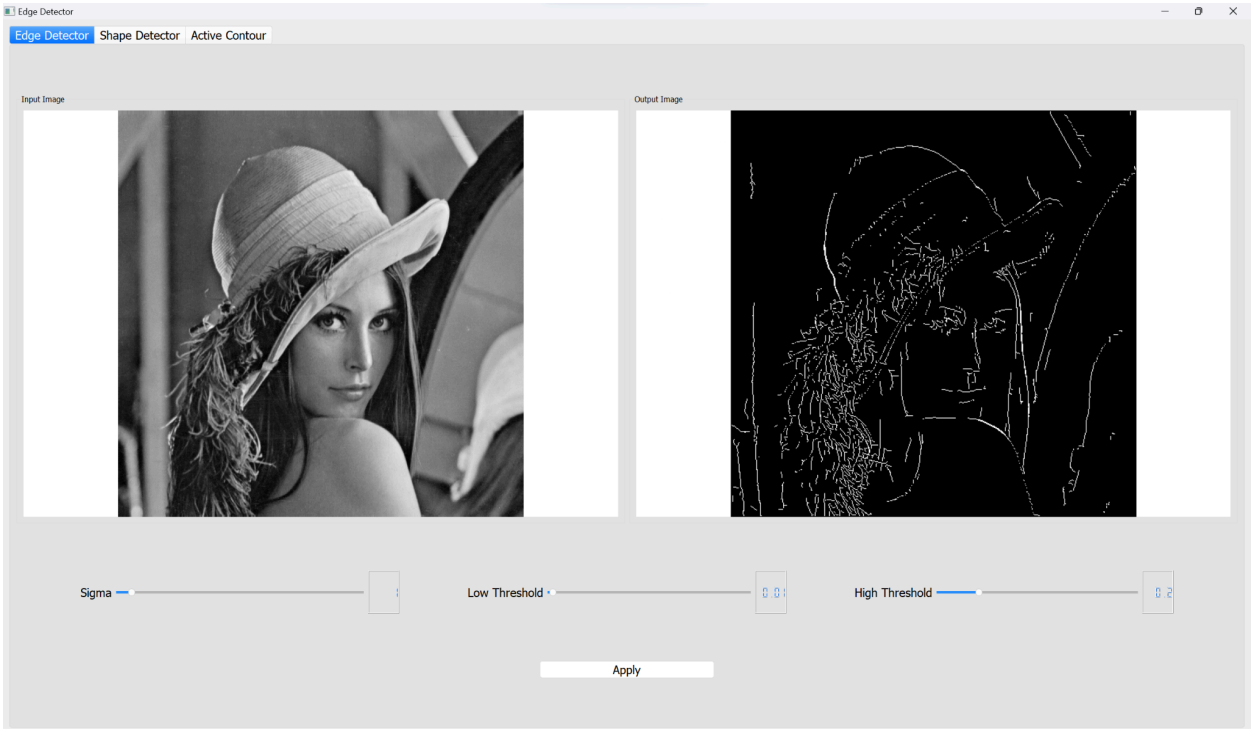
High threshold: The high threshold is used to identify strong edges. Strong edges are edges that are very pronounced and easy to see. The high threshold is typically set to a value that is a multiple of the low threshold.

Effect of low and high threshold: The low and high thresholds control the sensitivity of the edge detection algorithm. A lower low threshold will result in more weak edges being detected, while a higher low threshold will result in fewer weak edges being detected. A higher high threshold will result in more strong edges being detected, while a lower high threshold will result in fewer strong edges being detected.

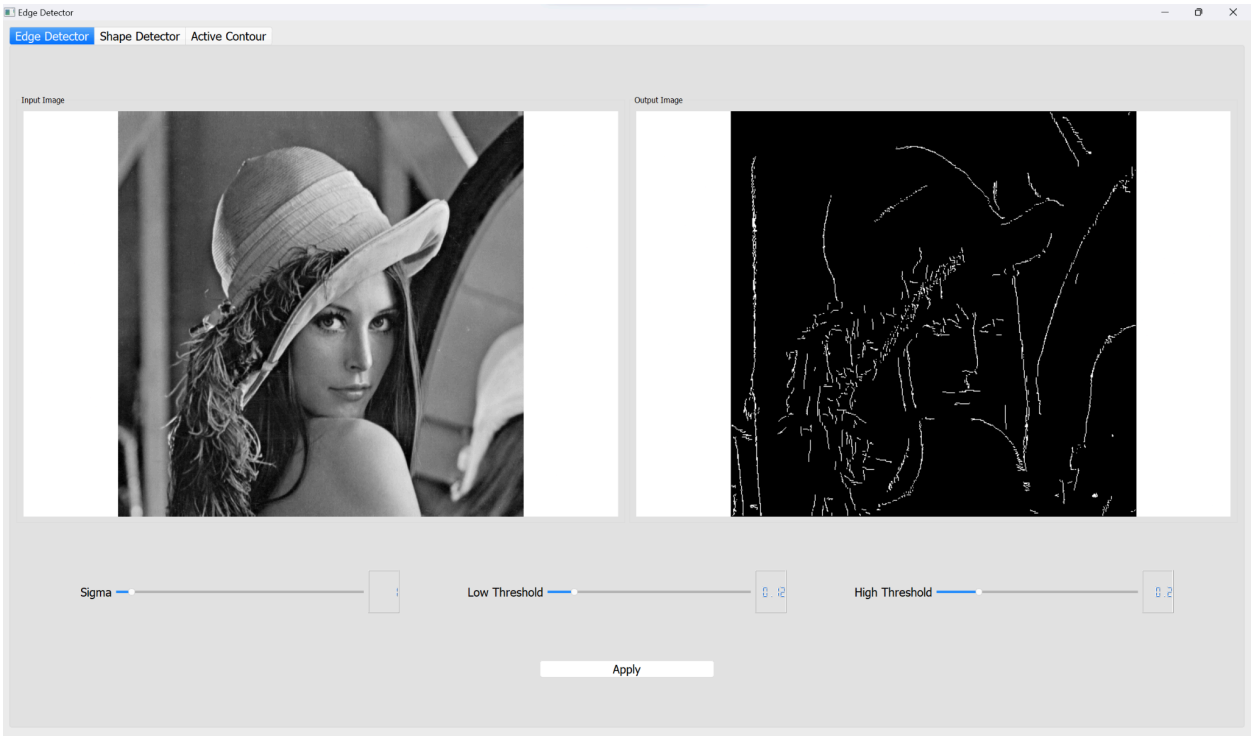
Next image shows when the low threshold is low and the high threshold is low



Next image shows when the low threshold is low and the high threshold is high



Next image shows when the low threshold is high and the high threshold is high



Smoothing the Image:

A higher value of sigma results in a more blurred/smoothed image.

Smoothing helps reduce noise in the image, making it easier to detect significant changes in intensity, which are indicative of edges.

Blur Radius:

The value of sigma determines the blur radius of the Gaussian filter.

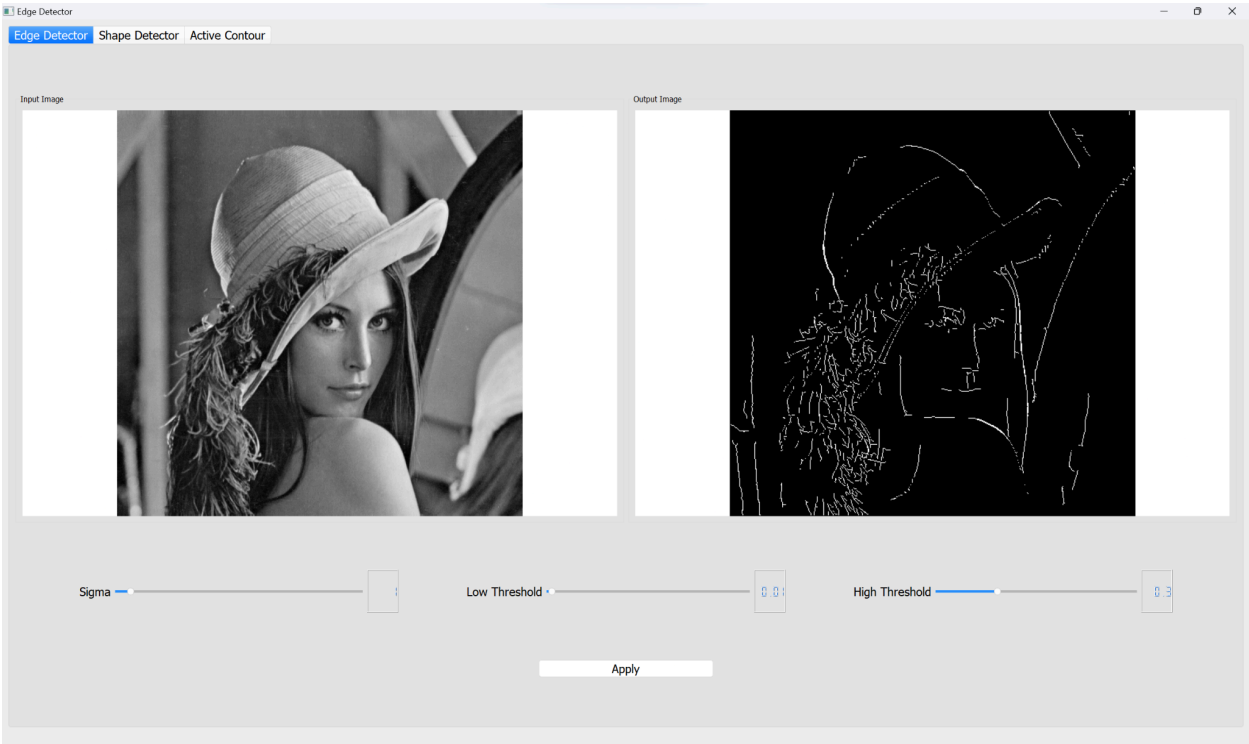
A larger sigma leads to a larger blur radius, which means more neighboring pixels are considered when calculating the smoothed value for each pixel.

Effect on Edge Detection:

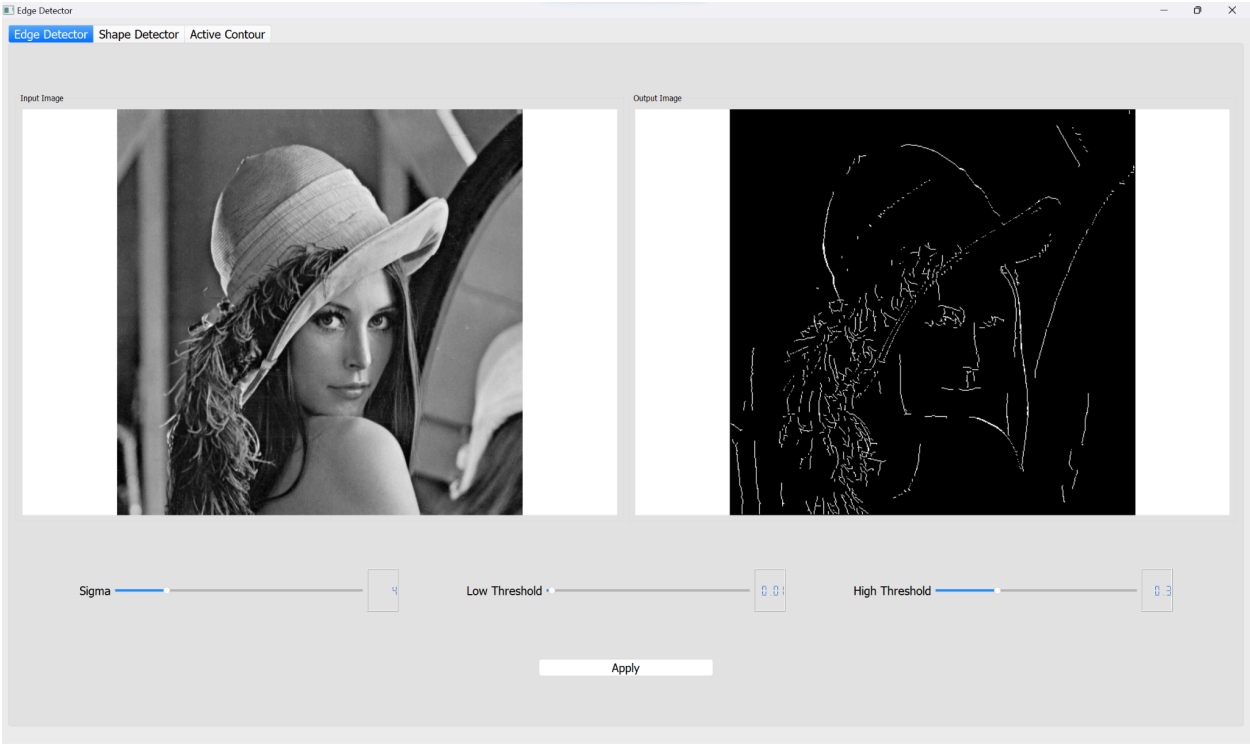
Increasing sigma can help in detecting broader edges or edges with gradual intensity changes.

However, if sigma is too large, fine details may be lost, and edges may become overly blurred, leading to less accurate edge detection.

Next image shows when the sigma is low



Next image shows when the sigma is high

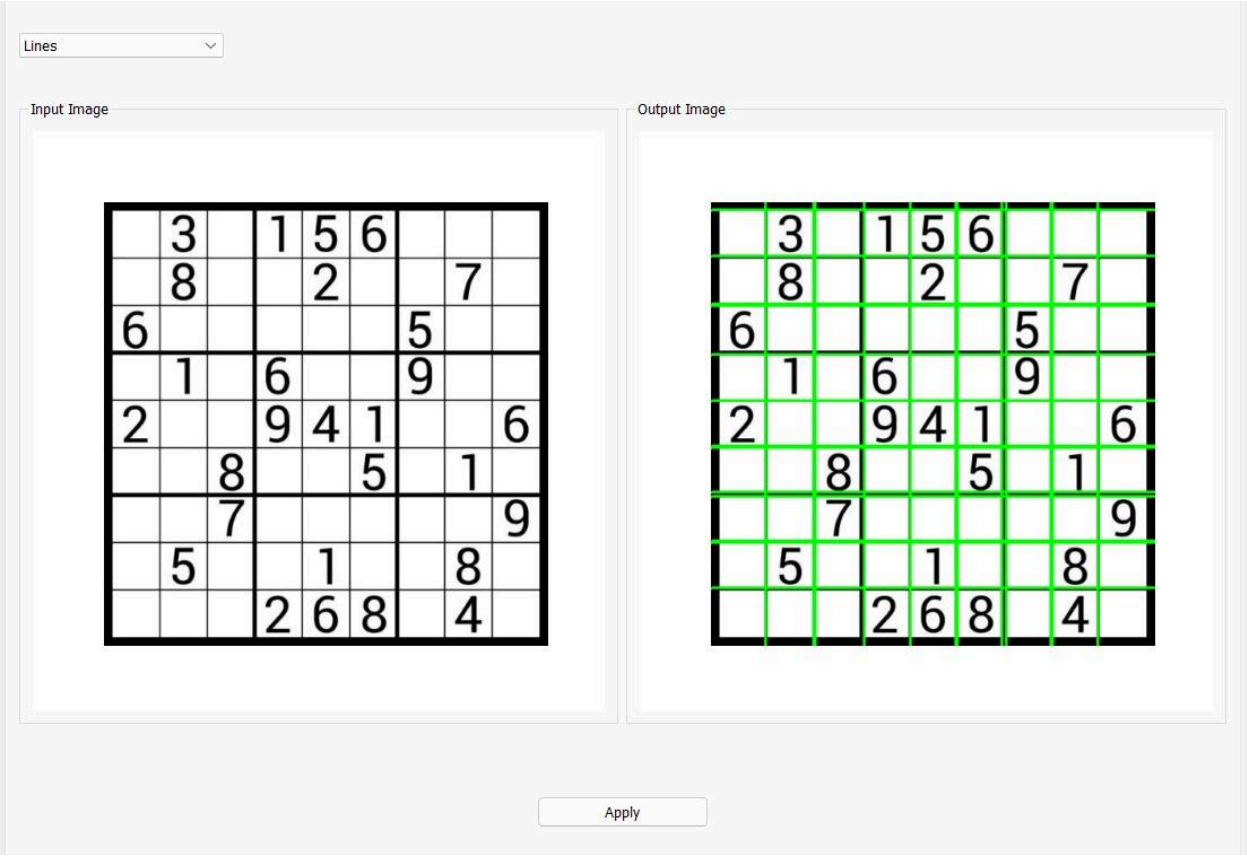


Hough Line Edge Detection

The Hough Line Transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the transform is to determine the parameters of lines present in an image that can be represented in mathematical form. It is particularly useful in scenarios where the image features are not perfectly linear or where they may be subject to noise or distortion.

Our implementation of the Hough Line Transform follows the standard approach of mapping points in the image space to a parameter space and then identifying lines as local maxima in the accumulator space. The algorithm involves the following key steps:
1. Edge Detection: The first stage is to reduce the input image to edge-only information. We used the Canny edge detector for this purpose, due to its ability to provide good edge connectivity and reduce the number of spurious responses to edge detection.

2. Mapping to Hough Space: Each edge point in the image is mapped to a sinusoidal curve in the Hough space, which is defined by the parameters (ρ, θ), representing the distance and angle from the origin to the closest point on the straight line.
3. Accumulator Array: An accumulator array is used to record the parameter values of potential lines. Each cell in the array represents a quantized parameter value of (ρ, θ). For each edge point and for each θ value, we calculate ρ and increment the corresponding accumulator cell.
4. Identifying Local Maxima: Lines are identified by finding local maxima in the accumulator array. The threshold for a local maximum is set to identify the most prominent lines in the image. Non-maximum suppression is also applied to avoid multiple responses to the same line.
5. Line Drawing: Once the prominent lines are identified in the Hough space, they are mapped back to the image space. The end points of these lines are calculated based on the size of the image, and the lines are drawn onto the original or a separate image for visualization.
Here is an example:



Active contour

Active contour models, also known as snakes, are popular tools in computer vision and image processing for object segmentation and boundary detection. These models evolve iteratively to fit object boundaries in images by minimizing an energy function, composed of internal and external energy terms. We present an implementation of active contouring using a greedy method. The method aims to find the optimal contour by iteratively adjusting its shape based on the local energy minimization.

Our implementation consists of several key components:

1. Initialization

- We initialize the contour by defining a circular shape around a specified center with a given radius. This initial contour serves as the starting point for the iterative optimization process.

2. Energy Functions

- We define three energy functions to guide the contour evolution:
    - Internal Energy: Captures the smoothness of the contour. It penalizes high curvature to maintain smoothness.
    - External Energy: Measures the image gradient along the contour. It attracts the contour towards object boundaries, where the gradient is strong.
    - Balloon Energy: Controls the inflation or deflation of the contour. It helps the contour adapt to object shapes by expanding or contracting it.

3. Contour Updating

- The main iterative process involves updating the contour to minimize the total energy. At each iteration, we consider moving each point in the contour to its neighboring positions and calculate the corresponding energy. The point with the minimum total energy is chosen as the new position for that iteration.

Results

To assess the impact of user-specified parameters on the active contouring algorithm, we conducted experiments by varying the values of alpha, beta, gamma, the number of iterations, center coordinates, and radius. Each parameter plays a crucial role in shaping the behavior and performance of the algorithm.

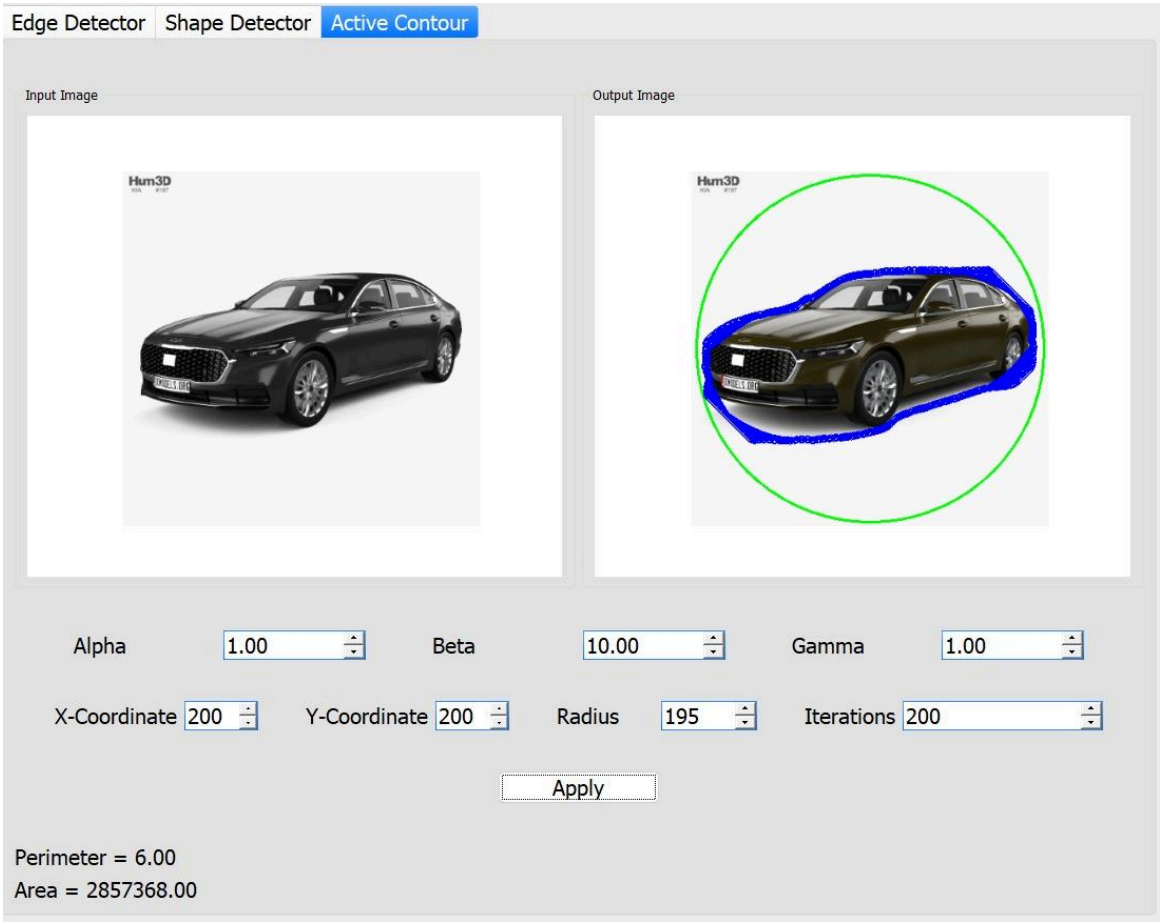
1. Effect of Alpha, Beta, and Gamma

- Alpha: This parameter controls the influence of internal energy, which maintains the smoothness of the contour. Higher values of alpha enforce smoother contours by penalizing high curvature. Lower values, on the other hand, allow the contour to adapt more flexibly to object boundaries but may result in jagged contours.
- Beta: Beta regulates the impact of external energy, which attracts the contour towards object boundaries based on image gradients. Increasing beta strengthens the attraction force towards edges, leading to tighter and more accurate contours. Conversely, lower values of beta reduce the influence of image gradients, resulting in smoother but potentially inaccurate contours.
- Gamma: Gamma governs the balloon energy, controlling the inflation or deflation of the contour. Higher values of gamma encourage expansion, allowing the contour to encompass larger regions. Lower values promote contraction, restricting the contour to smaller areas. Adjusting gamma is particularly useful for adapting the contour to objects with varying shapes and sizes.

2. Effect of Number of Iterations

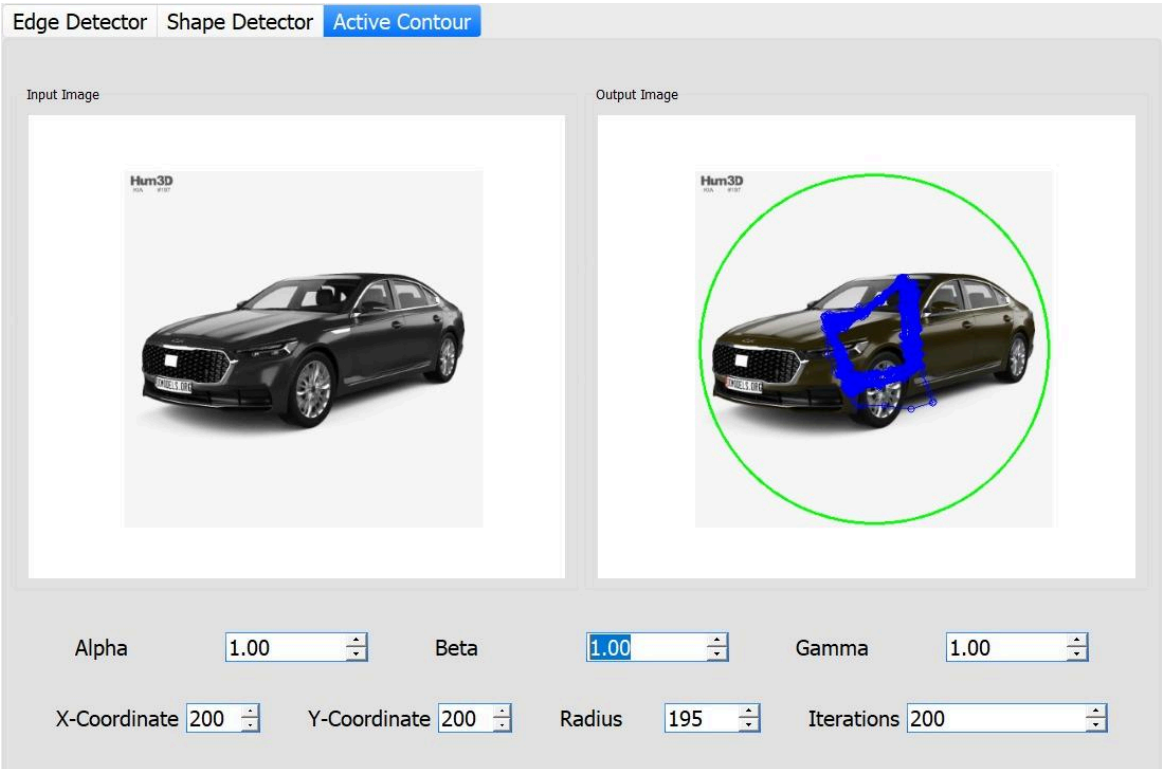
The number of iterations determines the duration of the optimization process. Increasing the number of iterations allows the algorithm to refine the contour further, potentially leading to better convergence and accuracy. However, a higher number of iterations also increase computational time. Conversely, reducing the number of iterations may speed up the process but could result in premature convergence and suboptimal contours, especially in complex images or with insufficient initialization.

Best Test Results :

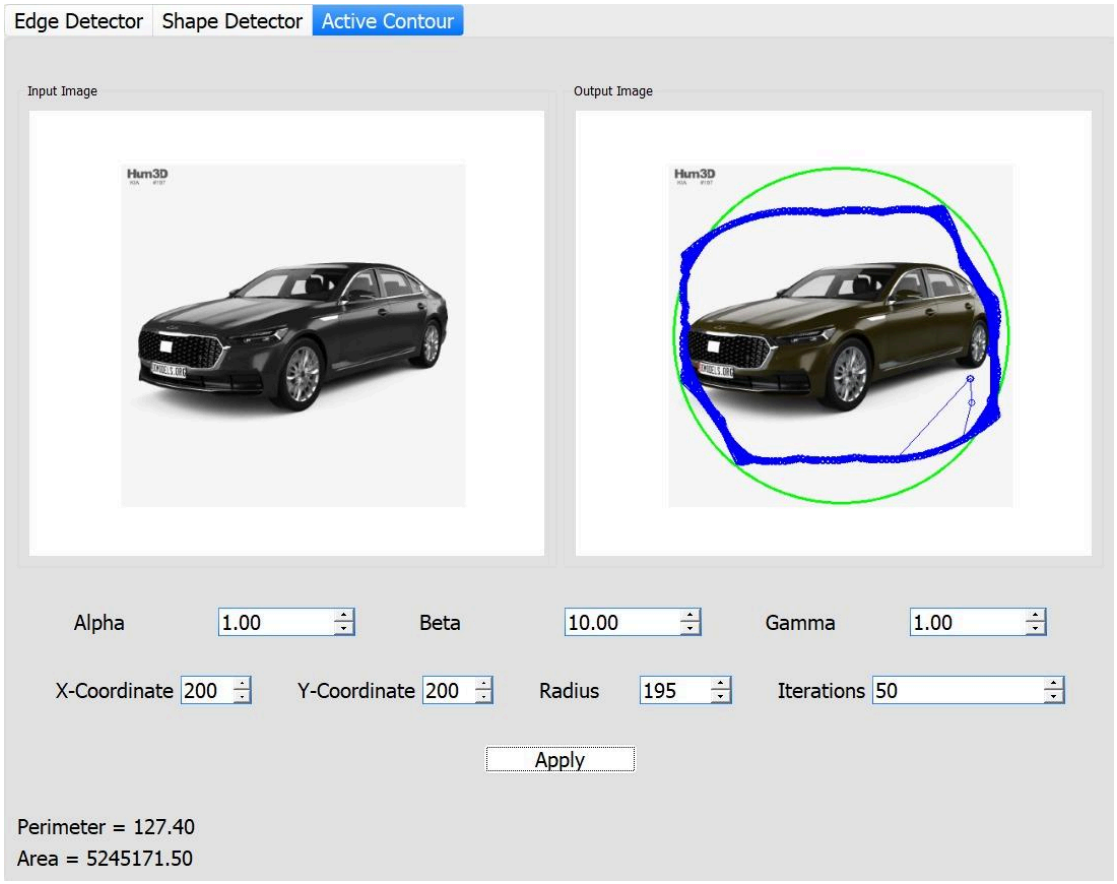


Changing Parameter : 1. Low Beta

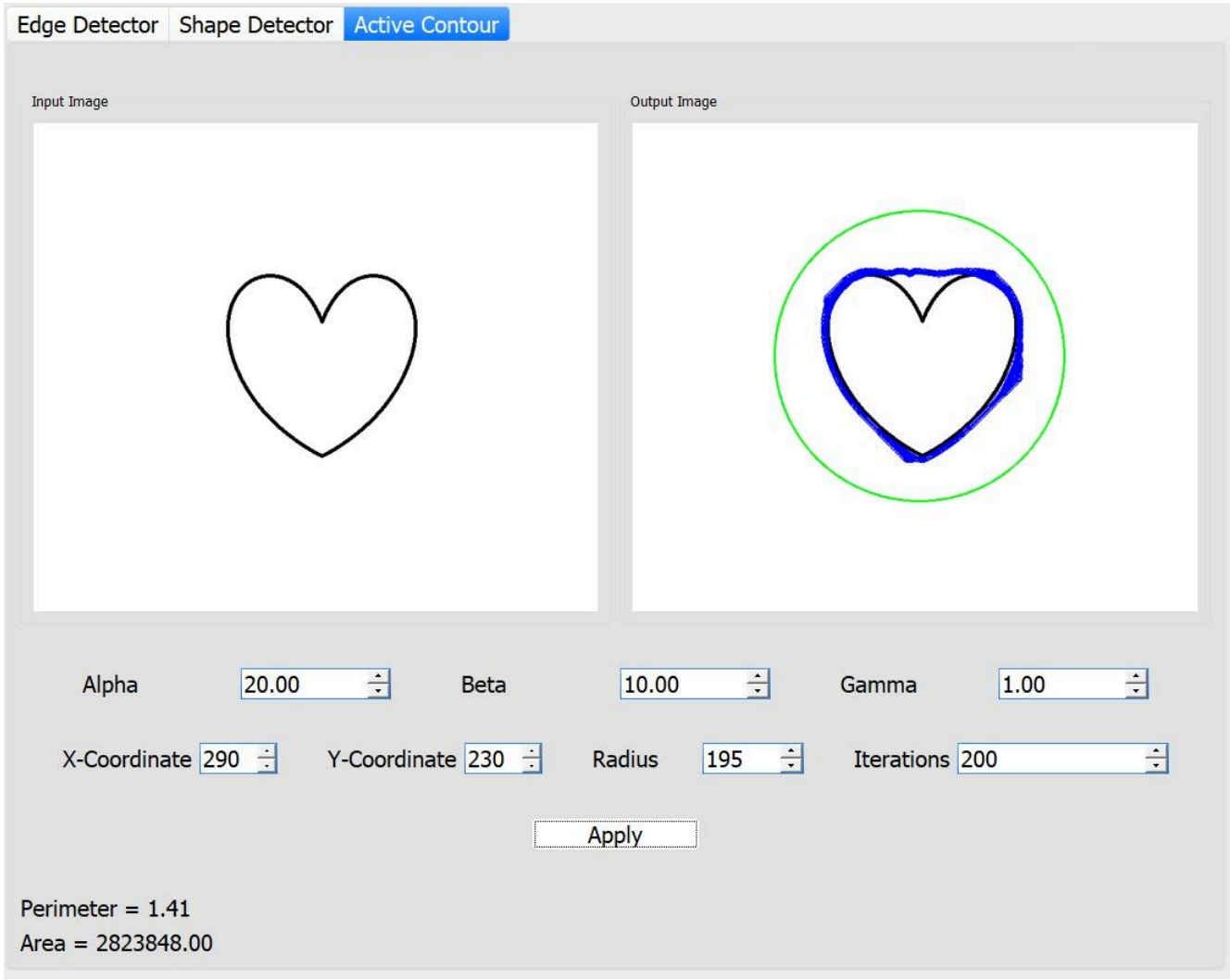
2.



2. Low number of Iterations



Other Test Results :



Circle Detection using Hough Transform

Description:

This feature enables the detection of circles within an input image using the Hough Transform technique. It identifies circular objects or patterns within images, providing valuable information for various computer vision applications.

Functionality:

1. Input Image Handling:
    - Accepts a numpy array representing the input image.
    - Ensures that modifications are made to a copy of the original image to preserve its integrity.
2. Preprocessing:
    - Converts the input image to grayscale if it's in color (BGR format) as many edge detection algorithms work on grayscale images.
    - Applies Gaussian blur to the grayscale image to reduce noise and smoothen edges.
    - Performs Canny edge detection on the smoothed image to identify potential edges.
3. Hough Transform for Circle Detection:
    - Initializes an accumulator array to detect circles of various radii.
    - Iterates over a range of radii, creating circle templates centered around edge points and updating the accumulator array based on their overlap.
    - Thresholds the accumulator array to retain potential circle centers based on a specified threshold.
    - Finds local maxima in the accumulator array to identify candidate circle centers.
4. 4. Circle Extraction and Drawing:
    - Extracts information about detected circles from the accumulator array.

- Draws circles on the original image using OpenCV's `cv2.circle` function, indicating the detected circles' positions and radii.

5. Output:

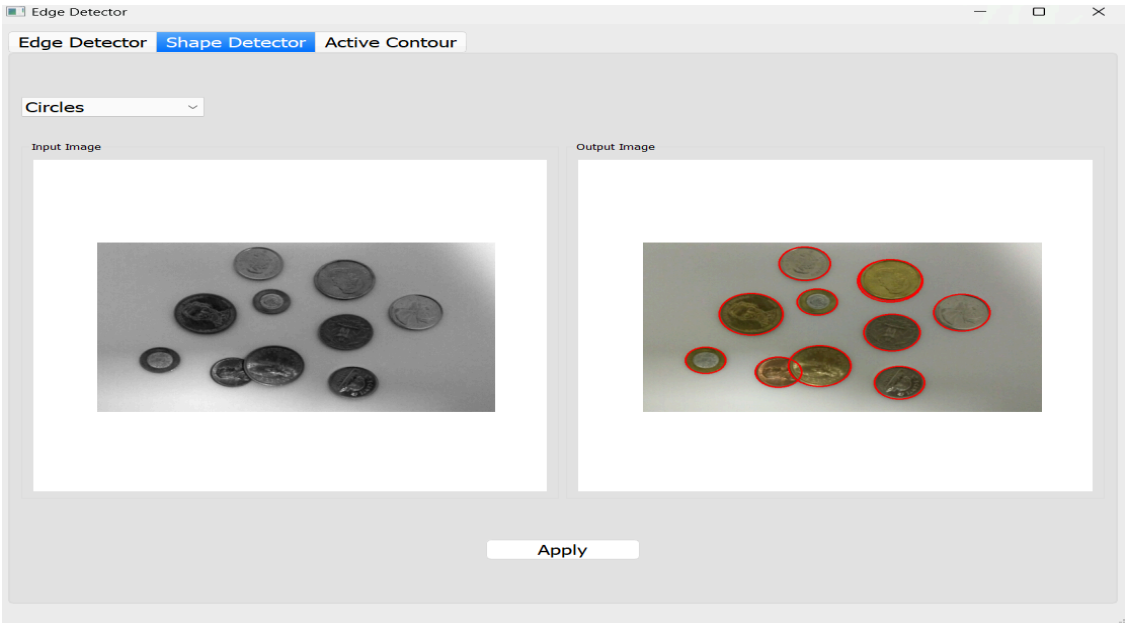
   - Returns the modified image with detected circles drawn, represented as a numpy array in RGB format.

Parameters:

- `threshold`: Threshold for circle detection, controlling the sensitivity of the algorithm.
- `region`: Region size for local maximum search, influencing the accuracy of circle detection.
- `radius`: Range of radii to search for circles, allowing customization based on the expected size of circles in the image.

Limitations:

- Performance may vary based on image quality, noise levels, and the presence of complex backgrounds.
- The accuracy of circle detection depends on the chosen parameter values, which may require tuning for optimal results.
- Detection may be affected by variations in lighting conditions or image artifacts.



Ellipse Detection using Hough Transform

Description:

This feature enables the detection of ellipses within an input image using the Hough Transform technique. It identifies elliptical shapes or patterns within images, providing valuable information for various computer vision applications.

Functionality:

1. Input Image Handling:
    - Accepts a numpy array representing the input image.
    - Converts the image to grayscale for edge detection and processing.
2. Edge Detection:
    - Applies Canny edge detection to the grayscale image to detect edges effectively.
3. Hough Transform for Ellipse Detection:
    - Defines parameters such as the minimum and maximum semi-major and semi-minor axis lengths, step sizes for axis lengths, and the number of steps for theta.
    - Initializes an accumulator to detect potential ellipse parameters such as center coordinates and axis lengths.
    - Votes for potential ellipse centers based on edge pixels and accumulates votes in the accumulator.
    - Shortlists candidate ellipses based on a specified bin threshold, representing the percentage of votes compared to the maximum possible votes.
    - Performs post-processing to remove duplicate ellipses based on a pixel threshold to ensure the uniqueness of detected ellipses.

4. Output:

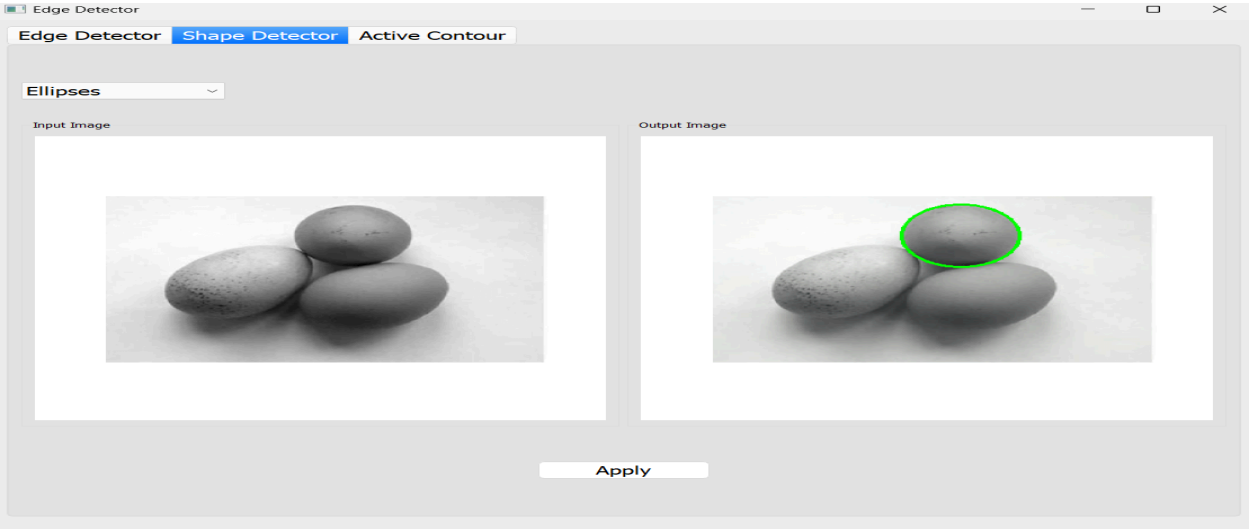
- Returns the output image with detected ellipses drawn on it, represented as a numpy array in BGR format.
- Provides a list of detected ellipses containing their center coordinates, semi-major and semi-minor axis lengths, and vote percentages.

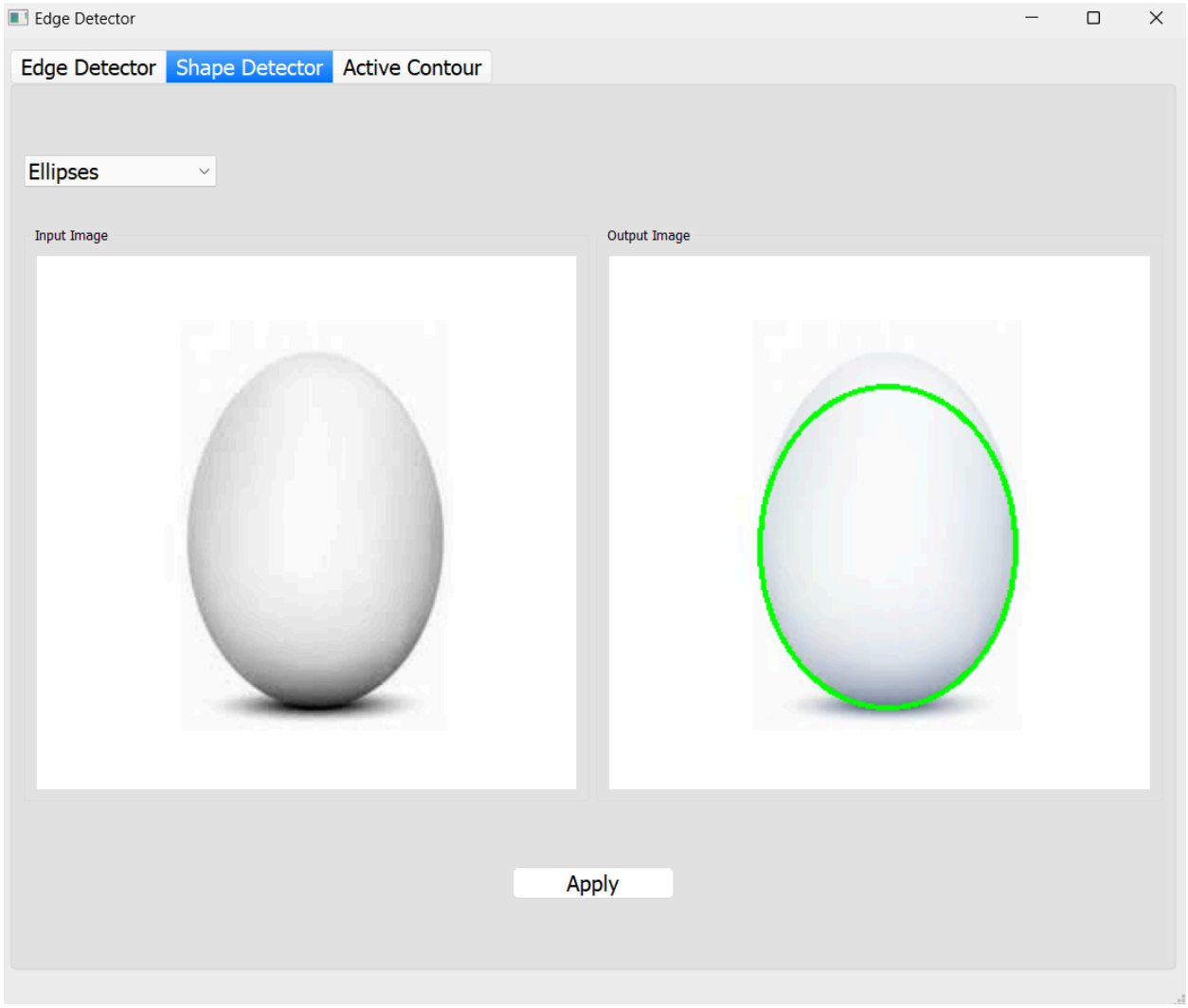
Parameters:

- a_min, a_max: Minimum and maximum semi-major axis lengths of ellipses to detect.
- b_min, b_max: Minimum and maximum semi-minor axis lengths of ellipses to detect.
- delta_a, delta_b: Step sizes for semi-major and semi-minor axis lengths, respectively.
- num_thetas: Number of steps for theta from 0 to 2π.
- bin_threshold: Thresholding value in percentage to shortlist candidate ellipses.
- min_edge_threshold, max_edge_threshold: Minimum and maximum threshold values for edge detection, controlling the sensitivity of the Canny edge detector.

Limitations:

- Performance may vary based on image quality, noise levels, and the presence of complex backgrounds or overlapping shapes.
- The accuracy of ellipse detection depends on the chosen parameter values, requiring tuning for optimal results.
- Detection may be influenced by variations in lighting conditions, image artifacts, or partial occlusions.





Team Members:

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