Faculty of Engineering

Cairo University

Computer Vision (SBE3230)

Assignment 02

“Edge and Boundary Detection”

Name	BN
Madonna Mosaad	49
Nariman Ahmed	71
Nancy Mahmoud	72
Yassien Tawfik	81

Under Supervision of :

Dr /Ahmed Badawy

Eng /Omar Hesham

Eng /Yara Wael

Table of Contents

Project Overview

This project implements the Hough Transform to detect lines, circles, and ellipses, alongside the Active Contour Model (snakes) for dynamic contour modeling, entirely through custom-written Python code without utilizing OpenCV's built-in functions. The use of manual coding techniques allows for precise adjustments and optimization of these image processing algorithms. A PyQt-based graphical user interface is integrated to provide an interactive platform for applying these methods to various image types, offering a direct means to assess their effectiveness and practicality in real-world scenarios.

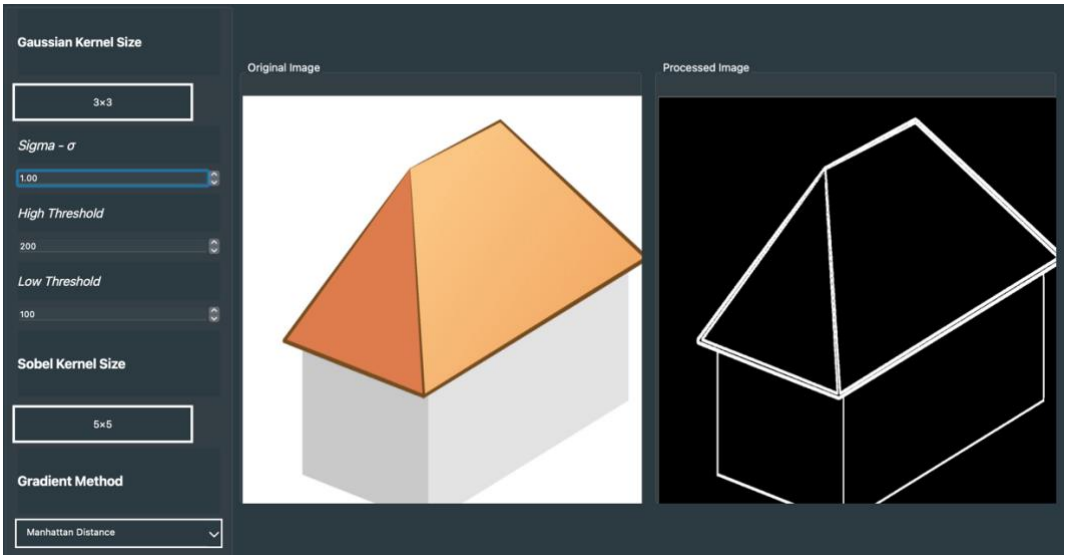
Features

Edge Detection Using Canny

The Canny Edge Detector in our project is designed to allow users full control over the edge detection process via a user-friendly graphical user interface (GUI). This interface enables users to adjust several key parameters that influence the detection results:

- **Gaussian Kernel Size**: Users can set the size of the Gaussian kernel, which is used to smooth the image prior to edge detection. This is crucial for noise reduction and detail minimization, which enhances edge prominence.
- **Sigma (σ)**: This parameter controls the standard deviation of the Gaussian filter. A higher sigma results in more pronounced smoothing, beneficial for highly noisy images.
- **High Threshold**: Defines the higher of two thresholds for detecting strong edges. It determines the required intensity gradient for a pixel to be identified as a strong edge.
- **Low Threshold**: This lower threshold aids in detecting fainter edges by defining the minimum intensity gradient necessary for a pixel to be considered part of an edge after strong edges have been detected.
- **Sobel Kernel Size**: Specifies the size of the kernel used to compute the image gradients. A larger kernel will result in more gradual gradients.
- **Gradient Method**: Users can select between 'Manhattan Distance' or 'Euclidean Distance' for calculating the intensity gradient of the image. Each method offers a different approach to how edges are detected regarding direction and sharpness.

User Interaction The GUI is equipped with an 'Apply' button, which must be clicked to update the edge detection results based on the modified settings. This feature ensures that users can experiment with different configurations and observe the impact of each parameter on the outcome without continuous reprocessing, enhancing the learning experience.



Observation

The ability to switch between Manhattan and Euclidean gradient methods provides flexibility in edge detection. The Manhattan method tends to emphasize horizontal and vertical edges, while the Euclidean approach is more sensitive to diagonal edges. Adjusting the thresholds and kernel sizes allows users to finely tune the sensitivity and specificity of the edge detection, accommodating different types of images and desired outcomes.

The manual adjustment of these parameters, combined with the 'Apply' button, offers a tailored edge detection experience and provides users with in-depth insights into the computational processes behind image processing.

Shape Detection (Lines, Circles, Ellipses)

The shape detection component of our project utilizes custom algorithms to identify lines, circles, and ellipses within images. Of the various adjustable parameters within the GUI, the single threshold setting is the most critical, as it directly influences the sensitivity and accuracy of the detection process.

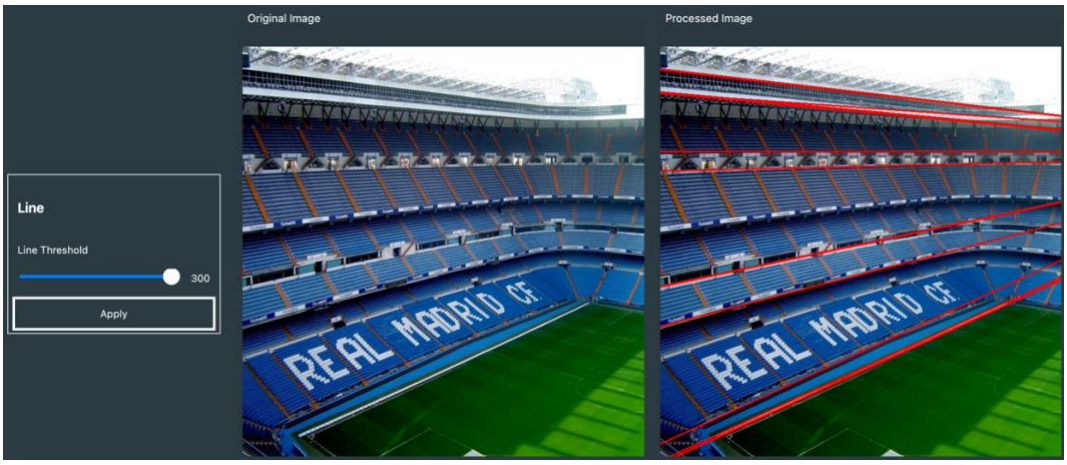
Control Parameters

- **Threshold**: This is the primary control parameter for shape detection. Adjusting this single threshold value allows users to find the optimal balance for shape recognition within their specific images. Setting the threshold appropriately is crucial for ensuring accurate detection while minimizing false positives.

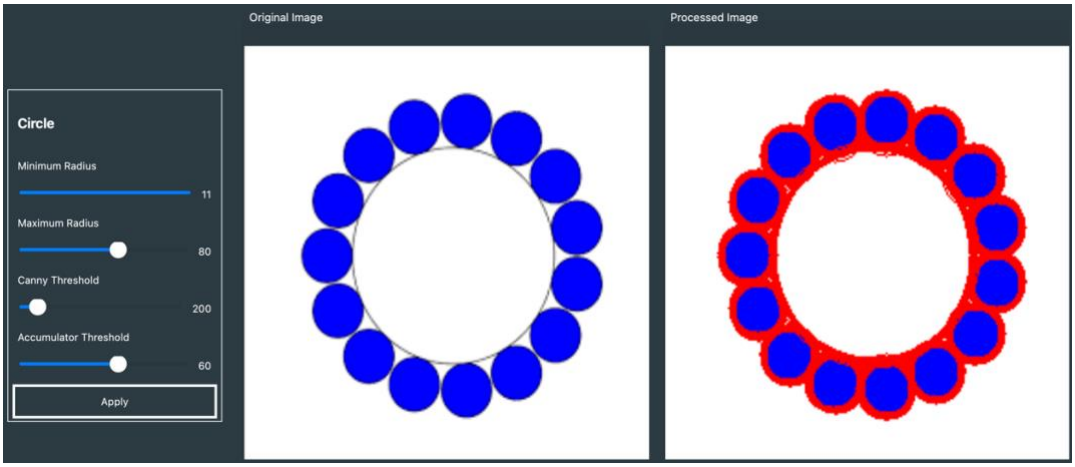
User Interaction The GUI is designed for straightforward manipulation of the threshold setting, enabling users to apply changes and immediately observe their effects on shape detection. This hands-on approach allows for iterative testing and optimization of the threshold for different types of images.

Testing the Detector To showcase the effectiveness and adaptability of the shape detection algorithms, test images are processed with varying threshold settings. Below are examples that demonstrate how the threshold impacts the detection quality for each type of shape:

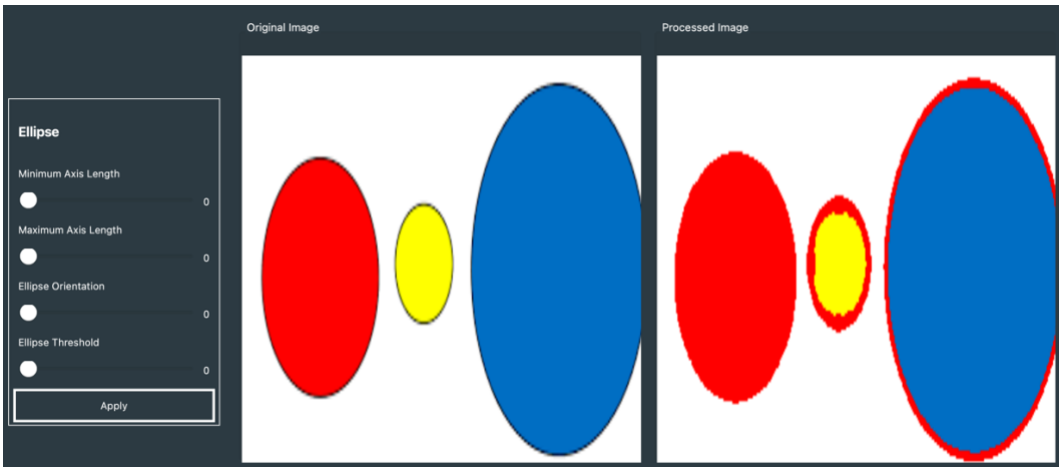
- **Line Detection**



- **Circle Detection**



- **Ellipse Detection**



Observation

The adjustment of the threshold is pivotal in determining the outcome of the detection process. Properly setting the threshold is key to balancing the detection of genuine shapes against avoiding false positives, particularly in images with complex backgrounds or overlapping shapes. This manual control over the threshold not only enhances the user's ability to fine-tune the detection parameters but also deepens the understanding of how different settings affect the algorithm's performance in real-world scenarios.

Active Contour Model (Snake)

The Active Contour Model, also known as "Snake", is a technique used in our project for contour modeling and shape approximation in images. This method adapts to the shape of an object in an image by evolving the contour with the help of internal and external forces. Among the various parameters available in the GUI, the most critical for controlling the behavior of the snake are the number of points, the weight factors, and the number of iterations, which significantly impact the accuracy and convergence of the contour.

Control Parameters

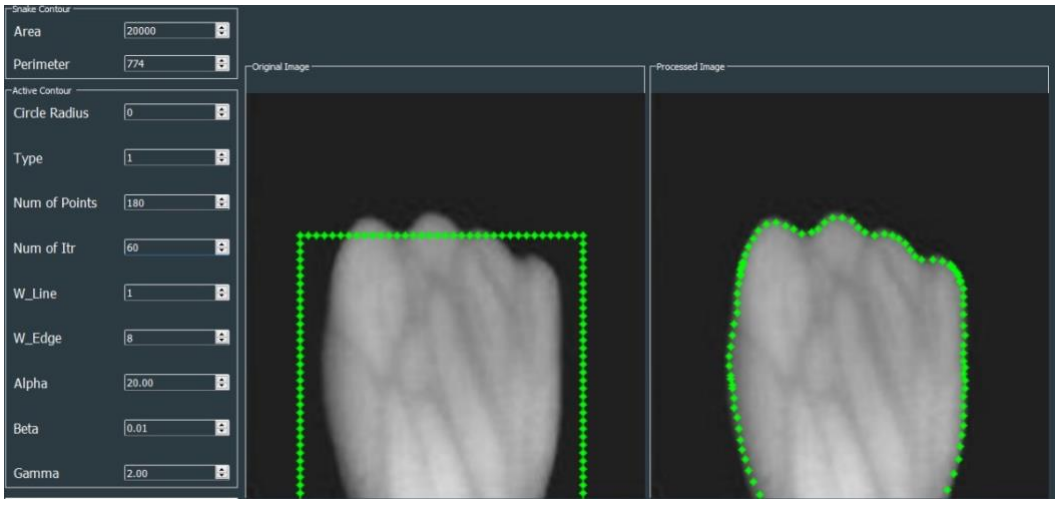
- **Number of Points**: Determines the number of control points in the initial snake, directly affecting the granularity and flexibility of the contour.
- W_{Line}, W_{Edge}, **Alpha**, **Beta**, and **Gamma**: These weights adjust the influence of internal and external forces on the snake:
    - W_{Line} controls the influence of the image's intensity on the snake.
    - W_{Edge} enhances the snake's responsiveness to edges, making it more likely to cling to sharp features.
    - **Alpha**, **Beta**, and **Gamma** manage the snake's elasticity, stiffness, and contraction, respectively, balancing the contour's smoothness against its accuracy in fitting to the object's boundaries.
- **Number of Iterations**: Sets how many times the snake algorithm will iterate, which can affect the time taken to reach a stable state and the final contour's adherence to the object edges.

User Interaction Users can modify these parameters via the GUI and apply them by pressing the 'Apply Contour' button to observe the snake's behavior as it conforms to the target shape. This interactive approach allows for real-time tuning of parameters to achieve the desired contour accuracy and dynamics.

Chain Code Representation In addition to visually displaying the evolving contour, our system generates a chain code for each contour. This chain code provides a compact numerical representation of the contour's path, which is invaluable for further analysis, such as calculating the perimeter and area or for use in machine learning models where shape features are required.

Testing the Detector To demonstrate the Active Contour Model's effectiveness, test images showing the progression of the snake from initialization to final fit are included below:

- **Tooth Contouring**:



- **Apple Contouring:**

Observation

Adjustments to the number of points and the weights (W_{Line}, W_{Edge}, Alpha, Beta, Gamma) are crucial for the contour's ability to accurately model complex shapes. These parameters must be carefully calibrated to balance between flexibility and stability of the contour, ensuring that the snake can effectively adapt to various object geometries without losing detail or deviating from true edges.

Conclusion

This project has successfully demonstrated the manual implementation of image processing techniques for edge and boundary detection, shape recognition, and contour modeling. Through the use of the Hough Transform and Active Contour Model (Snake), we have shown that precise control of parameters like thresholds and weights is crucial for accurate detection and modeling in diverse imaging scenarios. The interactive GUI provided practical insights into the algorithms' behavior, enhancing our understanding of their capabilities and limitations in real-world applications.

In summary, the project achieved its goal of deepening knowledge in computational imaging and highlighted the importance of parameter optimization in achieving high-quality image analysis outcomes.