

# **Edge and Boundary Detection**

## **Hough Transform and SNAKE**

Team 07

Faculty of Engineering - Cairo University

Dr. Ahmed Badawy

### **Collaborators**

Name	Section	B.N
Amir Hesham Khalil	1	10
Omar Mohamed Ahmed	1	46
Ahmed Emad	1	3
Hazem Zakaria	1	16
Omar Tarek	1	42

## Introduction

Edge and boundary detection algorithms play a fundamental role in various computer vision tasks, including object recognition, image segmentation, and scene understanding. They operate by identifying abrupt changes in intensity or color, which often signify the presence of object boundaries or discontinuities within an image. Accurate detection of these edges is essential for subsequent analysis and decision-making processes in vision-based systems.

The Hough Transform, introduced by Paul Hough in the 1960s, revolutionized edge detection by providing a robust method for detecting shapes, particularly lines and curves, within noisy images. Unlike traditional edge detection algorithms that rely on gradient-based approaches, the Hough Transform works by transforming the image space into parameter spaces, where geometric shapes can be represented by simple equations. By detecting intersections in this parameter space, the Hough Transform effectively identifies lines and curves, even in the presence of noise and occlusions, making it a cornerstone in computer vision applications.

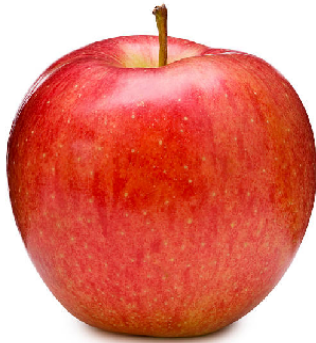
On the other hand, the Snake model, inspired by the behavior of snakes in nature, offers a flexible and adaptive approach to boundary detection. Developed by Michael Kass, Andrew Witkin, and Demetri Terzopoulos in the late 1980s, this model represents contours as a series of interconnected points or vertices, which iteratively adjust their positions to conform to the underlying image features.

In this report, we showcase our implementation and the final results for the previously mentioned processing techniques.

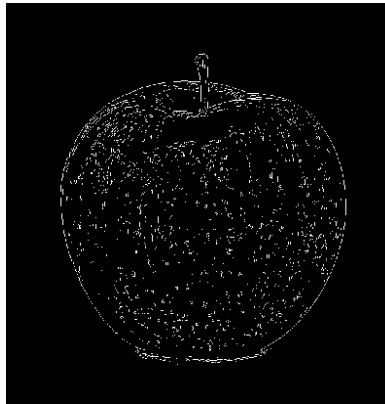
## Experiments and Results

### 1. Canny Edge Detection

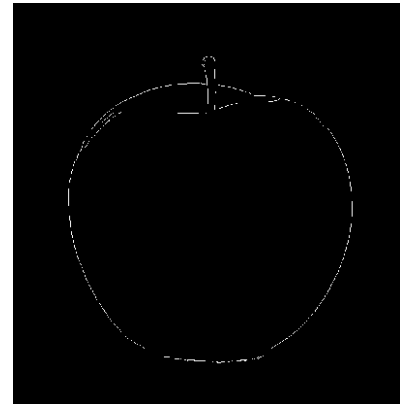
Canny Edge Detection stands as one of the most widely used and highly regarded algorithms in the realm of computer vision for detecting edges in images. Named after its creator, John F. Canny, this technique is renowned for its ability to accurately identify edges while minimizing noise and false detections. Here are some experiments results produced by our software:



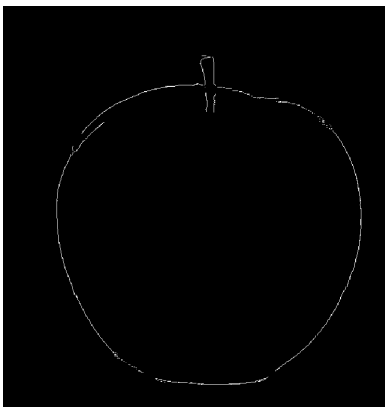
*Original Image*



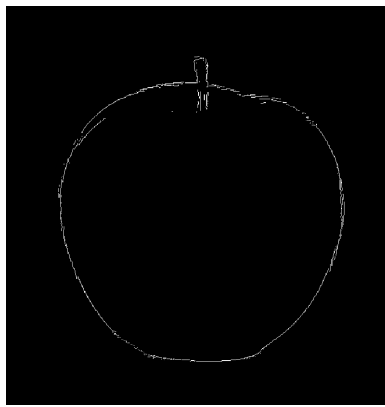
*Sigma: 1  
Low Threshold: 0.1  
High Threshold: 0.1*



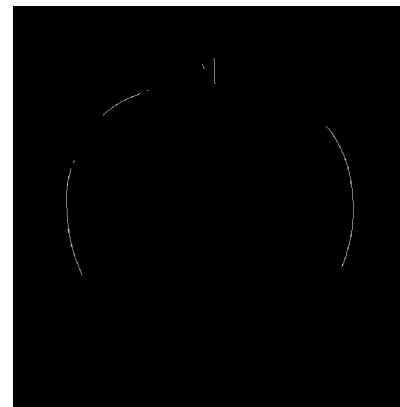
*Sigma: 2  
Low Threshold: 0.5  
High Threshold: 0.3*



*Sigma: 4  
Low Threshold: 0.3  
High Threshold: 0.4*



*Sigma: 9  
Low Threshold: 0.3  
High Threshold: 0.4*



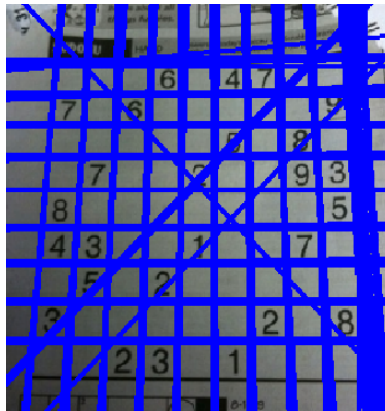
*Sigma: 3  
Low Threshold: 0.5  
High Threshold: 0.8*

## 2. Hough Detection

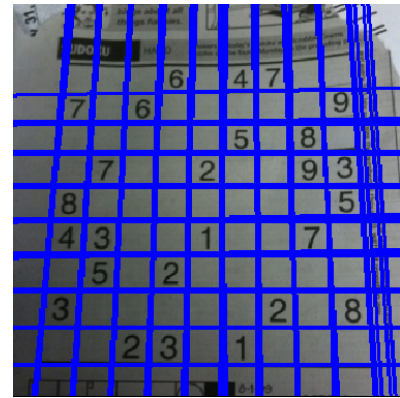
Hough Detection, also known as the Hough Transform, is a powerful technique in computer vision used primarily for detecting geometric shapes within images, notably lines and curves.



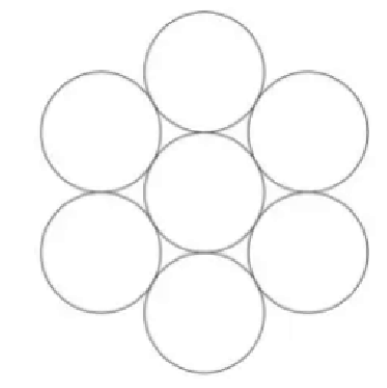
*Sudoku Photo*



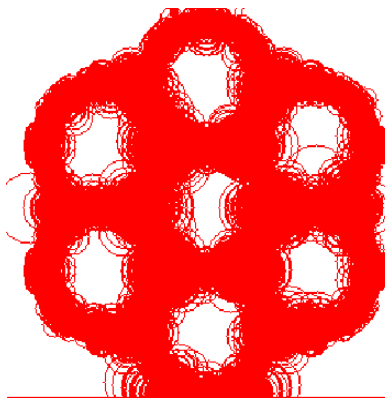
*Line Hough Transform  
Threshold: 108*



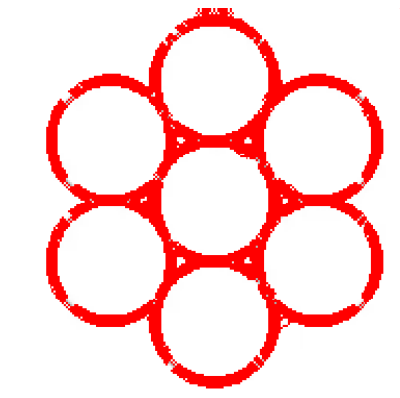
*Line Hough Transform  
Threshold: 130*



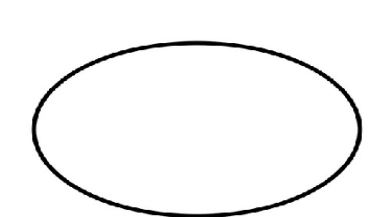
*Tangent Circles*



*Circle Hough Transform  
Threshold: 70*



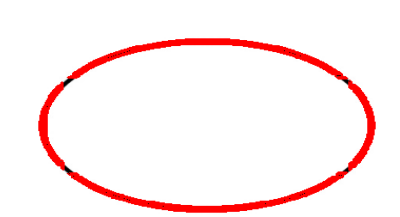
*Circle Hough Transform  
Threshold: 165*



*Simple Ellipse*



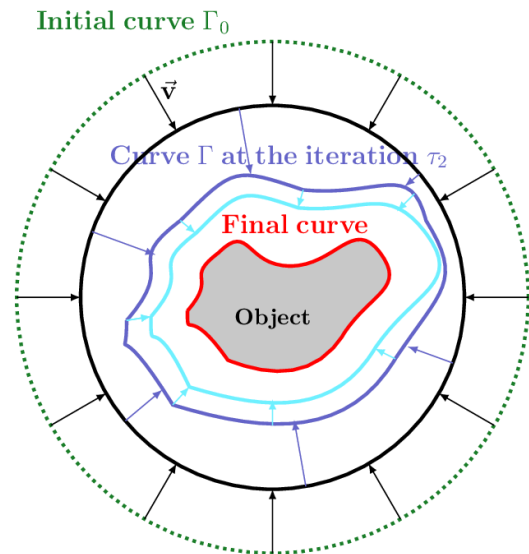
*Ellipse Hough Transform  
Threshold: 170*



*Ellipse Hough Transform  
Threshold: 170*

### 3. Active Contour

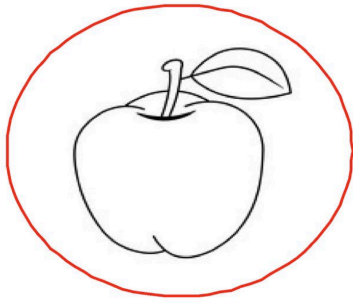
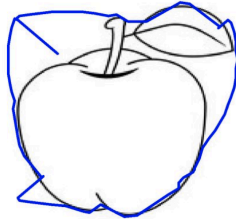
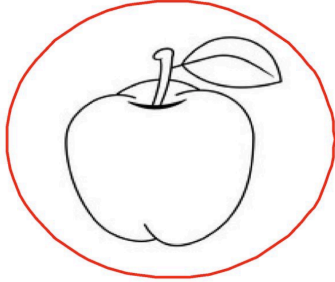
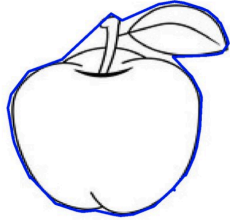
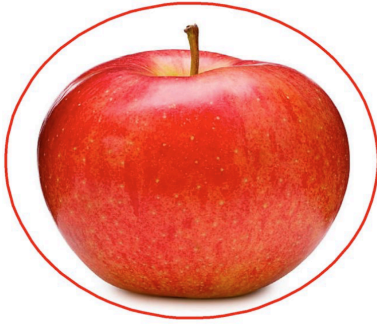
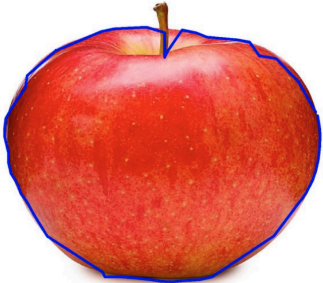
Active contour technique, or snakes, is like a digital detective for images. It does segmentation by finding object edges by adjusting a contour to hug them snugly. Forces pull the contour toward edges while keeping it smooth. We can give hints to guide the contour. It's like solving a puzzle of object edges, making images easier for computers to understand.



Before showing the results, it might be useful if we discussed the controlled parameters used in the active contour technique:

1. **Alpha:** Weight of the internal energy term related to the contour smoothness or curvature.
2. **Beta:** Weight of the internal energy term related to the contour's elasticity or flexibility.
3. **max\_num\_iter:** Maximum number of iterations allowed for the contour optimization process. It determines when the optimization process should stop if convergence is not reached earlier.

The `max_num_iter` parameter, governing the convergence behavior of the algorithm, ensures that the contour refinement process terminates within a reasonable computational time. This parameter strikes a balance between computational efficiency and convergence accuracy, enabling the method to effectively capture object boundaries without excessive computational overhead.

Before AC	After AC	Parameters/Results
		<b>Params:</b> <ul style="list-style-type: none"> <li>- Alpha = 1.6</li> <li>- Beta = 0.01</li> <li>- max_num_iter = 36</li> </ul> <b>Resulted Contour:</b> <ul style="list-style-type: none"> <li>- Area = 179.5</li> <li>- Perimeter = 93</li> </ul>
		<b>Params:</b> <ul style="list-style-type: none"> <li>- Alpha = 0.1</li> <li>- Beta = 0.02</li> <li>- max_num_iter = 150</li> </ul> <b>Resulted Contour:</b> <ul style="list-style-type: none"> <li>- Area = 121.5</li> <li>- Perimeter = 64</li> </ul>
		<b>Params:</b> <ul style="list-style-type: none"> <li>- Alpha = 1.3</li> <li>- Beta = 0.02</li> <li>- max_num_iter = 150</li> </ul> <b>Resulted Contour:</b> <ul style="list-style-type: none"> <li>- Area = 138.5</li> <li>- Perimeter = 74</li> </ul>

The active contour technique, utilizing parameters alpha, beta, and max\_num\_iter, has demonstrated commendable performance when applied to the provided images. The selected parameter values have effectively balanced the contour's responsiveness to object boundaries (alpha), its elasticity (beta), and the maximum number of iterations allowed for convergence (max\_num\_iter). A constant encouraging factor of value = 0.1 was utilized to ensure contour points stay away from the edges of the overall image, thus converging precisely on the object.

## Software Design and Architecture

The application is built upon a robust software design architecture that leverages the Python programming language, PyQt framework, Qt Designer for UI design, and follows the Model-View-Controller (MVC) pattern for efficient organization and separation of concerns.

### 1. Model-View-Controller (MVC) Pattern

- **Model:** The Model component represents the data and business logic of the application. In our context, it encapsulates image processing algorithms, data structures for image representation, and manipulation functions.

The model in our case is represented in Image processing algorithms, data structures, and functions reside within the Model component, ensuring encapsulation and modularity.

- **View:** The View component is responsible for presenting the user interface to the user. Qt Designer is utilized to design the graphical user interface (GUI) elements, including buttons, sliders, image displays, and histograms.

The view in our case encompasses the graphical user interface designed using Qt Designer. It includes various widgets for user interaction and image display.

- **Controller:** The Controller acts as an intermediary between the Model and the View, handling user input and triggering appropriate actions. It binds GUI events to corresponding image processing functions, ensuring seamless interaction between the user and the application.

The controller in our case bridges the gap between the Model and the View, handling user events and orchestrating image processing tasks based on user input.

## 2. Used Technologies and Tools

- **Python:** Python serves as the primary programming language for our application, offering simplicity, readability, and extensive libraries for image processing tasks.
- **PyQt:** PyQt is a set of Python bindings for the Qt application framework. It enables the integration of Qt functionalities within Python applications, providing access to a vast array of GUI components and features for building cross-platform desktop applications.
- **Qt Designer:** Qt Designer is utilized for designing the graphical user interface of our application. It offers a drag-and-drop interface for creating and arranging UI elements, streamlining the GUI design process. Designed UI files (.ui) created in Qt Designer are seamlessly integrated into the Python codebase using PyQt, allowing for rapid development and modification of the user interface.

### A. Goals

- **Modularity:** The MVC pattern facilitates modular design, enabling independent development and testing of Model, View, and Controller components.
- **Separation of Concerns:** MVC separates the presentation logic (View) from the application logic (Model), enhancing maintainability and scalability.
- **Ease of Development:** Python, PyQt, and Qt Designer collectively provide a conducive environment for rapid application development, allowing us, as developers, to focus on functionality rather than low-level details of GUI implementation.

