Convex Hull and Line Intersection Visualization

21K-3416 Arif **21k-3187 Maaz** **21k-3371 Fardeen**

November, 2023

Abstract

This project aims to develop a visualization tool for convex hull algorithms and line intersection. The implemented system utilizes Python and the Tkinter library to create an interactive graphical user interface. Convex hull algorithms, including QuickHull, GrahamScan, JarvisMarch, BruteForce, and monotone, are incorporated, allowing users to visualize and compare their outcomes. Additionally, the system includes functionality for visualizing the intersection of two lines.

1 Introduction

Convex hull algorithms play a crucial role in computational geometry, with applications ranging from computer graphics to geographic information systems. Moreover, line intersection is a fundamental geometric operation with applications in areas such as computer-aided design and robotics. This project introduces a visual exploration of convex hull algorithms and line intersection to enhance understanding and facilitate comparisons.

Programming Design

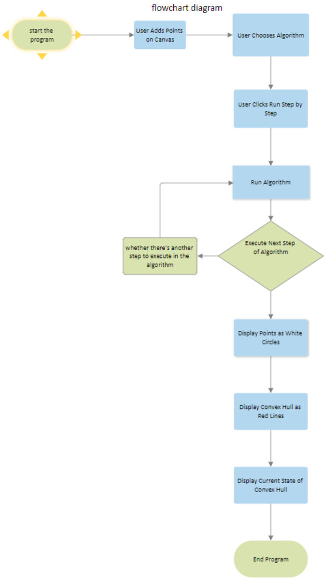

flowchart diagram
start the program
User Adds Points on Canvas
User Chooses Algorithm
User Clicks Run Step by Step
Run Algorithm
Execute Next Step of Algorithm
whether there's another step to execute in the algorithm
Display Points as White Circles
Display Convex Hull as Red Lines
Display Current State of Convex Hull
End Program

Figure 1: Program Flowchart

In Figure 1, Start Program: This is the starting point of the program.User AddsPointsonCanvas: Represents the user action of adding points on the canvas. User Chooses Algorithm: Represents the user's action of choosing a specific algorithm.User Clicks Run Step by Step: Represents the user action of running the algorithm step by step.Run Algorithm:

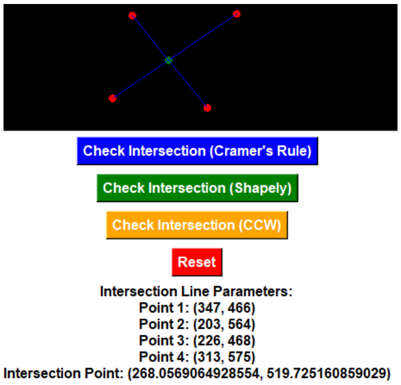
Represents the process of executing the chosen algorithm. Execute the Next Step of the Algorithm:Represents a decision point, indicating whether there's another step to execute in the algorithm. Display Points as White Circles:Represents the process of displaying points on the canvas as white circles.Display Convex Hull as Red Lines: Represents the process of displaying the convex hull as red lines on the canvas. Display Current State of Convex: Represents the process of displaying the current state of the convex hull during step-by-step execution.End Program: This is the endpoint of the program. Language: Python

Experimental Setup

The experimental setup involves interactively adding points on the canvas. Users can choose convex hull algorithms or line intersections, run them, and observe the results.

Results and Discussion

1.1 Line Intersection Visualization


Check Intersection (Cramer's Rule)
Check Intersection (Shapely)
Check Intersection (CCW)
Reset

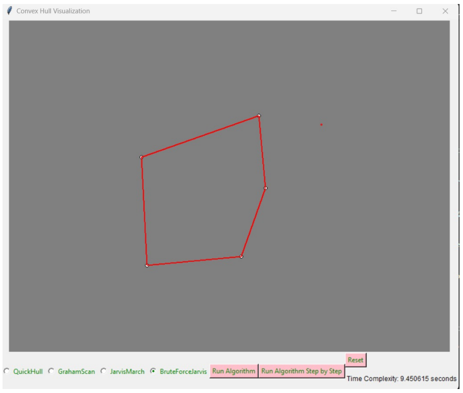
Intersection Line Parameters:
Point 1: (347, 466)
Point 2: (203, 564)
Point 3: (226, 468)
Point 4: (313, 575)
Intersection Point: (268.0569064928554, 519.725160859029)

Figure 2: Illustration of Line Intersection: Two lines intersect at a point.

In Figure 2, you can see an illustration of a line intersection where two lines intersect at a specific point. This point of intersection is calculated by our program.

1.2 Convex Hull Visualization

In Figure 3, you can see an illustration of a convex hull that calculates the time and creates a hull while comparing other points. This is calculated by our program.



○ QuickHull ○ GrahamScan ○ JarvisMarch ● BruteForceJarvis

Time Complexity: 9.450615 seconds

Figure 3: Convex Hull Visualization - Screenshot

Conclusion

This project successfully implements an interactive visualization tool for convex hull algorithms and line intersection. The comparative analysis of algorithms provides valuable insights into their performance. The addition of line intersections enhances the versatility of the system.

References

- Shapely Documentation
- Tkinter Documentation
- Monotone Documentation