CONVEX HULL AND LINE SEGMENT ALGORITHMS IMPLEMENTATION AND ANALYSIS:

Project Team

Arooba Minhas	21K-3094
Bushra Khan	21K-3081
Alaina Usmani	21K-3155

Session 2023-2024

Supervised by

Mr Faisal Ali

Department of Computer Science

National University of Computer and Emerging Sciences
Karachi, Pakistan

November, 2023

Certificate of Approval

The Department of Computer Science, National University of Computer and Emerging Sciences, accepts this thesis titled *CONVEX HULL AND LINE SEGMENT ALGORITHMS IMPLEMENTATION AND ANALYSIS:*, submitted by Arooba Minhas (21K-3094), Bushra Khan (21K-3081), and Alaina Usmani (21K-3155), in its current form, and it is satisfying the dissertation requirements for the project of Design and Analysis of Algorithm course in Artificial Intelligence.

Supervisor

Mr Faisal Ali   Signature: ________________

Abstract

This report presents the implementation and analysis of five convex hull algorithms—Brute Force, Jarvis March, Gramham Scan, Quick Elimination, and Andrew's Algorithm, as well as a Line Segment Intersection algorithm using three different methods—Algebraic, CCW Computation, and Vector Cross Product. The implementation is carried out in Python, featuring a graphical user interface (GUI) for visualizing the results of both sets of algorithms. These algorithms are applied to respective sets of points and line segments, and their time and space complexities are thoroughly analyzed. The report aims to provide a comprehensive understanding of the performance characteristics and suitability of each algorithm in their respective problem domains.

0.1 Introduction

In the realm of computational geometry, convex hull algorithms stand as pivotal solutions to the challenge of identifying the convex hull within a given set of points. This report delves into the exploration of five distinct algorithms, each bringing its own unique approach to resolving the convex hull problem. Concurrently, the report also delves into the intricacies of Line Segment Intersection—a fundamental problem in computational geometry. This particular challenge involves determining the intersection of two provided line segments. Within this context, the report scrutinizes three different algorithms, each offering its distinct methodology for tackling the Line Segment Intersection problem.

0.2 Programming Design

0.2.1

Convex Hull Algorithms:

0.2.1 Brute Force Convex Hull Algorithm:

The Brute Force algorithm iteratively checks every triplet of points for a valid convex hull. This algorithm has a time complexity of O(n^4) and a space complexity of O(n).
0.2.2 Jarvis March Convex Hull Algorithm:

Jarvis March, also known as the Gift Wrapping algorithm, selects the point with the lowest y-coordinate as the starting point and iteratively selects the next point with the smallest polar angle. The time complexity of Jarvis March is O(n^2), with a space complexity of O(n). **0.2.3 Graham Scan Convex Hull Algorithm:**

The Graham Scan algorithm involves sorting points based on their polar angles and constructing the convex hull in a counterclockwise manner. The time complexity of Graham Scan is O (n log n), and its space complexity is O(n).

0.2.4 Quick Elimination Convex Hull Algorithm:

Quick Elimination is an optimization of Graham Scan, where the point set is divided into smaller subsets, and Graham Scan is applied to each subset. The time complexity is O (n log h), and the space complexity is O(n).

0.2.5 Andrew's Monotone Chain Convex Hull Algorithm:

Andrew's Algorithm sorts the points and computes the convex hull in two separate chains—lower and upper. The time complexity is O (n log n), and the space complexity is O(n).

0.3 Line Segment Algorithms:

0.3.1

0.2.6 Algebraic Line Segment Intersection Algorithm:

The Algebraic method calculates the slopes of the line segments and checks for intersection using algebraic equations. The time complexity is dependent on the solution of a system of equations and is analyzed further in the results section.

0.2.7 CCW Computation Line Segment Intersection Algorithm:

The CCW (Counter Clockwise) computation method involves determining the orientation of three points. The algorithm checks the orientations of relevant point triplets to identify intersections efficiently.

0.2.8 Vector Cross Product Line Segment Intersection Algorithm:

The Vector Cross Product method utilizes vector operations to determine whether two line segments intersect. This approach offers an alternative computational perspective.

0.4 Experimental Setup

Implemented in Python, the algorithms feature a graphical user interface (GUI) tailored for interactive visualization. This GUI facilitates user engagement by enabling the input of points or line segments, the selection of a specific algorithm, and the observation of the corresponding convex hull construction or intersection detection, depending on the context.

Experimental Setup Flow:

0.4.1

Initialization:

Set up the Python environment. Import necessary libraries: tkinter for GUI, matplotlib for plotting, compare to key for sorting, time for measuring execution time, sys for memory usage, and sympy for algebraic computations. Graphical User Interface (GUI) Initialization:

For Convex Hull Algorithms:

Create a GUI for visualizing convex hull construction. Define a canvas for plotting points and convex hulls. Add buttons for each convex hull algorithm (Graham Scan, Jarvis March, Quick Elimination, Andrew's Algorithm, Brute Force). Include buttons for clearing points and displaying time and space complexities. Design labels for time and space complexities.

For Line Segment Intersection Algorithm:

Create a GUI for visualizing line segment intersections. Define a canvas for plotting line segments and intersection points. Add buttons for different line segment intersection methods(Algebraic, CCW Computation, Vector Cross Product).Include buttons for adding points, clearing canvas, and displaying intersection results. Design labels for displaying intersection results.

0.4.2

Algorithm Implementation:

Implement Convex Hull Algorithms in separate functions/classes. Graham Scan, Jarvis March, Quick Elimination, Andrew's Algorithm, Brute Force. Implement Line Segment Intersection Algorithm in a separate class. Methods for Algebraic, CCW Computation, Vector Cross Product.

0.4.3

Experimental Execution:

For Convex Hull Algorithms:

Measure the execution time using the time library. Measure space complexity using sys.getsizeof(). Update the GUI labels with time and space complexity results.

For Line Segment Intersection Algorithm:

Implement methods to check intersection using different techniques. Update the GUI labels with intersection results.

0.5 Results and Discussions:

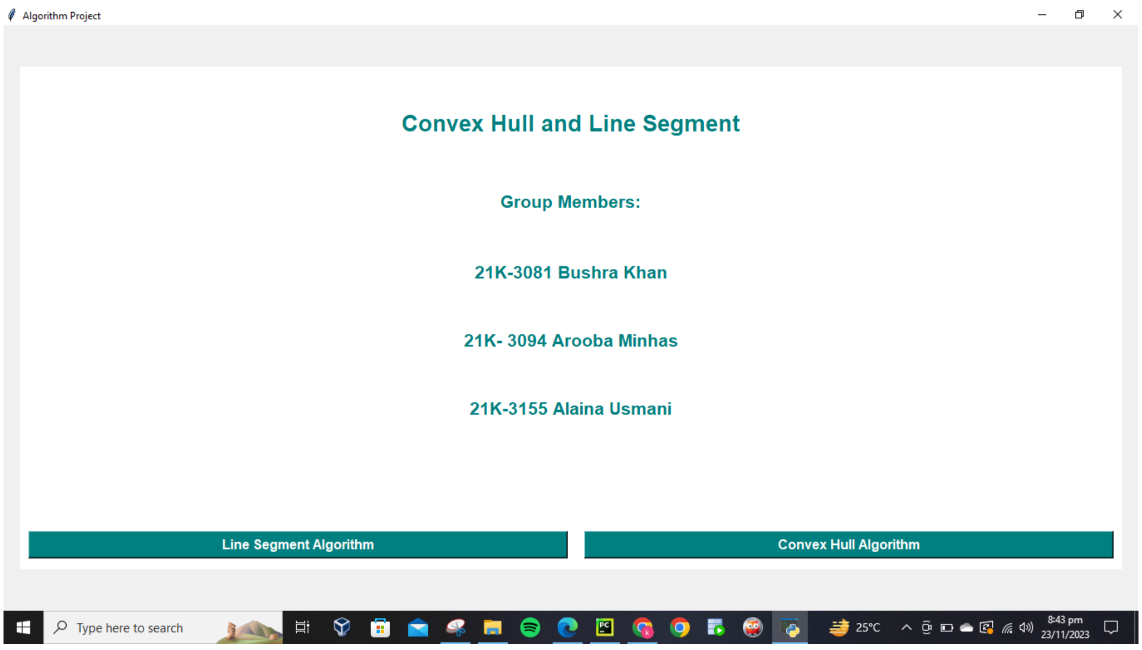


Figure 1: Title Page

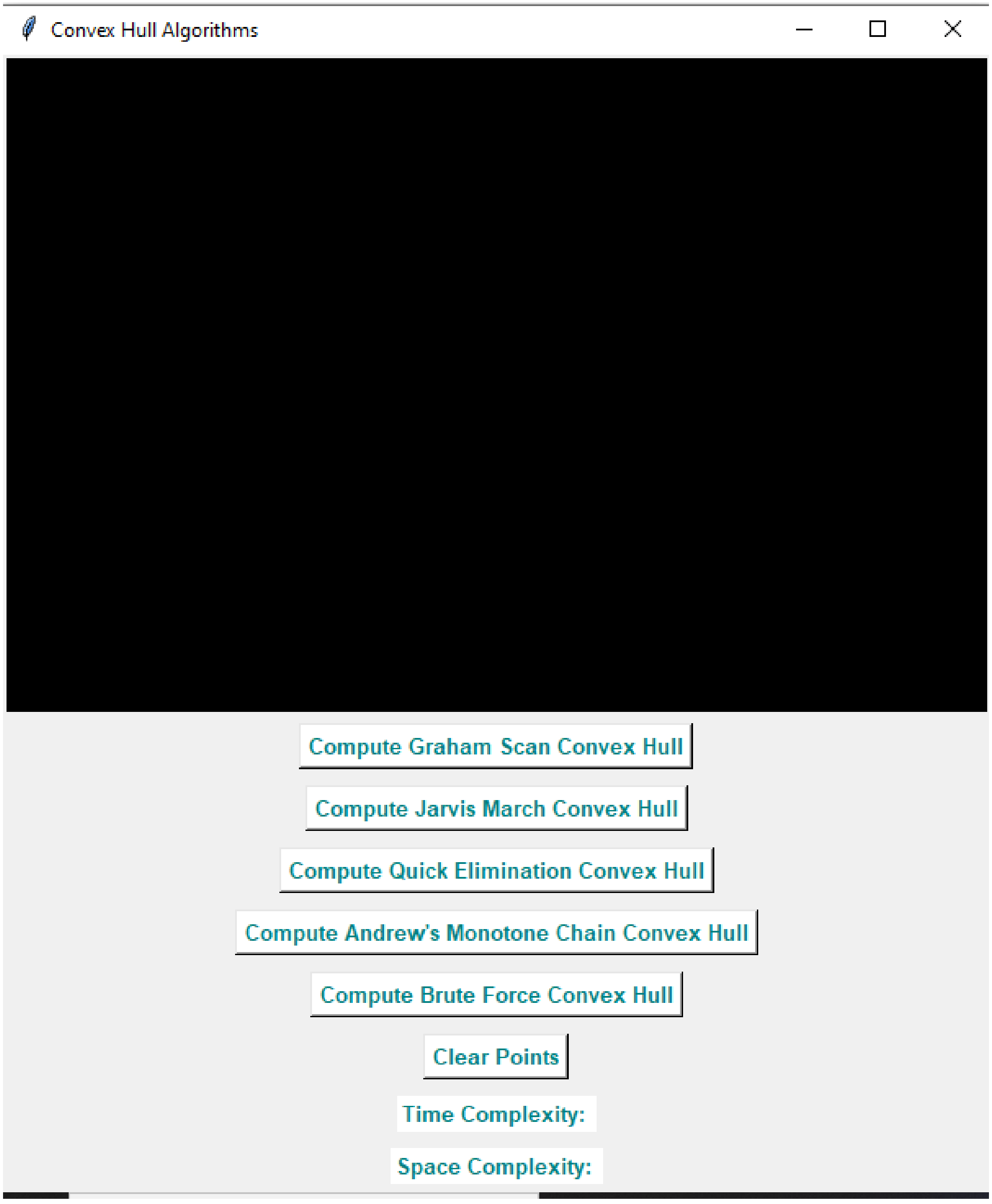

Convex Hull Algorithms
Compute Graham Scan Convex Hull
Compute Jarvis March Convex Hull
Compute Quick Elimination Convex Hull
Compute Andrew's Monotone Chain Convex Hull
Compute Brute Force Convex Hull
Clear Points
Time Complexity:
Space Complexity:

Figure 2: Convex Hull Algorithm Page

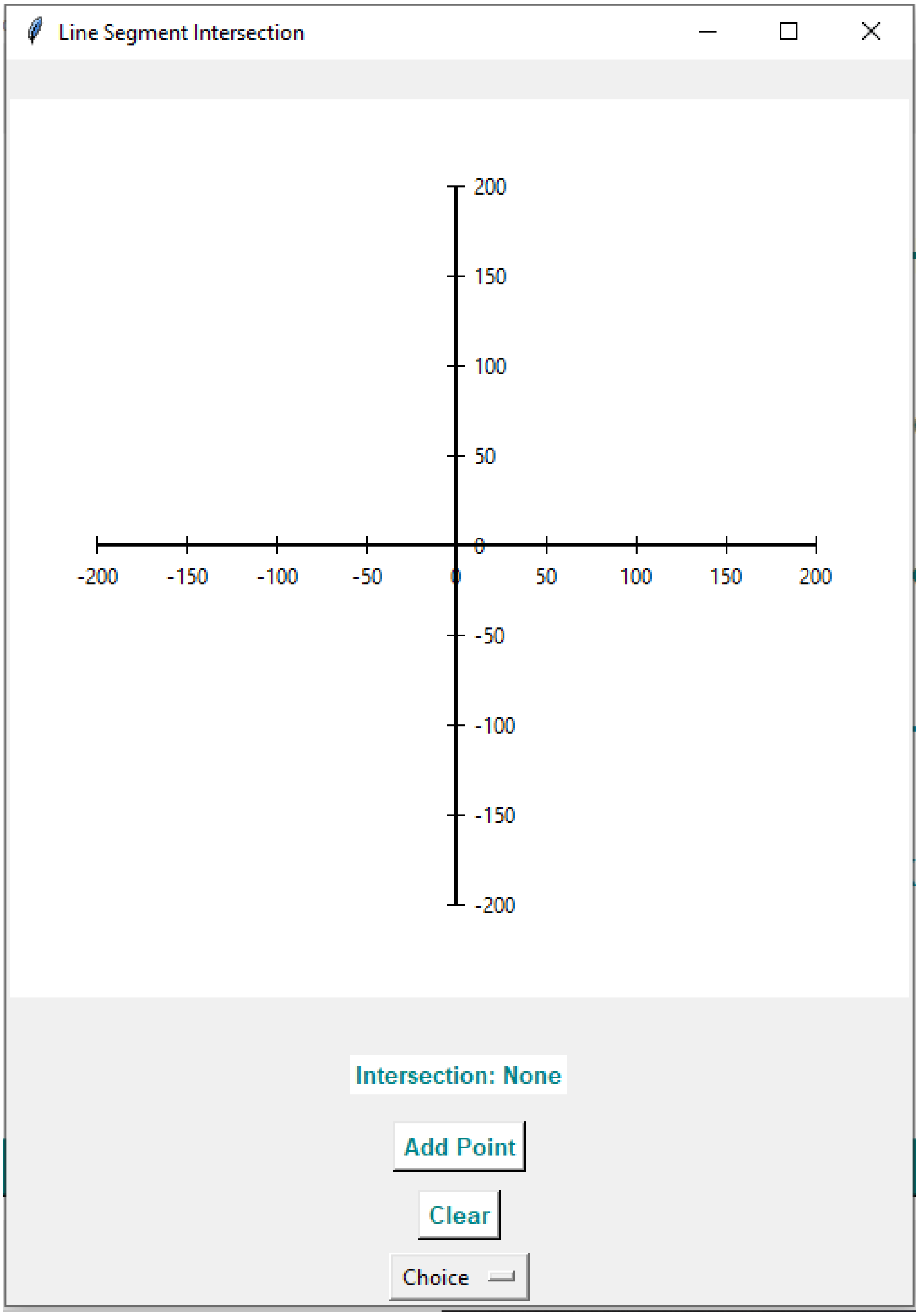
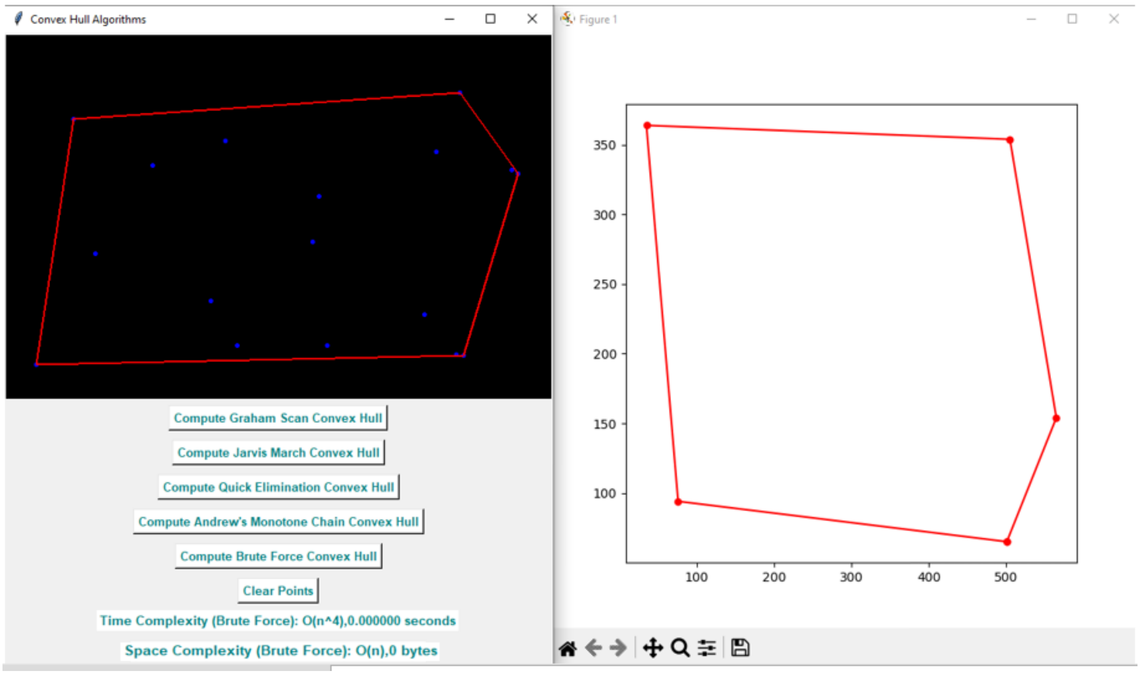


Figure 3: Line Segment Algorithm Page

0.6 Convex Hull Algorithms

0.6.1 Brute Force Convex Hull:

The Brute Force algorithm exhibits high time complexity, making it impractical for large datasets. The resulting convex hull may be accurate, but the computational cost is significant.

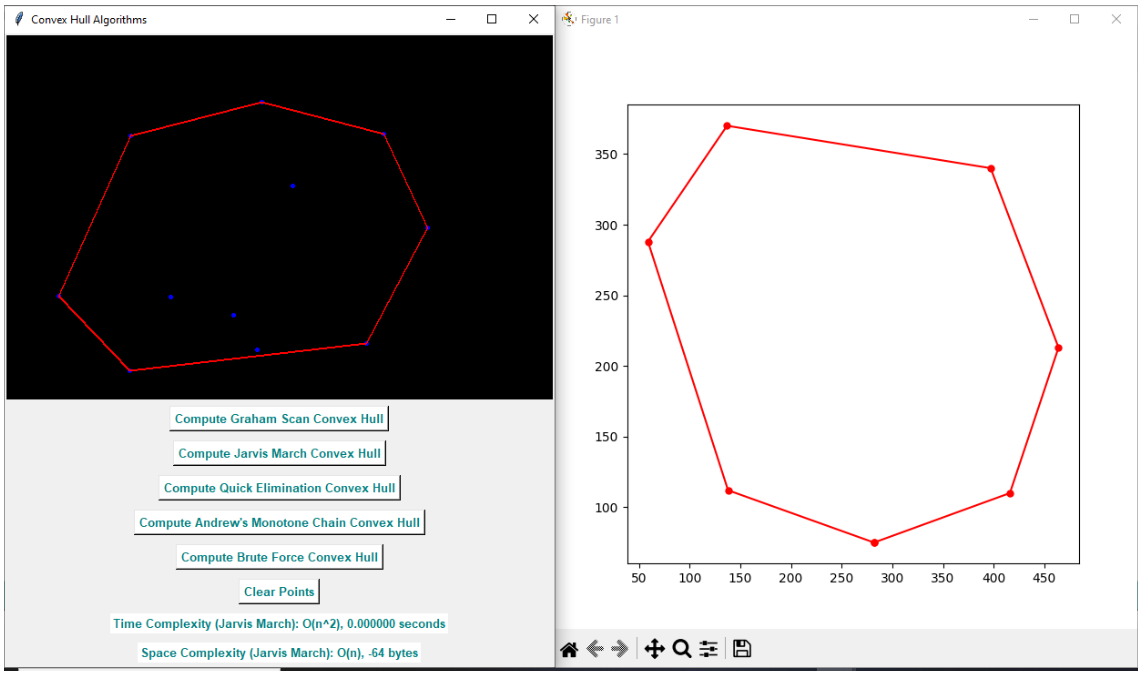

Convex Hull Algorithms
Figure 1
Compute Graham Scan Convex Hull
Compute Jarvis March Convex Hull
Compute Quick Elimination Convex Hull
Compute Andrew's Monotone Chain Convex Hull
Compute Brute Force Convex Hull
Clear Points
350
300
250
200
150
100
100
200
300
400
500

Time Complexity (Brute Force): O(n^4),0.000000 seconds
Space Complexity (Brute Force): O(n),0 bytes

Figure 4: Brute Force

0.6.2 Jarvis March Convex Hull:

Jarvis March provides a simple and intuitive solution with moderate time complexity. However, it may struggle with degenerate cases.


Convex Hull Algorithms
Figure 1
Compute Graham Scan Convex Hull
Compute Jarvis March Convex Hull
Compute Quick Elimination Convex Hull
Compute Andrew's Monotone Chain Convex Hull
Compute Brute Force Convex Hull
Clear Points
350
300
250
200
150
100
50
100
150
200
250
300
350
400
450

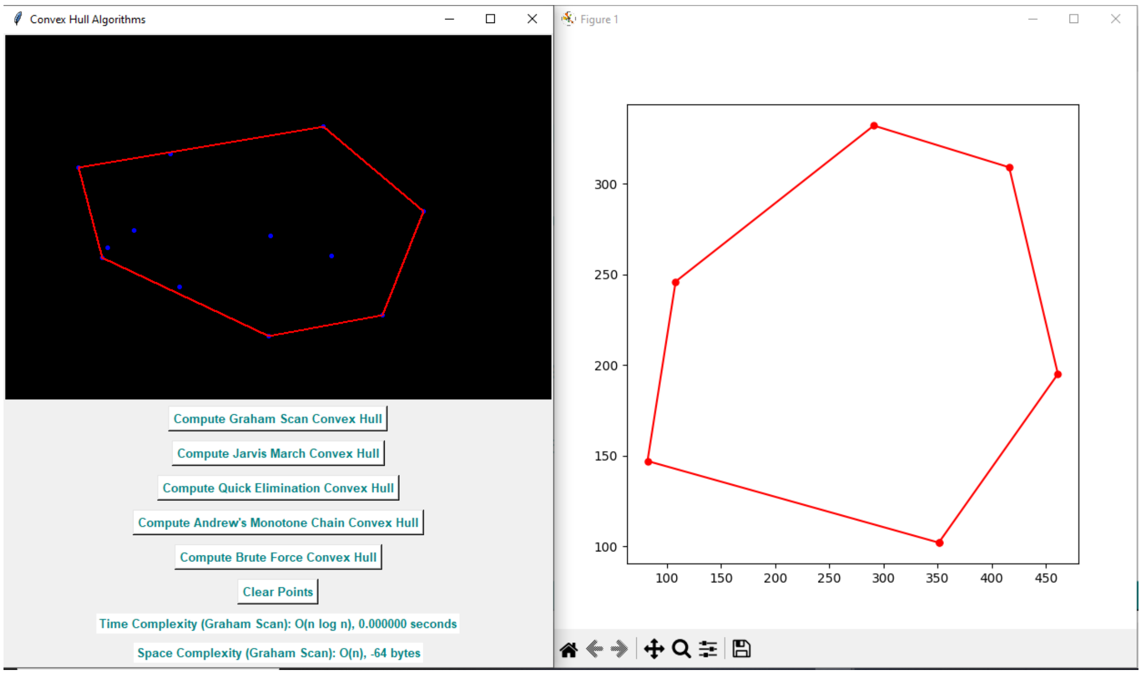
Time Complexity (Jarvis March): O(n^2), 0.000000 seconds

Space Complexity (Jarvis March): O(n), -64 bytes

Figure 5: Jarvis March

0.6.3 Graham Scan Convex Hull:

Graham Scan demonstrates improved efficiency, especially for large datasets. Its sorting step contributes to better overall performance.


Convex Hull Algorithms
Compute Graham Scan Convex Hull
Compute Jarvis March Convex Hull
Compute Quick Elimination Convex Hull
Compute Andrew's Monotone Chain Convex Hull
Compute Brute Force Convex Hull
Clear Points
Figure 1
300
250
200
150
100
100
150
200
250
300
350
400
450

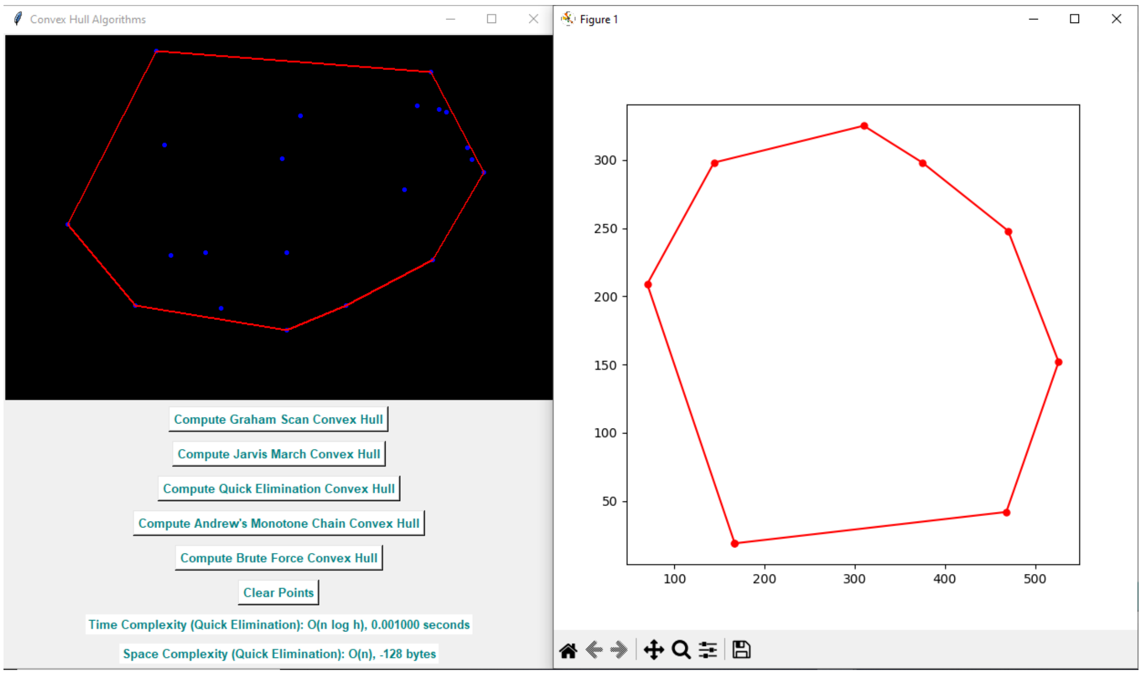
Time Complexity (Graham Scan): O(n log n), 0.000000 seconds

Space Complexity (Graham Scan): O(n), -64 bytes

Figure 6: Graham Scan

0.6.4 Quick Elimination Convex Hull:

Quick Elimination optimizes Graham Scan by dividing the point set into smaller subsets. This approach reduces the time complexity for specific cases.


Convex Hull Algorithms
Figure 1
Compute Graham Scan Convex Hull
Compute Jarvis March Convex Hull
Compute Quick Elimination Convex Hull
Compute Andrew's Monotone Chain Convex Hull
Compute Brute Force Convex Hull
Clear Points
300
250
200
150
100
50
100
200
300
400
500

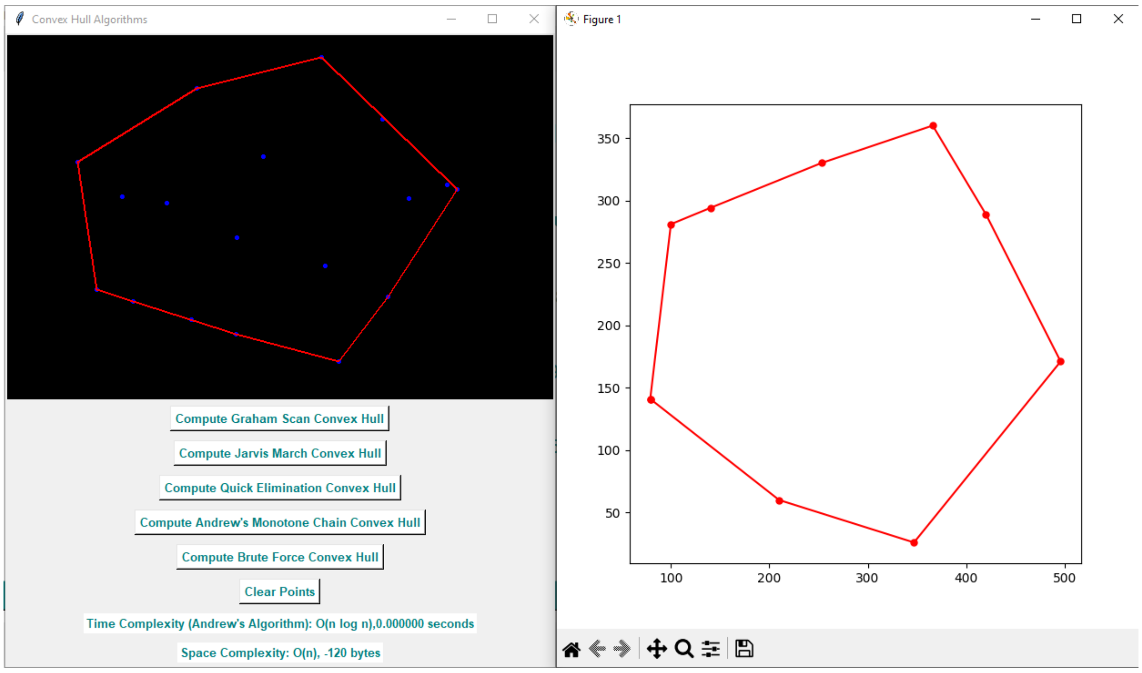
Time Complexity (Quick Elimination): O(n log h), 0.001000 seconds

Space Complexity (Quick Elimination): O(n), -128 bytes

Figure 7: Quick Elimination

0.6.5 Andrew's Monotone Chain Convex Hull:

Andrew's Algorithm offers a balance between efficiency and simplicity. It performs well for moderate-sized datasets.


Convex Hull Algorithms
Figure 1
Compute Graham Scan Convex Hull
Compute Jarvis March Convex Hull
Compute Quick Elimination Convex Hull
Compute Andrew's Monotone Chain Convex Hull
Compute Brute Force Convex Hull
Clear Points
350
300
250
200
150
100
50
100
200
300
400
500

Time Complexity (Andrew's Algorithm): O(n log n),0.000000 seconds

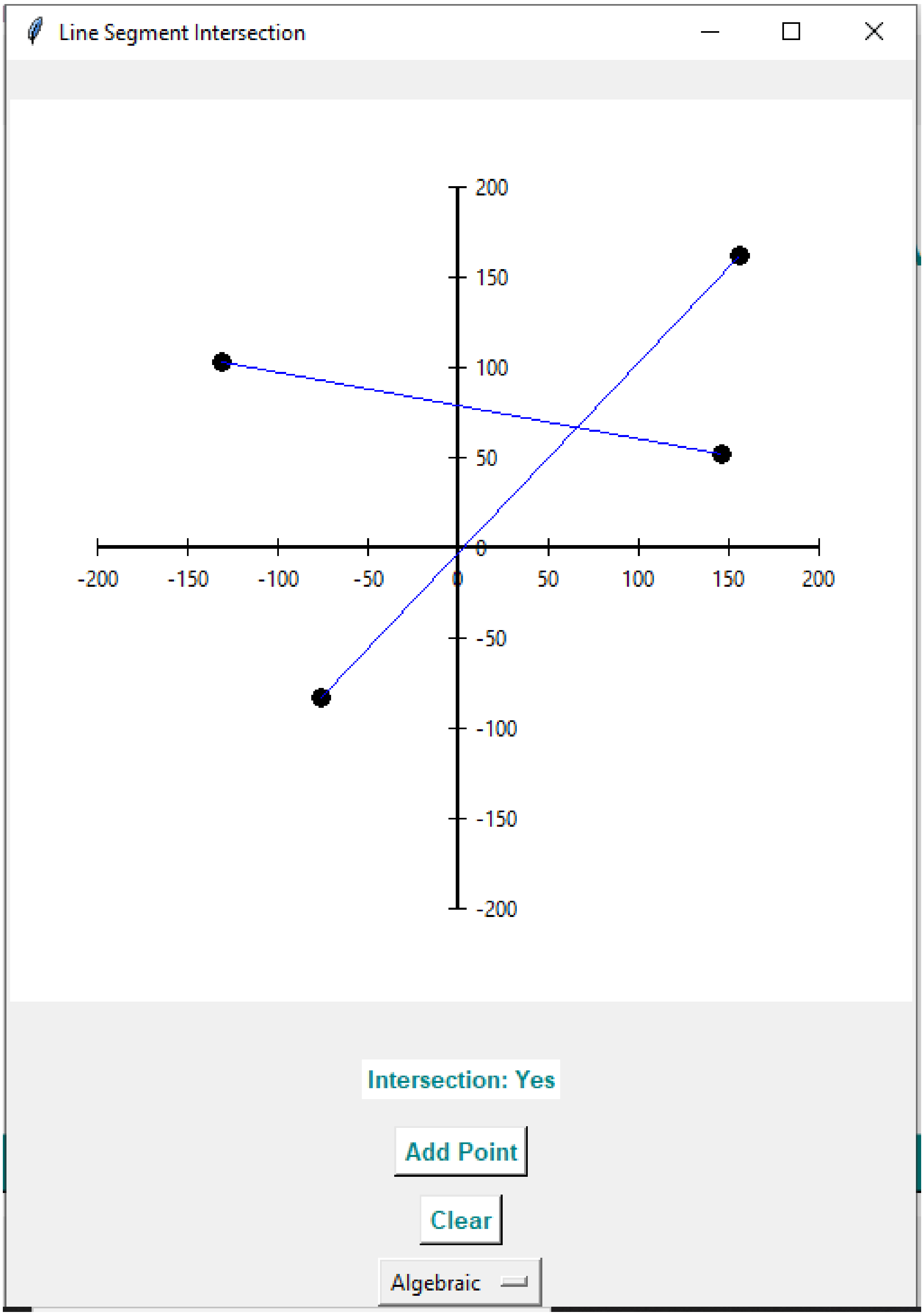
Space Complexity: O(n), -120 bytes

Figure 8: Andrew's Monotone Chain

0.7 Line Segment Algorithms

0.7.1 Algebraic Line Segment Intersection:

The algebraic method exhibits a time complexity dependent on the solution of a system of equations. The accuracy of the intersection detection is high, but the computational cost may vary.

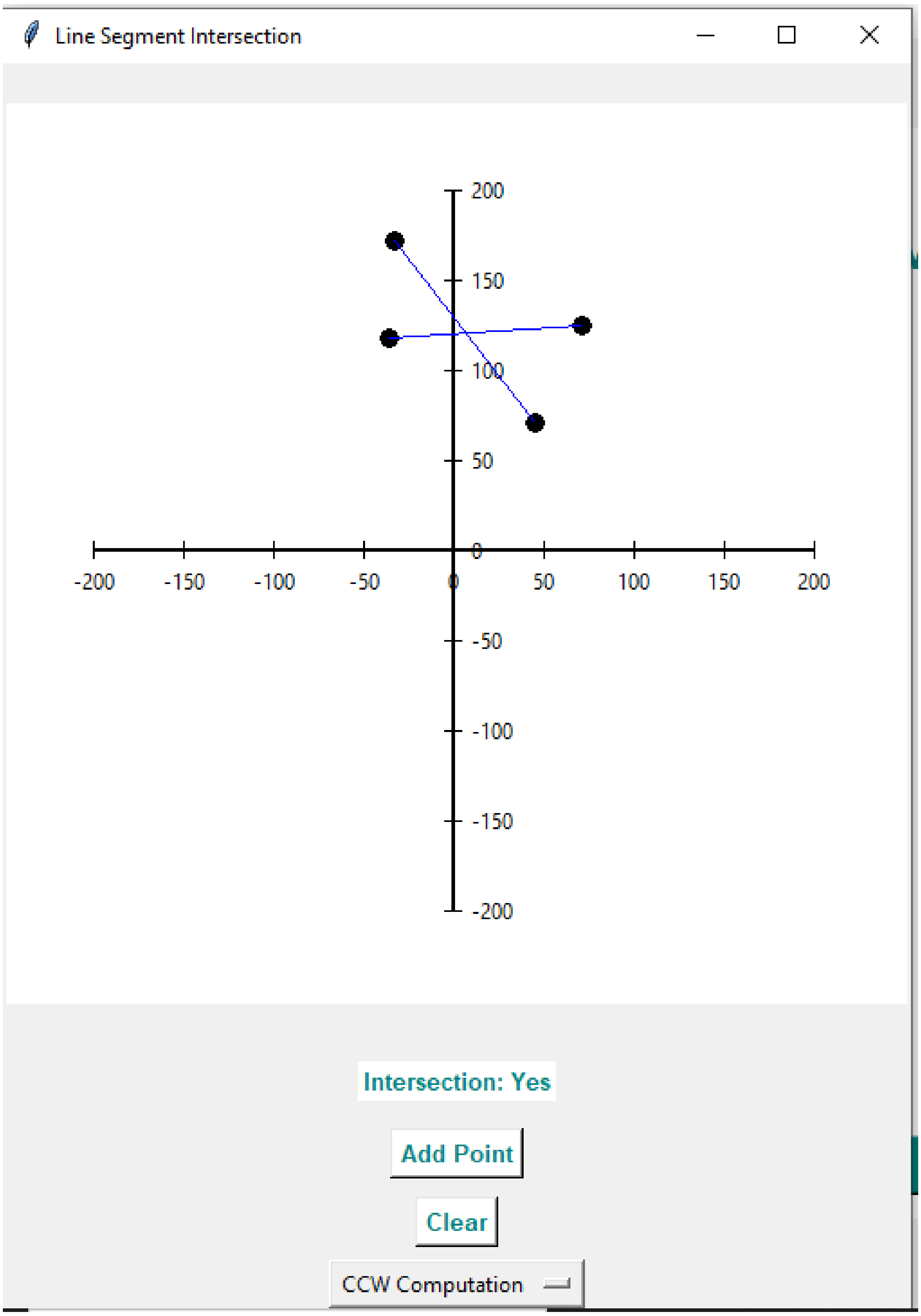


Intersection: Yes

Figure 9: Algebraic

0.7.2 CCW Computation Line Segment Intersection:

The CCW computation method provides efficient intersection detection by analyzing the orientations of point triplets. It proves to be a reliable algorithm for line segment intersection.



Intersection: Yes

Figure 10: CCW

0.7.3 Vector Cross Product Line Segment Intersection:

The Vector Cross Product method offers an alternative approach with efficient intersection detection. The algorithm's performance is comparable to the CCW computation method.

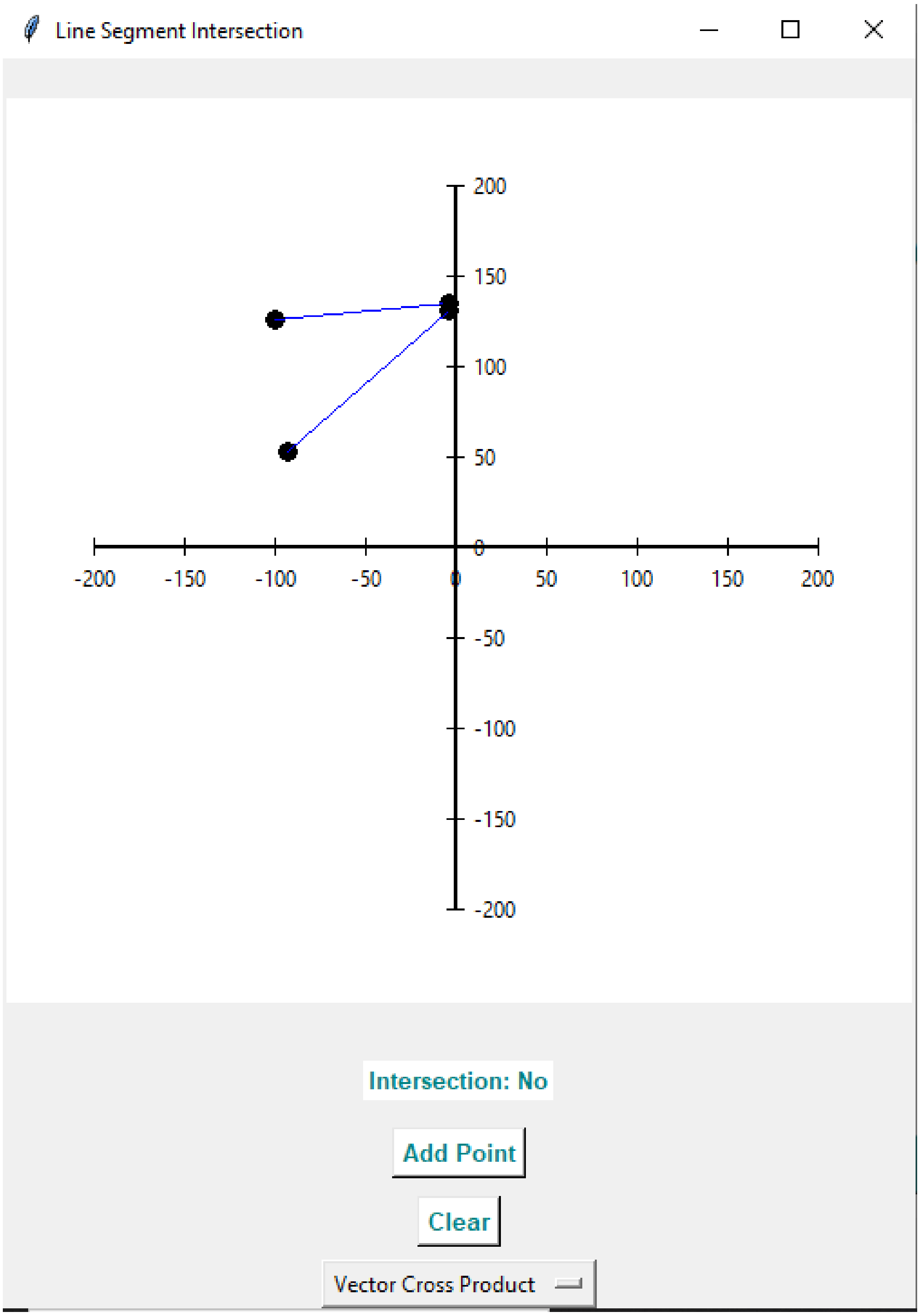


Figure 11: Vector Cross Product

0.8 CONCLUSION:

In conclusion, the choice of a convex hull algorithm depends on the specific requirements of the application. For large datasets, Graham Scan and Quick Elimination prove to be efficient, while Jarvis March and Andrew's Algorithm strike a balance between performance and simplicity. Additionally, the choice of a Line Segment Intersection algorithm depends on specific requirements and considerations. The CCW computation and Vector Cross Product methods demonstrate efficient and reliable intersection detection. The algebraic method, while accurate, may have variable computational costs. The provided GUI enhances user interaction and visualization of the algorithms.

0.9 REFERENCES:

- https://brilliant.org/wiki/convex-hull/
- https://www.youtube.com/watch?v=jjc92zWs1UAt=316spp=ygUaY29udmV4IGh1bGwgdGhyb3Vn
- https://www.youtube.com/watch v=B2AJoQSZf4Mpp=ygUaY29udmV4IGh1bGwgdGhyb3VnaCBh