Geometric Algorithm Project

By
Muhammad Hamza Altaf - 21K-4588
Sohaib Shamsi - 21K-3278

Supervised By
Dr. Nasir Uddin
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE
FAST National University of Computer and Emerging Sciences

1 Abstract

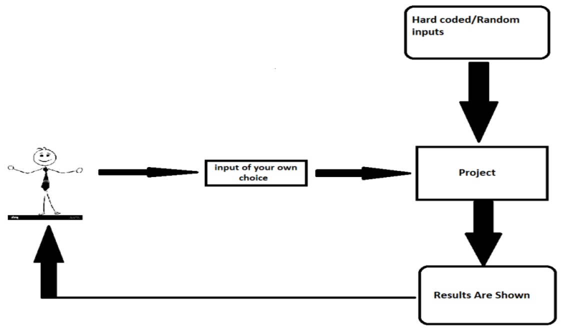
This project implements and evaluates geometric algorithms with diverse computational complexities, encompassing line segment intersection and convex hull computation through brute force, Jarvis March, Graham Scan, Quick Elimination, and a research-driven algorithm.

2 Introduction

The project delves into geometric algorithms for determining line segment intersections and computing convex hulls. It involves the implementation and comparison of various algorithms, each with distinct computational complexities.

3 Programming Design

The implementation employs Tkinter for the graphical user interface (GUI), itertools for handling combinations, random for number generation, and Matplotlib for plotting.We used Python as a programming language, system architecture, and user input methods. Include a clear diagram illustrating the system's components and interactions.



4 Experimental Setup

The GUI, built on Tkinter, enables users to draw line segments, input coordinates, and engage with algorithms. Matplotlib facilitates visualization and random point generation. User-initiated mouse clicks trigger the execution of each algorithm.

5 Results and Discussion

5.1 Line Intersection Algorithms

For detecting intersections of line segments, three algorithms are utilized:

5.1.1 CCW Check

- **Description:** This algorithm examines line segment intersections by assessing Counter Clockwise (CCW) turns. It operates with a time complexity of $O(n)$, where n represents the number of input points.

5.1.2 Determinant Method

- **Description:** The Gradient Method detects intersections by comparing slopes, and it exhibits a time complexity of $O(n)$.

5.1.3 Bounding Box

- **Description:** The Franklin Algorithm, inspired by Franklin Antonio's work, has a time complexity of $O(n \log n)$ primarily attributable to the sorting step.

5.2 Convex Hull Algorithms

5.2.1 Brute Force

- **Description:** Enables the interactive addition of points on a canvas. It exhibits a time complexity of $O(n^3)$, where n denotes the number of input points.

5.2.2 Jarvis March

- **Description:** Triggered by a button click, the algorithm advances step by step, visually presenting the construction of the convex hull. It operates with a time complexity of $O(nh)$, where n is the number of input points, and h is the number of vertices on the convex hull.

5.2.3 Graham Scan

- **Description:** Activated by double-clicking on the canvas, this algorithm utilizes the Graham Scan method to determine the convex hull. It possesses a time complexity of $O(n \log n)$.

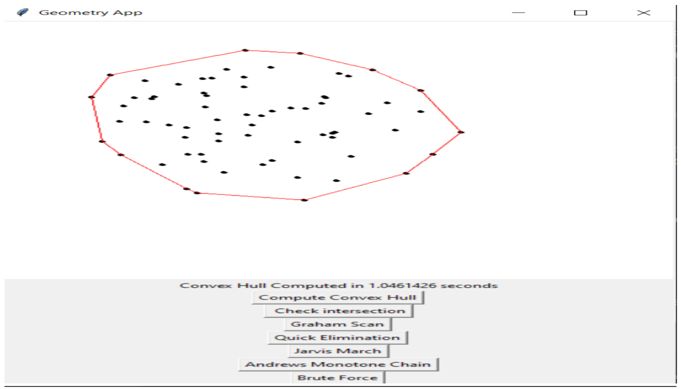
5.2.4 Quick Elimination

- **Description:** Involves the generation of random points, establishment of a rectangle, and elimination of points within that region. The time complexity relies on the number of points; if the elimination process is linear, the overall time complexity could be $O(n)$.

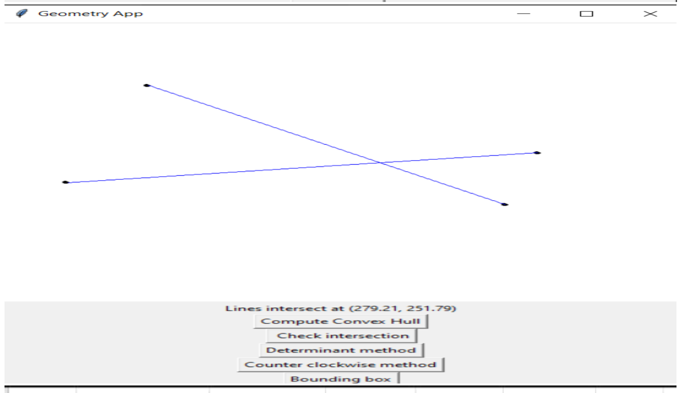
5.2.5 Research-Based Algorithm (Andrew's Monotone Chain Algorithm)

- **Description:** The algorithm works by sorting the points based on their x-coordinates (and y-coordinates in case of ties) and then constructing the upper and lower parts of the convex hull separately.

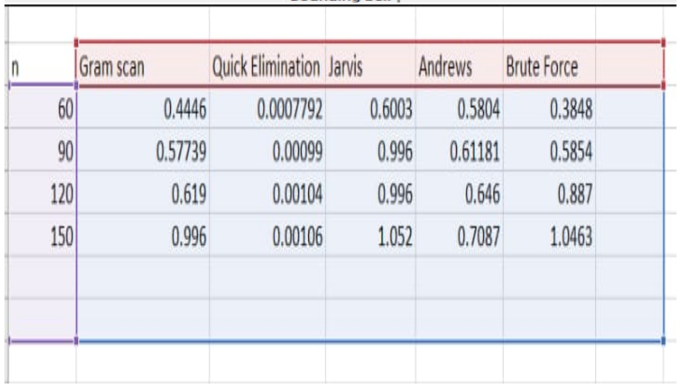
5.2.6 Screenshots, Table And Graph)



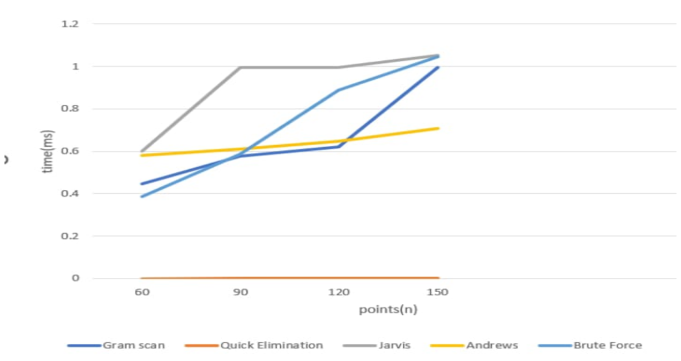
Convex Hull Computed in 1.0461426 seconds



Lines intersect at (279.21, 251.79)



n	Gram scan	Quick Elimination	Jarvis	Andrews	Brute Force
60	0.4446	0.0007792	0.6003	0.5804	0.3848
90	0.57739	0.00099	0.996	0.61181	0.5854
120	0.619	0.00104	0.996	0.646	0.887
150	0.996	0.00106	1.052	0.7087	1.0463



6 Conclusion

This project has effectively implemented and scrutinized a variety of geometric algorithms, offering a thorough comprehension of functionalities and complexities. The interactive user interfaces, detailed algorithmic steps, and complexity analyses contribute to a comprehensive exploration of these algorithms.

7 References

Online resources ChatGPT: OpenAI's language model. https://www.openai.com/gpt Geometric Algorithms - Brilliant.org: CGAL - Computational Geometry Algorithms Library: Convex Hull Algorithms - GeeksforGeeks: Convex Hull Algorithms - GeeksforGeeks

Appendix

.1 Line Segment Algorithms

.1.1 CCW Check

Extract Points return (val1 * val2 ¡ 0) and (val3 * val4 ¡ 0) and additional$_{c}onditions(line1, line2)$

.1.2 Gradient Method

line1 and line2 Calculate slopes m1 and m2 return m1 != m2 or ((x1 ¡= x3 ¡= x2 or x1 ¿= x3 ¿= x2) and (y1 ¡= y3 ¡= y2 or y1 ¿= y3 ¿= y2))

.1.3 bounding box

The bounding box algorithm determines the minimum and maximum coordinates of a set of objects in a two-dimensional space, creating a rectangle aligned with the coordinate axes that encapsulates the entire set. The algorithm initializes with the coordinates of the first object and iterates through the remaining objects, updating the minimum and maximum coordinates. The final values define the bounding box, providing a simplified representation of the spatial extent of the objects. This algorithm is commonly used in computer graphics and collision detection for efficient spatial indexing.

.2 Convex Hull Algorithms

.2.1 Brute Force

Iterate over distinct point pairs Create line between points Find left points and check condition Add left points to hull Return unique points in convex$_{h}ull$

.2.2 Jarvis March

Initialize result list with lowest point a Repeat until first point is reached Find next point q using orientation function Set last found point as q Append q to result list Return result list as convex hull

.2.3 Graham Scan

Find point p1 with lowest y-coordinate sort remaining points by polar angle Initialize stack with p1 and first two sorted points Iterate over sorted points Update stack based on polar angle Return indices of points

.2.4 Quick Elimination

Generate random points for a rectangle Calculate min and max coordinates then Print (x,y)

.2.5 Andrew's Monotone Chain Algorithm

Andrew's Monotone Chain Algorithm, a convex hull construction method, operates by sorting a set of points based on their x-coordinates and y-coordinates in the case of ties. The algorithm then proceeds to construct the upper and lower parts of the convex hull separately. In the sorting process, it ensures that the points are ordered in a monotonic fashion along the x-axis. Subsequently, the upper hull is built by scanning the sorted points from left to right, eliminating non-essential vertices to maintain convexity. The lower hull is constructed similarly but in the opposite direction. The result is a convex hull that envelops the given set of points, and Andrew's Monotone Chain Algorithm has a time complexity of O(n log n), where n is the number of input points. This algorithm finds applications in computational geometry and computer graphics for tasks such as collision detection and shape analysis.