FAST NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCE

DESIGN AND ANALYSIS OF ALGORITHM PROJECT REPORT

Group Members:

QASIM HASAN 21K-3210
MUHAMMAD HAMZA 21K-3293
MUHAMMAD TALHA SHAIKH 21K-4564

Instructor:

ANAUM HAMID

Section:

BCS-5J

Contents

1 Introduction

This project explores geometric algorithms for spatial data and computational geometry, focusing on user interaction through a visually appealing interface, with two main objectives.

- Line Segment Intersection: Performing intersection of two line segments using three different methods.
  - Orientation Based
  - Cross Product Based
  - Parametric Representation
- Convex Hull Solution: Implement and showcase various algorithms for solving the Convex Hull problem, a fundamental task in computational geometry. The project demonstrates the following Convex Hull algorithms through user-friendly visualization:
  - Brute Force
  - Jarvis March
  - Graham Scan
  - Quick Elimination
  - Monotone Chain

2 Programming Design

2.1 Checking Line Segment Intersection

2.1.1 Method 1: Orientation Based

Orientation-based line intersection checks use the sign of the cross product of vectors formed by three points. For line segments AB and CD, they intersect if the orientations of (A, B, C) and (A, B, D) are opposite signs and (C, D, A) and (C, D, B) are also opposite signs. This method determines relative positions efficiently in geometric computations.

2.1.2 Method 2: Cross Product Based

Cross product-based algorithms rely on the cross product of vectors formed by the line segments to determine their relative orientations. The sign of the cross product helps identify whether the line segments intersect or not.

2.1.3 Method 3: Parametric Representation

Parametric representation involves representing each line segment in parametric form. This method often uses parameters such as t to express points on the line segments. Intersection is then determined by checking if the parameter ranges overlap.

2.2 Convex Hull Solution

2.2.1 Brute Force

The Convex Hull Brute Force algorithm is a computational geometry method that systematically evaluates points to identify the outermost vertices of a convex hull. Despite its simplicity, it is computationally expensive and lacks efficiency compared to more advanced algorithms.

2.2.2 Jarvis March

The Jarvis March algorithm, also known as the Gift Wrapping algorithm, is a computational geometry method for convex hull construction. It starts from the lowest y-coordinate and iteratively adds the smallest polar angle. It's effective for small data sets or nearly convex shapes.

2.2.3 Graham Scan

The Graham Scan algorithm is a computational geometry method for convex hull construction. It starts with the lowest y-coordinate pivot and sorts remaining points based on polar angles. It is efficient for large data sets and general convex shapes, offering a balance between simplicity and efficiency.

2.2.4 Quick Elimination

Quick Elimination is a method that quickly identifies extreme points along the x-axis, forms an initial convex hull candidate, and eliminates interior points, making it effective for data sets with numerous interior points.

2.2.5 Monotone Chain

The Monotone Chain algorithm, also known as Andrew's Algorithm, calculates the convex hull of a set of points in a two-dimensional plane. The algorithm begins by sorting the input points based on their x-coordinates, and in cases of ties, it considers y-coordinates. Subsequently, it constructs the upper hull by traversing the sorted points from left to right, adding each point to the convex hull while ensuring a left-turn orientation. The lower hull is then built by traversing the sorted points from right to left in a similar fashion. Finally, the upper and lower hulls are combined, excluding duplicate endpoints, to yield the convex hull of the entire point set. The algorithm exploits the property that the upper and lower hulls can be independently constructed and then merged to efficiently determine the convex polygon that encompasses the given points.

3 Experimental Setup

Setup of our programs is simple user just have to click in the boundary on the screen to input points after inputting desired points the user can select any three line intersection methods and all 5 convex hull methods which are all available in our program front page. For the user interface, we employed HTML and CSS technologies to create an intuitive and visually appealing environment. The HTML structure provides the backbone, while CSS styles enhance the layout and aesthetics, ensuring a seamless user experience.

Figure 1: HTML CSS AND JAVA

JavaScript is utilized in our geometric algorithms project, ensuring efficiency and scalability. The website's logic is written in JavaScript, seamlessly integrated into the HTML/CSS framework for a visually appealing user experience. The combination of Paper.js and HTML5 Canvas simplifies geometric element rendering.

4 Result and Discussion

Line Intersection Algorithms

I.Orientation Based

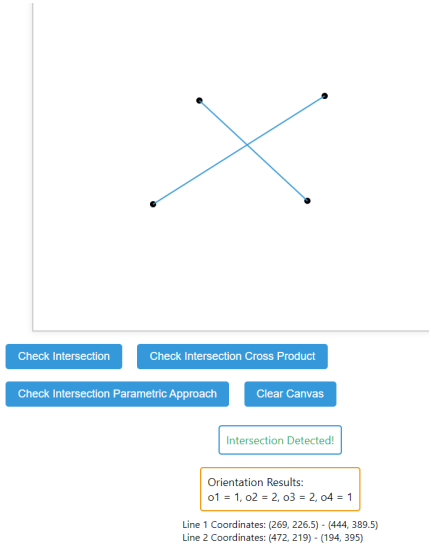

Figure 2: Orientation Based

Space Complexity: $O(1)$
Time Complexity: $O(1)$ since we are working for 2 line intersection
Benefits:

- These algorithms are often more efficient in practice, especially for specific geometric problems like line segment intersection checks.
- The constant time complexity per comparison makes them suitable for real-time applications and large datasets.

Disadvantages:

- Sensitivity to numerical precision can be a concern, potentially leading to inaccuracies or errors in certain cases.
- While effective for certain geometric problems, orientation-based algorithms may not be suitable for all types of geometric computations. Their efficiency depends on the problem at hand.

II. Cross Product Based

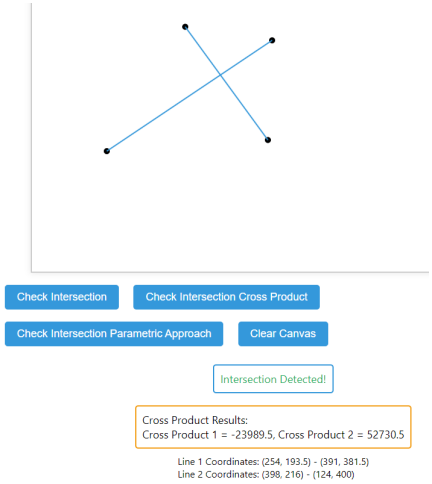

Figure 3: Cross Based Product

Figure 4: Parametric Representation

Space Complexity: $O(1)$ since we are working for 2 line intersection
Time Complexity: $O(1)$
Benefits:

- Relies on cross product computations for convex hull determination.
- Demonstrates efficiency for certain geometric arrangements.

Disadvantages:

- Requires sorting of points, contributing to the time complexity.
- Sensitivity to collinear points may impact performance.

III. Parametric Representation

Space Complexity: $O(1)$
Time Complexity: $O(1)$ since we are working for 2 line intersection
Benefits:

- Utilizes parametric representation for convex hull calculation.
- Can handle various geometric configurations efficiently.

Disadvantages:

- Complexity may increase for certain geometric arrangements.
- Sensitivity to input data characteristics.

Convex Hull Algorithms

I. Brute Force

Space Complexity: $O(1)$ (constant space)
Time Complexity: $O(n^3)$ in the worst case (where n is the number of points)

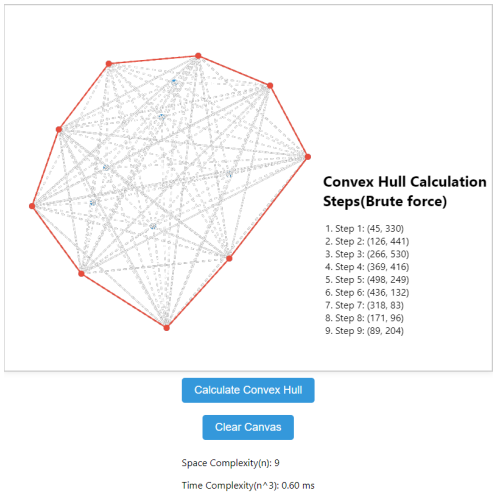

Figure 5: Brute force

Benefits:

- Simple and easy to implement.
- Guaranteed to find the correct solution.

Disadvantages:

- Inefficient for large data sets due to its quadratic time complexity.

II. Jarvis March (Gift Wrapping)

Space Complexity: $O(n + h)$ (constant space)
Time Complexity: $O(nh)$ in the worst case (where h is the number of points on the convex hull)

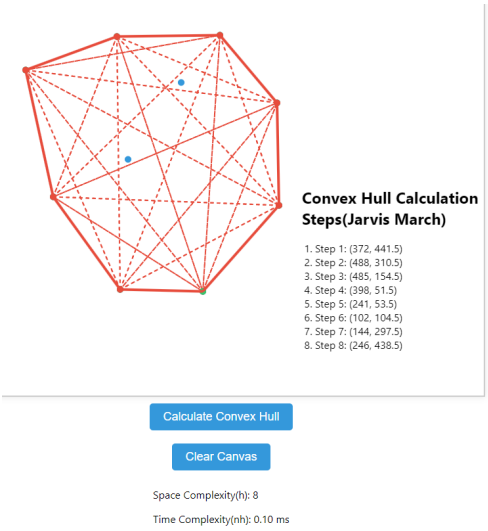

Figure 6: Jarvis Match

Benefits:

- Works well for a small number of points on the convex hull.

Disadvantages:

- Can be inefficient for dense point sets.
- Relatively high time complexity compared to other convex hull algorithms.

III. Graham Scan

Space Complexity: $O(n)$ (linear space for the sorted points)
Time Complexity: $O(n \log n)$ in the worst case
Benefits:

- More efficient than Jarvis March for most cases.
- Can handle larger data sets.

Disadvantages:

- Requires sorting of points, which adds to the time complexity.

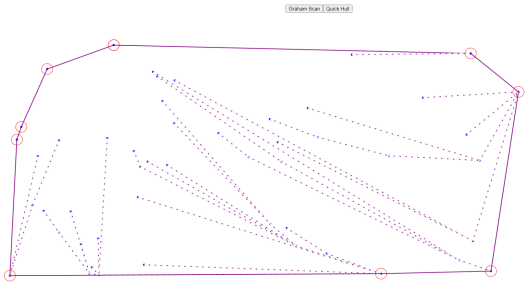

Figure 7: Graham scan

IV. Quick Elimination

Space Complexity: $O(n)$ (linear space for the set of points)
Time Complexity: $\theta(n \log n)$ in the average case (where n is the number of points) where as worst case is $O(n^2)$

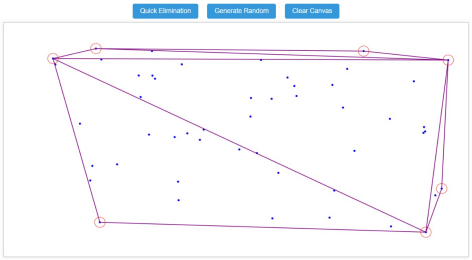

Figure 8: Quick Elimination

Benefits:

- Efficiently reduces the set of points by eliminating unlikely convex hull candidates.
- Particularly effective for data sets with many interior points.

Disadvantages:

- Requires careful handling of extreme points identification and candidate formation.
- May not be as straightforward to implement as simpler algorithms.

V. Monotone Chain Algorithm

Space Complexity: $O(n)$ (linear space for the set of points)
Time Complexity: $O(n \log n)$ in the worst case (where n is the number of points)

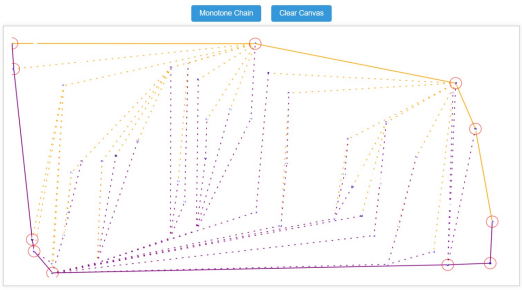

Figure 9: Monotone Chain

Benefits:

- Efficiently computes the convex hull of a set of points in a two-dimensional plane.
- Takes advantage of the monotonicity property, simplifying the hull construction process.

Disadvantages:

- Assumes distinct input points and may require additional handling for duplicate points.
- The algorithm's efficiency relies on proper sorting of input points, which may add overhead.

5 Conclusion

The research paper delves into the analysis of five convex hull algorithms: Brute Force, Jarvis March, Graham Scan, Quick Elimination, and Monotone Chain. Each algorithm is scrutinized for its strengths and applications in the realm of computational geometry. Brute Force, while conceptually simple, is noted for its computational intensity. Jarvis March proves effective for a limited number of points, while Graham Scan exhibits efficiency when dealing with larger data sets. Quick Elimination stands out for its effectiveness in data sets featuring numerous interior points. Notably, Monotone Chain, with its ability to efficiently compute the convex hull in a two-dimensional plane by taking advantage of monotonicity, emerges as a valuable addition to the set of algorithms explored. The research paper further extends its investigation to three line intersection methods: Orientation Based, Cross Product-Based, and Parametric Representation, providing a comprehensive overview of various techniques in computational geometry.

6 References

Line Intersection

References

[1] Parametric Representation Algorithm. https://www.sciencedirect.com/topics/computer-science/parametric-form

Convex Hull Algorithms

References

[1] Monotone Chain Algorithm. https://www.sciencedirect.com/science/article/abs/pii/0020019079900723

[2] Jarvis March Algorithm. https://www.codingninjas.com/studio/library/convex-hull-jarvis-s-algorithm

[3] Graham Scan. https://www.geeksforgeeks.org/convex-hull-using-graham-scan/

[4] Quick Elimination. https://www.geeksforgeeks.org/quickhull-algorithm-convex-hull/