Convex Hull and Line Intersection Algorithms

Abstract

Geometric algorithms play a crucial role in various computational tasks, offering solutions to problems such as line segment intersection and convex hull computation. In this project, we implemented and analyzed several geometric algorithms, focusing on their time and space complexities. The algorithms include methods for determining the intersection of two line segments and computing the convex hull using brute force, Jarvis March, Graham scan, Quick Elimination, and Andrew's Monotone Chain.

The project Uses a user-friendly interface for input, allowing users to interactively draw geometric objects. The algorithms' step-by-step execution is visually presented to enhance understanding. Additionally, we discuss the computational complexities of each algorithm, comparing their performance through execution time analysis.

The report provides insights into the design and implementation details of the project, including the programming language used and system architecture. Experimental setups and results, including screenshots, execution times, and comparative analyses of various algorithms, are presented. The conclusion summarizes the findings and highlights the strengths and limitations of each algorithm.

1 Introduction

This project explores fundamental geometric algorithms with a focus on line segment intersection and convex hull computation. For line segment intersection, we implement two algorithms: one using counterclockwise angle and the other utilizing the bounding box check method. These techniques find applications in computer graphics, geographic information systems, and robotics.

In the realm of convex hull computation, we investigate various algorithms, including brute force, Jarvis March, Graham scan, Quick Elimination, and Andrew's Monotone Chain. The goal is to offer a comprehensive understanding of design, implementation, and performance.

The project incorporates a user-friendly interface enabling interactive drawing of geometric objects, facilitating convex hull calculation and line intersection checks. The step-by-step execution of algorithms is visually demonstrated through a gif image, enhancing comprehension.

Our report discusses design choices, experimental setups, and results, including screenshots and execution time analyses. The investigation showcases practical applications, highlighting trade-offs and efficiencies in each algorithm.

In summary, this project delves into the intricacies of geometric algorithms, providing practical insights into their application and efficiency in solving computational problems.

2 Programming Design

In this extensive project, we utilized Python for its versatility and a rich set of tools to implement various geometric algorithms. Our user interface, created with tkinter, provides a straightforward platform for users to draw points and lines interactively. The computational part covers a range of algorithms, including those for line intersection and convex hull computation.

For line intersection, we explored two different methods: one analyzes the angles formed by lines, and the other uses a bounding box check. This helps us determine if lines cross each other, crucial for applications like computer graphics or mapping.

When it comes to the convex hull, we examined a collection of algorithms. The brute force method considers all possibilities, Jarvis March involves a gift-wrapping approach, Graham scan sorts points efficiently, Quick Elimination rapidly narrows down possibilities, and Andrew's Monotone Chain handles sets of points with a step-by-step process.

To visually depict these algorithms' progression, we used matplotlib to illustrate each step dynamically. Each algorithm's results were plotted, offering a clear understanding of how the math unfolds. This integrated system seamlessly incorporates user interaction, algorithmic computations, and visualization components, as depicted in the informative system diagram presented in Figure 01.

3 Experimental Setup

The experiments were conducted using the following software environment:

- IDE: Visual Studio Code
- Operating System: Windows 10
- Python Version: 3.12
- Libraries: Tkinter for the graphical user interface, Matplotlib for visualization

The use of Python and relevant libraries allowed for a flexible and efficient implementation of the geometric algorithms.

4 Results and Discussion

In evaluating the implemented geometric algorithms, we focus on the time complexities of convex hull and line intersection methods.

4.1 Convex Hull Algorithms

The convex hull algorithms—Brute Force, Jarvis March, Graham Scan, Quick Elimination, and Monotone Chain—exhibit varying levels of efficiency. The Monotone Chain algorithm emerges as the most efficient, thanks to its optimized step-by-step process for computing the convex hull. Quick Elimination follows closely, leveraging strategic optimizations. Graham Scan demonstrates good efficiency, particularly due to its effective point sorting. Jarvis March provides a moderate level of efficiency, employing a gift-wrapping strategy. Brute Force,

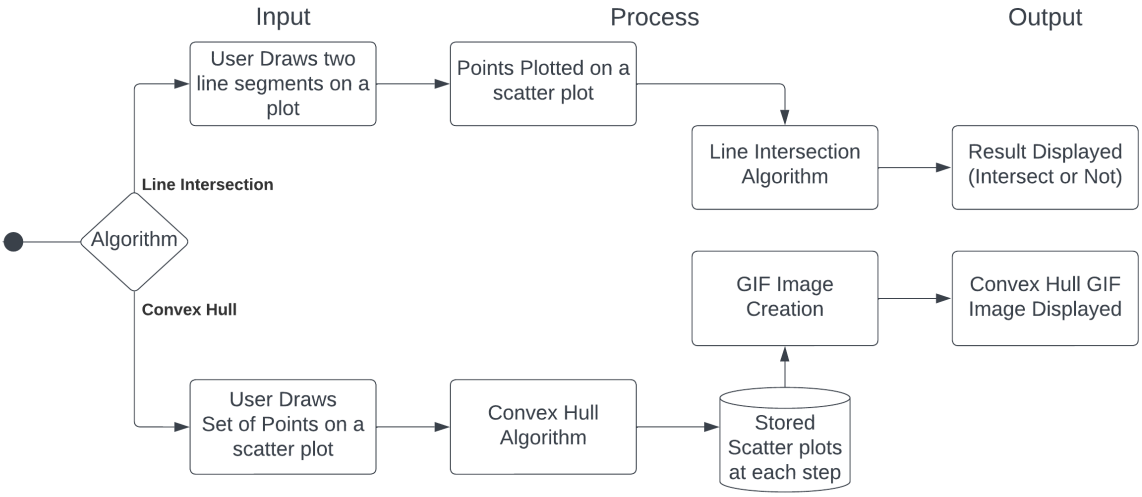

Figure 1: System Diagram

while straightforward, lags in efficiency due to its exhaustive nature.

4.2 Line Intersection Algorithms

In the realm of line intersection algorithms, the bounding box check method stands out as the most efficient. It excels in scenarios where a quick decision on line intersection is crucial. The counterclockwise angle method, while accurate, introduces additional trigonometric calculations, resulting in a moderate level of efficiency.

4.3 Input-Output Demonstration

Figure 2 demonstrates the Brute Force Convex Hull Output GIF last step plot that is created after the user submits the points drawn onto the graph as input, This visual representation of output truly extends the comprehension of the algorithms.

On the Other Hand, Figure 3 demonstrates the scatter plot of the two lines drawn by the user as input and the clockwise angle between one line and one point of the other line, This plot clearly depicts whether the two lines are intersecting or not

5 Conclusion

In summary, this project illuminates the intricacies of geometric algorithms, emphasizing their significance in computational problem-solving. Through the implementation and analysis of various methods for line segment intersection and convex hull computation, we gained insights into their time complexities and efficiencies. The user-friendly interface, coupled with visual representations of algorithmic steps, enhances accessibility and understanding. Monotone Chain proved most efficient for convex hulls, while the bounding box check method excelled in line intersection.

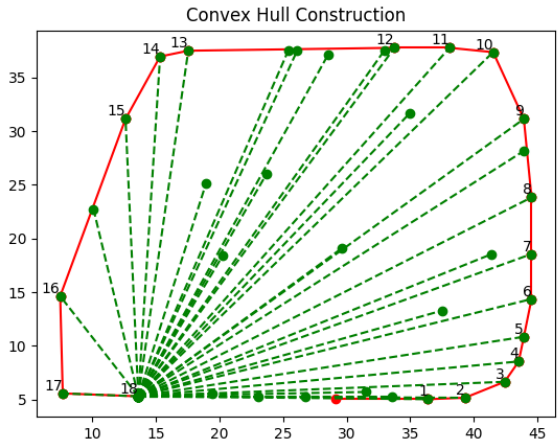

Figure 2: Convex Hull Construction Using Brute Force

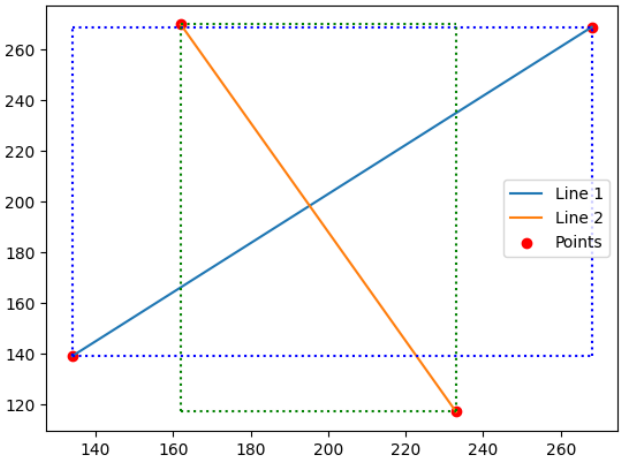

Figure 3: Line Intersection Using Bounding Box Check