Algorithms for determining the intersection of lines and a comparison of algorithms for computing convex hulls

1st Uzair Shahzad
National University of Computer and Emerging Sciences
Karachi, Pakistan
k213263@nu.edu.pk

2nd Talal Abdullah
National University of Computer and Emerging Sciences
Karachi, Pakistan
k213259@nu.edu.pk

3rd Umair Baig
National University of Computer and Emerging Sciences
Karachi, Pakistan
k213313@nu.edu.pk

***Abstract*—This research delves into the implementation and performance evaluation of convex hull algorithms, as well as methods for identifying intersections between line segments within the realm of computational geometry. The initial section of the paper provides an overview of the 'Sweep Line' algorithm, a technique employed for detecting intersections among line segments. The examined convex hull algorithms encompass classical approaches, such as the Brute Force method, Graham Scan, Jarvis March, Point Elimination, and Quick Hull. Implemented in the Python programming language, each algorithm undergoes thorough implementation, with a specific focus on calculating execution times. The primary goal is to quantify the computational efficiency of these algorithms in delineating convex hulls from arbitrary sets of points. Through a meticulous comparative analysis of execution times, this study seeks to provide valuable insights into the relative computational efficiencies of each algorithm, thereby contributing to the field of knowledge in computational geometry.**

I. Introduction

Situated within the expansive realm of computational geometry, this research project delves into the intricate implementation and empirical evaluation of algorithms pivotal to solving geometric problems. The focal points are two distinct categories of algorithms—convex hull determination and line segment intersection—each addressing fundamental challenges within their respective domains.

The initial exploration centers around the Sweep Line algorithm, a crucial methodology dedicated to identifying intersections among line segments. Recognized for its efficiency in handling various line segment configurations, the Sweep Line algorithm plays a pivotal role in applications such as computer graphics and geographical information systems.

The computation of convex hulls presents a foundational challenge in computational geometry, with diverse applications spanning computer graphics to geographic information systems. This research project meticulously implements and empirically evaluates five distinct convex hull algorithms, employing the Python programming language.

The scrutinized algorithms include the classical Brute Force methodology, valued for its simplicity but challenged by scalability; the Graham Scan, utilizing angular sorting for expedited convex hull determination; Jarvis March, a gift-wrapping algorithm known for its intuitive approach; Quick Elimination, employing a divide-and-conquer strategy; and Quick Hull, a hybrid algorithm combining divide-and-conquer and incremental techniques.

Motivated by the imperative need for a nuanced understanding of the computational efficiencies of these algorithms, our research endeavor involves the meticulous implementation of each algorithm and subsequent measurement of their execution times. This empirical approach transcends theoretical considerations, providing a rigorous comparative analysis to discern practical performance disparities among the algorithms. The study's outcomes are poised to contribute essential insights to the computational geometry domain, guiding both researchers and practitioners in the informed selection of algorithms tailored to specific application requirements.

II. Your programming design

A. Convex Hull Algorithms

Both implementations utilize the Python programming language, and for visualizing algorithm results, the well-known Python plotting package "Matplotlib" is employed. The implementation approach for all algorithms is iterative, with the exception of the Quick Hull algorithm, which adopts a recursive nature. Here's a brief overview of the algorithms discussed in this paper:

1) Brute Force: This approach is not an algorithm but rather a trial-and-error approach to solving the convex hull problem. A random point is selected from the given set of points, and then by comparison with every other point in the set, it is determined whether the selected point should be part of the hull or not. Complexity of this algorithm is $O(N^3)$

2) Jarvis March: This approach helps deselect the points by starting with the smallest y-coordinate point and checks for the next point having a counter-clockwise angle with the previous point. The final result gives the set of points that should be part of the hull. Complexity of this algorithm is O(Nh)

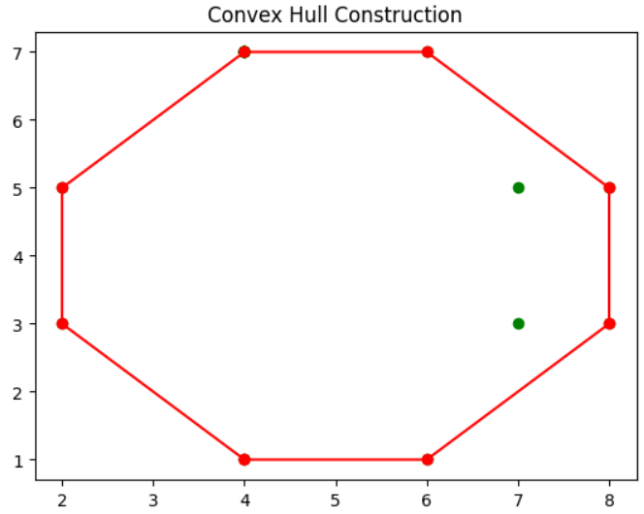

Fig. 1. Jarvis March Visualization

3) Graham Scan: Similar to Jarvis March, this approach also takes the smallest y-coordinate point as the first point and then points having counter-clockwise angles to the previous point are included in the hull. The only difference is that points not having a counter-clockwise angle are removed from the set as the iterations proceed. Complexity of this algorithm is O(Nlog(N))

4) Quick Elimination: Diverging from the discussed approaches, this algorithm follows a distinctive path by initially eliminating non-hull points and subsequently determining the hull on the remaining points. The process commences by creating a quadrilateral using the extreme points in both coordinates. Points within the quadrilateral are then excluded from the set. Subsequently, the ray scatter algorithm is applied to eliminate points within the quadrilateral. Following point removal, any other convex hull algorithm can be employed to identify the ultimate hull. The complexity of this algorithm is O(Nlog(N)).

5) Monotone Chain: The Monotone Chain Algorithm efficiently computes the convex hull of a set of points in the plane. It employs a divide-and-conquer strategy, initially sorting the points lexicographically. The algorithm then constructs the upper and lower halves of the convex hull independently by iteratively adding points while maintaining convexity.The time complexity of the Monotone Chain Algorithm is dominated by the initial sorting step, resulting in an overall time complexity of O(N log N).

B. Line Intersection Algorithms

1) Counter-Clockwise Technique: The initial approach used entails identifying the intersection between line segments by

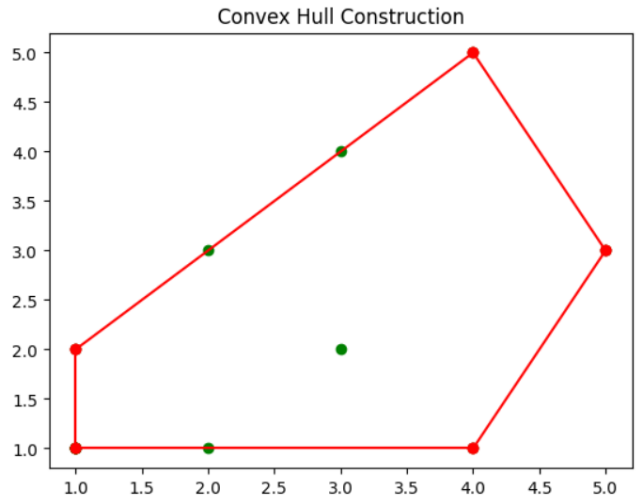

Fig. 2. Quick Elimination Visualization

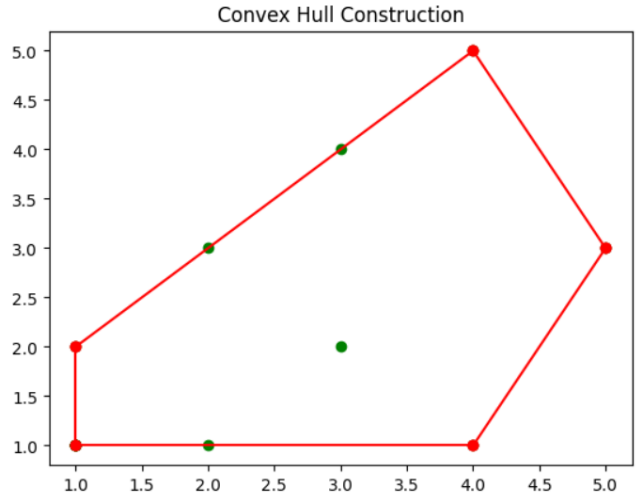

Fig. 3. Monotone Chain Visualization

evaluating their counterclockwise area from each point of one segment to the other.

2) Using Slopes: Another method implemented in this project involves identifying intersections between line segments by assessing their individual slopes. If the slopes of the two lines differ, an intersection is recognized; otherwise, they are considered non-intersecting.

3) Using Parametric Equations: This technique involves solving parametric equations of the line segments and if the results lie between 0 and 1, intersection is identified; otherwise not.

III. Experimental Setup

A. Convex Hull Algorithms

Within a specified range, the set of points is randomly produced (1000 in our experiment). An array of tuples, each containing the point's x and y coordinates, is used to represent

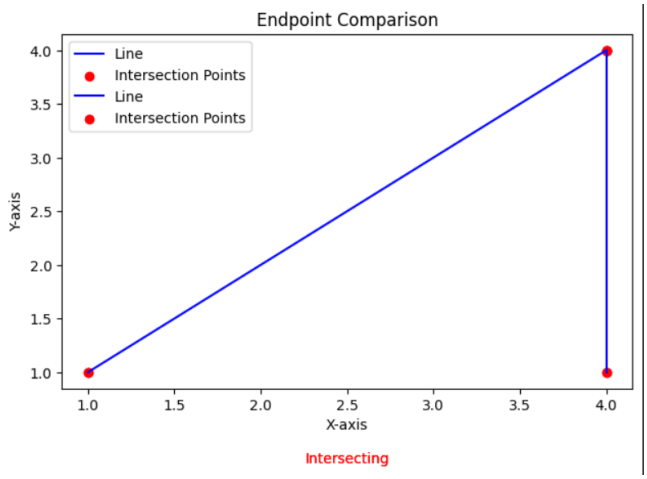

Fig. 4. Line Intersection plotting with Matplotlib

the points. The array format is the same across all implementations. The help clock function in the Python 'time' package is used to measure the execution time of all algorithms. Convex Hulls that are produced can be visually represented using the Python plotting software "Matplotlib."

B. Line Intersection Algorithms

Line segments are created with the help of '2DLine' object of Matplotlib.line. Line segments are created y clicking on the canvas of plotting figure window which appears when the program is executed. User draws two line segments by clicking four times. If the drawn line segments intersect, respective display is appeared on the cosole and vice-versa.

C. Visual Comparison of Algorithms

To get a visual representation of comparison between Convex Hull Algorithms, another plotting library 'Plotly' is used.

IV. Results and Discussion

following the visual comparison and computation of all convex hull techniques' execution times. It is evident that the Quick Hull algorithm is the top performer. Since this technique is frequently utilized in the industry, this experiment also supports the concept of applying it in contemporary computer graphics applications. The Line Intersection Algorithms used in this experiment are merely intended as an example of how different algorithms can be applied to solve the same problem.

V. Conclusion

References

[1] 1Department of Master of Computer Applications, Siddaganga Institute of Technology, Tumkur, India
[2] 2Comuter Engineering Department, School of Technology and Business Studies, Dalarna University, Borlänge, Sweden
[3] 50 M. A. Jayaram et al.: Convex Hulls in Image Processing: A Scoping Review
[4] American Journal of Intelligent Systems 2016, 6(2): 48-58
[5] Dunder M M, Fung G, Krishnapuram B, Rao R.B, Multiple Instance learning algorithms for computer –Aided detection, IEEE Transactions on Biomedical Engineering, 2008, pp 1015-1021

TABLE I
Table Type Styles

Table 1 Execution Times of Convex Hull Algortihms		
Algorithm	***No. Of Points***	***Execution Time***
Brute Force	100	4.046
	200	23.621
	300	77.025
	400	196.814
	500	145.045
Jarvis March	100	1.549
	200	1.694
	300	1.756
	400	1.757
	500	1.860
Graham Scan	100	0.231
	200	0.251
	300	0.254
	400	0.240
	500	0.258
Point Elimination	100	5.797
	200	6.511
	300	9.904
	400	29.978
	500	31.003
Monotone Chain	100	5.797
	200	0.000627
	300	0.008361
	400	0.0019426
	500	0.0160577