Geometric Algorithms

By

Abdul Rafay - 21K-4591
Fahad Yousuf - 21K-4839
Huzaifa Asad - 21K-4838

Supervised
By

Dr. Farukh Saleem
Associate Professor

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

Abstract

The provided Python code is an implementation of algorithms for computing the convex hull of a set of points in a 2D plane. The convex hull is the smallest convex polygon that encloses all the given points. The implemented algorithms include the brute force method, Jarvis March, Graham Scan, Quick Elimination combined with Graham Scan, and Monotone Chain. In this visualization tool, a Tkinter-based GUI is used to generate random points on a canvas, allowing users to apply different convex hull algorithms and observe the step-by-step visualization of how each algorithm constructs the convex hull. The tool also provides functionality to check for line intersection using both the Graham Scan algorithm and the Slope-Intercept method.

Introduction

the computation of convex hulls using various algorithms. It outlines the motivation for the project, highlighting the significance of understanding and visualizing convex hull computations. The introduction sets the stage by presenting the problem of convex hull computation, the challenges involved, and the objectives of the tool, which includes providing an interactive environment for users to explore and compare different convex hull algorithms.

Program Design

The implementation is done in Python, leveraging the Tkinter library for creating a graphical user interface. Python's simplicity and readability make it well-suited for this educational visualization tool. For Convex Hull we have used the following Algorithms:

- Brute Force
- Graham Scan
- Jarvis March(Package Wrap)
- Quick Elimination
- Research Paper method

For Line intersection we have used the following algorithms:

- Graham Scan
- Slope-Intercept

The code includes classes and functions representing geometric entities like points and convex hulls. The GUI is designed to facilitate user interaction, allowing random point generation, algorithm selection, and step-by-step visualization. A clear and intuitive diagram (implemented in Tkinter canvas) displays the points and the evolving convex hull during each step of the selected algorithm. The design decisions, such as the use of classes for points and encapsulation of algorithms, contribute to the code's readability and maintainability.

Experimental Setup

0.1 Dataset

The experimental setup involves the canvas where random points are generated by the user through the GUI. The number and distribution of these points are determined by the user's choice.

0.2 Environment

The code does not involve an extensive experimental setup in terms of hardware or external dependencies. It runs on any environment supporting Python and Tkinter.

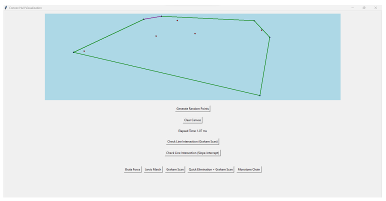
0.3 Parameters

While the user can interactively control parameters such as the number of generated points, the code's core algorithms do not have specific configurable parameters.

1 Results

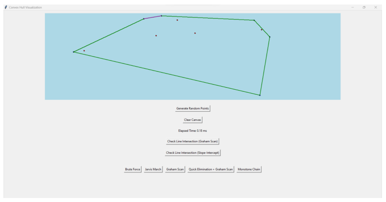
This section involves the interactive visualization of the convex hull construction process. Screenshots or visual outputs of the GUI during different stages of the algorithm execution are presented.

1.1 Brute Force


Generate Random Points
Clear Canvas
Check Line Intersection (Graham Scan)
Check Line Intersection (Slope-Intercept)
Brute Force
Jarvis March
Graham Scan
Quick Elimination + Graham Scan
Monotone Chain

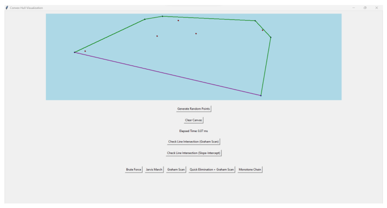
Elapsed Time: 1.07 ms

1.2 Graham Scan


Generate Random Points
Clear Canvas
Check Line Intersection (Graham Scan)
Check Line Intersection (Slope-Intercept)
Brute Force
Jarvis March
Graham Scan
Quick Elimination + Graham Scan
Monotone Chain

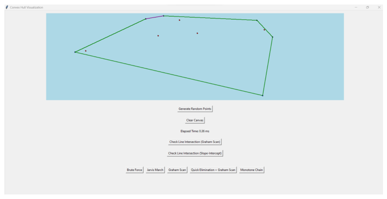
Elapsed Time: 0.18 ms

1.3 Jarvis March


Generate Random Points
Clear Canvas
Check Line Intersection (Graham Scan)
Check Line Intersection (Slope-Intercept)
Brute Force
Jarvis March
Graham Scan
Quick Elimination + Graham Scan
Monotone Chain

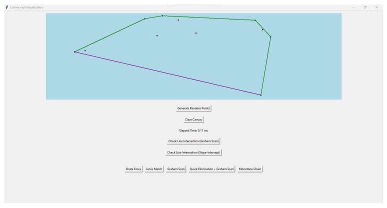
Elapsed Time: 0.07 ms

1.4 Quick Elimination


Generate Random Points
Clear Canvas
Check Line Intersection (Graham Scan)
Check Line Intersection (Slope-Intercept)
Brute Force
Jarvis March
Graham Scan
Quick Elimination + Graham Scan
Monotone Chain

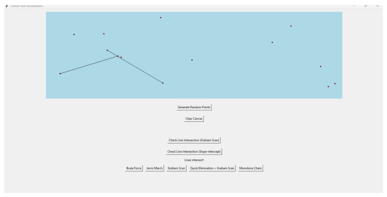
Elapsed Time: 0.36 ms

1.5 Monotone Chain


Generate Random Points
Clear Canvas
Check Line Intersection (Graham Scan)
Check Line Intersection (Slope-Intercept)
Brute Force
Jarvis March
Graham Scan
Quick Elimination + Graham Scan
Monotone Chain

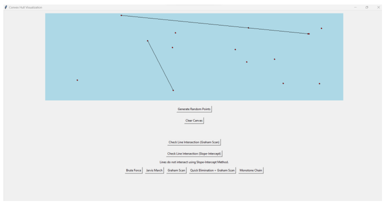
Elapsed Time: 0.11 ms

1.6 graham scan line Intersection


Generate Random Points
Clear Canvas
Check Line Intersection (Graham Scan)
Check Line Intersection (Slope-Intercept)
Brute Force
Jarvis March
Graham Scan
Quick Elimination + Graham Scan
Monotone Chain

Lines intersect!

1.7 Slope Intercept line Intersection


Convex Hull Visualization
Generate Random Points
Clear Canvas
Check Line Intersection (Graham Scan)
Check Line Intersection (Slope-Intercept)
Brute Force
Jarvis March
Graham Scan
Quick Elimination + Graham Scan
Monotone Chain

Lines do not intersect using Slope-Intercept Method.

Conclusion

The conclusion summarizes the key findings from the experiments and the insights gained through the visualization tool. It reiterates the objectives achieved and discusses the contributions of the project. Potential areas for improvement or future work may also be highlighted.

References

This Project cannot be made possible by the guidance of the following books. 1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms. MIT Press.

2. de Berg, M., Cheong, O., van Kreveld, M., & Overmars, M. (2008). Computational Geometry: Algorithms and Applications. Springer.