**REPORT – DSA Part B**

**Visualization of Geometric Algorithms**

**Name: Prarthana R USN: 1RV23IS089**

1. **Introduction**

In the realm of computational geometry, visualization plays a crucial role in understanding complex algorithmic concepts. This project implements five fundamental geometric algorithms through interactive web-based visualizations, leveraging JavaScript and HTML5 Canvas technologies. The implementation encompasses 2D Convex Hull, 3D Convex Hull, KD Tree Search, Delaunay Triangulation, and Voronoi Diagrams, providing a comprehensive platform for exploring these essential geometric computations. Through this visualization project, the gap between theoretical understanding and practical implementation is bridged.

1. **Problem Description**

Modern computational geometry applications require a deep understanding of complex spatial relationships and algorithmic processes. This project addresses the challenge of making these concepts more accessible and comprehensible through interactive visualization. Our implementation creates an engaging platform where users can directly interact with geometric algorithms through a web interface, inputting points and observing real-time results of algorithmic computations.

The project emphasizes user interaction and real-time visualization, allowing immediate feedback as users experiment with different input configurations. This approach transforms abstract geometric concepts into tangible, visual experiences, making it easier for users to grasp the underlying principles and behaviours of each algorithm. The interactive nature of the implementations ensures that users can explore various scenarios and understand how different input patterns affect the algorithmic outcomes.

**3. DSA Concepts Used**

1. **Sorting Algorithms** - The Monotone Chain algorithm sorts points primarily by their x-coordinates (and y-coordinates in case of ties), which is essential for constructing both the upper and lower hulls efficiently.

2. **Stack-Based Processing -** In the 2D Convex Hull implementation, a stack is used to keep track of points that form the convex shape as the algorithm iterates through sorted points.

3. **Recursive Tree Structures -** The KD Tree Search employs a binary tree structure for spatial partitioning. This recursive construction alternates between dimensions at each level, facilitating efficient range queries and nearest neighbour searches by organizing spatial data.

4. **Graph Theory Concepts -** Delaunay Triangulation combines principles from graph theory with geometric properties to maintain adjacency relationships. The implementation ensures that triangles formed do not contain any points inside their circumcircles, adhering to the empty circle property essential for optimal triangulation.

5. **Priority Queues –** In Voronoi Diagrams, Fortune's algorithm uses a priority queue to handle site events and circle events systematically, allowing for efficient diagram construction through event-driven processing.

**4. Implementation Details and Algorithm Workflows**

The workflow of each algorithm represents a carefully orchestrated sequence of operations. In the 2D Convex Hull algorithm, the process begins with point sorting and proceeds through two passes to construct the lower and upper hulls. The algorithm maintains hull properties through strategic point inclusion and removal, ultimately connecting the final points to form the complete convex hull.

The 3D Convex Hull workflow extends these concepts into three dimensions, beginning with facet initialization and proceeding through normal vector computation. The algorithm tests point-plane relationships to determine facet validity, building the hull through incremental triangulation and facet connection. The final visualization leverages Three.js to render the complex 3D structure.

For KD Tree Search, the workflow centers around tree construction and query processing. The algorithm recursively partitions the point set, creating a balanced spatial index structure. During range queries, the tree is traversed efficiently, examining only relevant branches based on the search region's boundaries.

Delaunay Triangulation follows an incremental construction workflow, beginning with a super-triangle initialization. Points are inserted sequentially, with the algorithm locating containing triangles, performing splits, and managing edge flips to maintain the Delaunay property. The process concludes with the removal of super-triangle elements.

The Voronoi Diagram construction workflow processes sites sequentially, building cells through careful boundary computation. The algorithm handles special cases and infinity edges, generating a complete diagram with clearly defined regions. The final visualization includes cell coloring and boundary rendering for clear region differentiation.

**5. Applications**

Geometric algorithms have diverse applications across various technical domains. In **computer graphics and gaming**, they are essential for terrain generation, collision detection, and procedural content creation, enhancing spatial relationships and environmental interactions. **Geographic Information Systems (GIS)** utilize these algorithms for spatial analysis, enabling efficient territory division and location-based services, which are crucial for urban planning and resource management. In scientific fields, they support molecular modelling and astronomical data processing, aiding in understanding spatial relationships in particle physics and weather pattern analysis. The engineering and design sectors leverage these algorithms in **Computer-Aided Design (CAD)** systems for optimizing layouts and mesh generation in finite element analysis. Finally, within **machine learning and AI**, geometric algorithms facilitate feature space partitioning and nearest neighbour searches, contributing to effective cluster analysis and pattern recognition for spatial data classification.

**6. Snapshots**

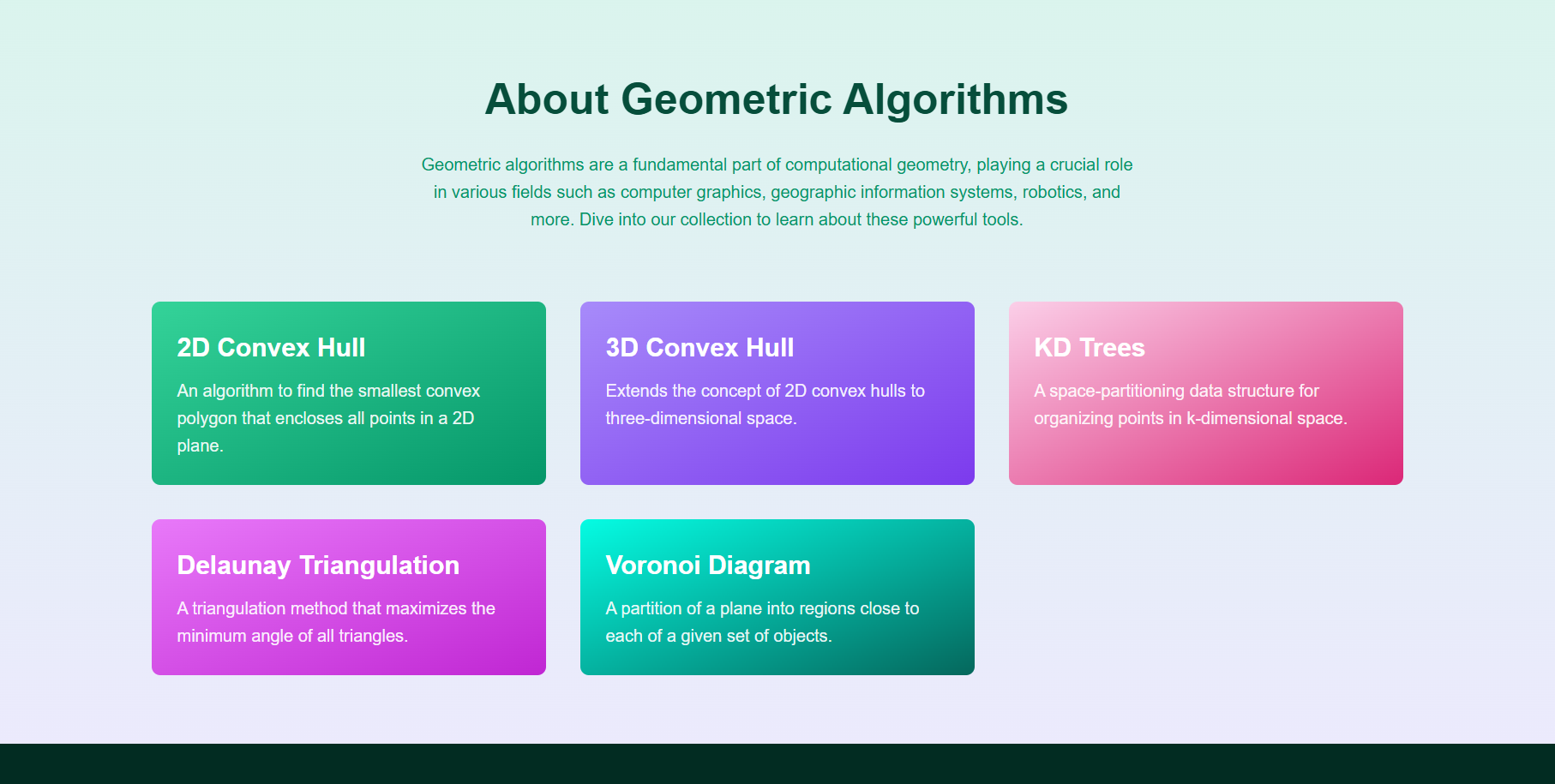
****

Fig 1. Home Page

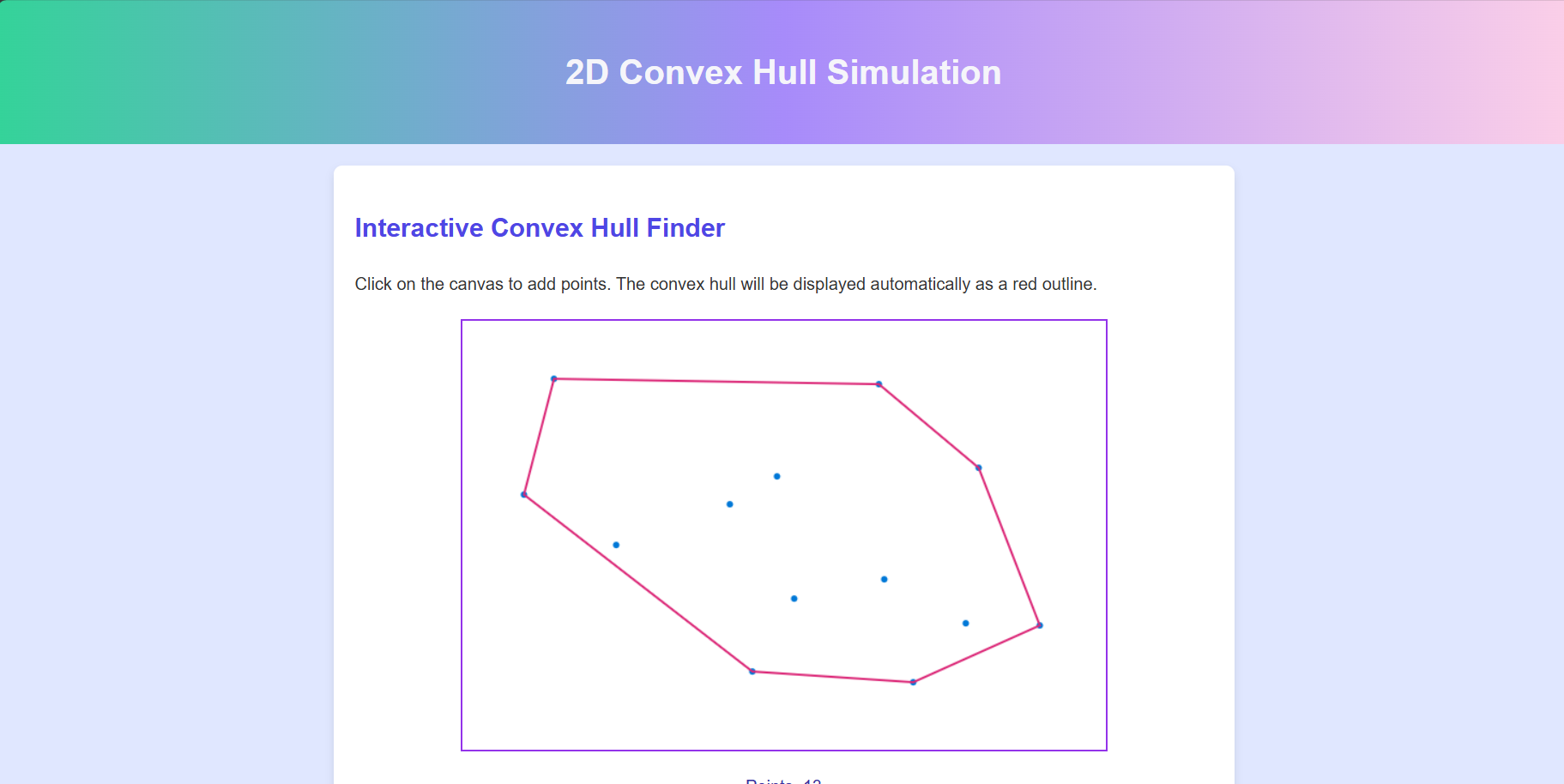
****

Fig 2. 2D Convex Hull

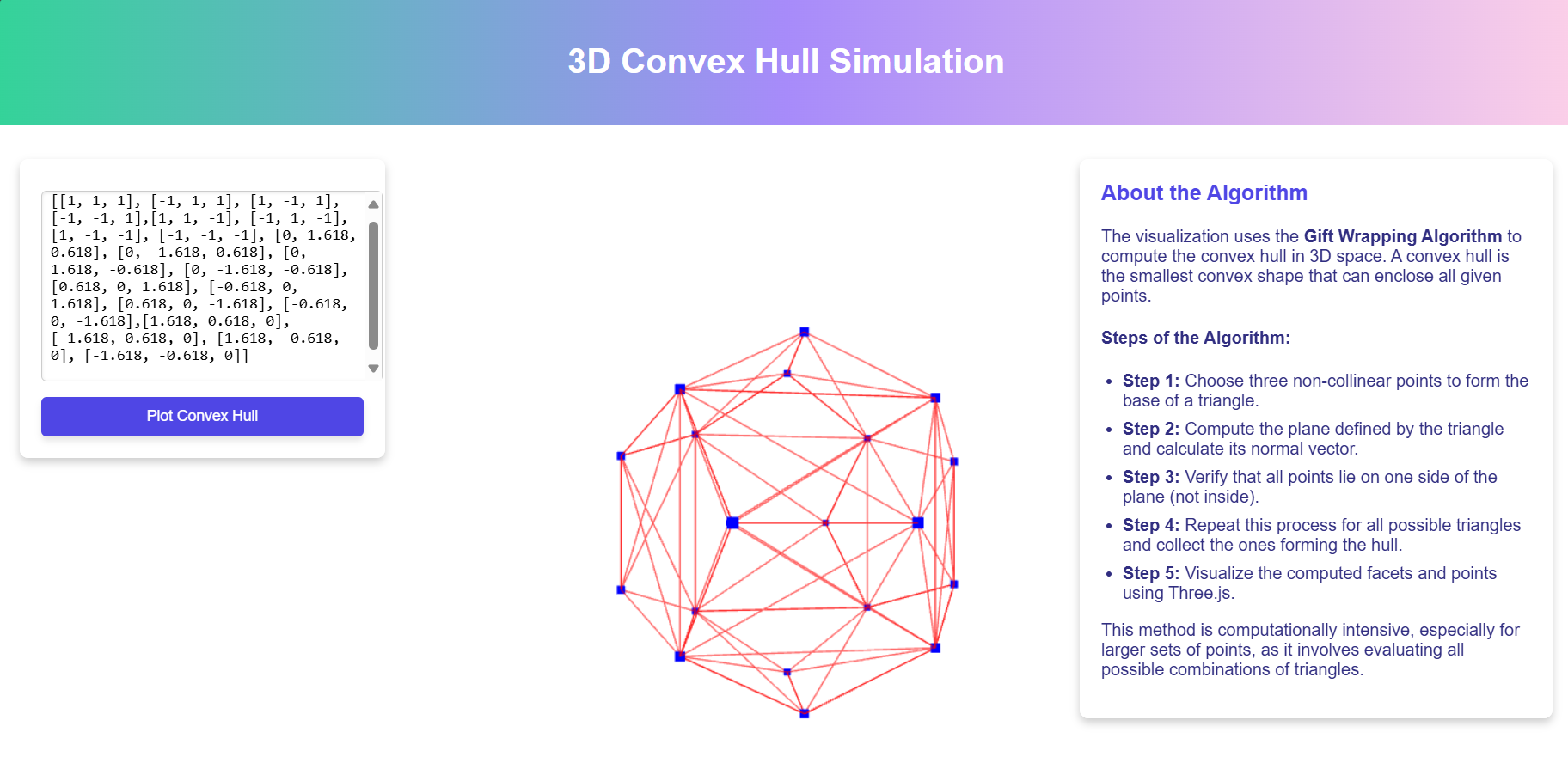
****

Fig 3. 3D Convex Hull

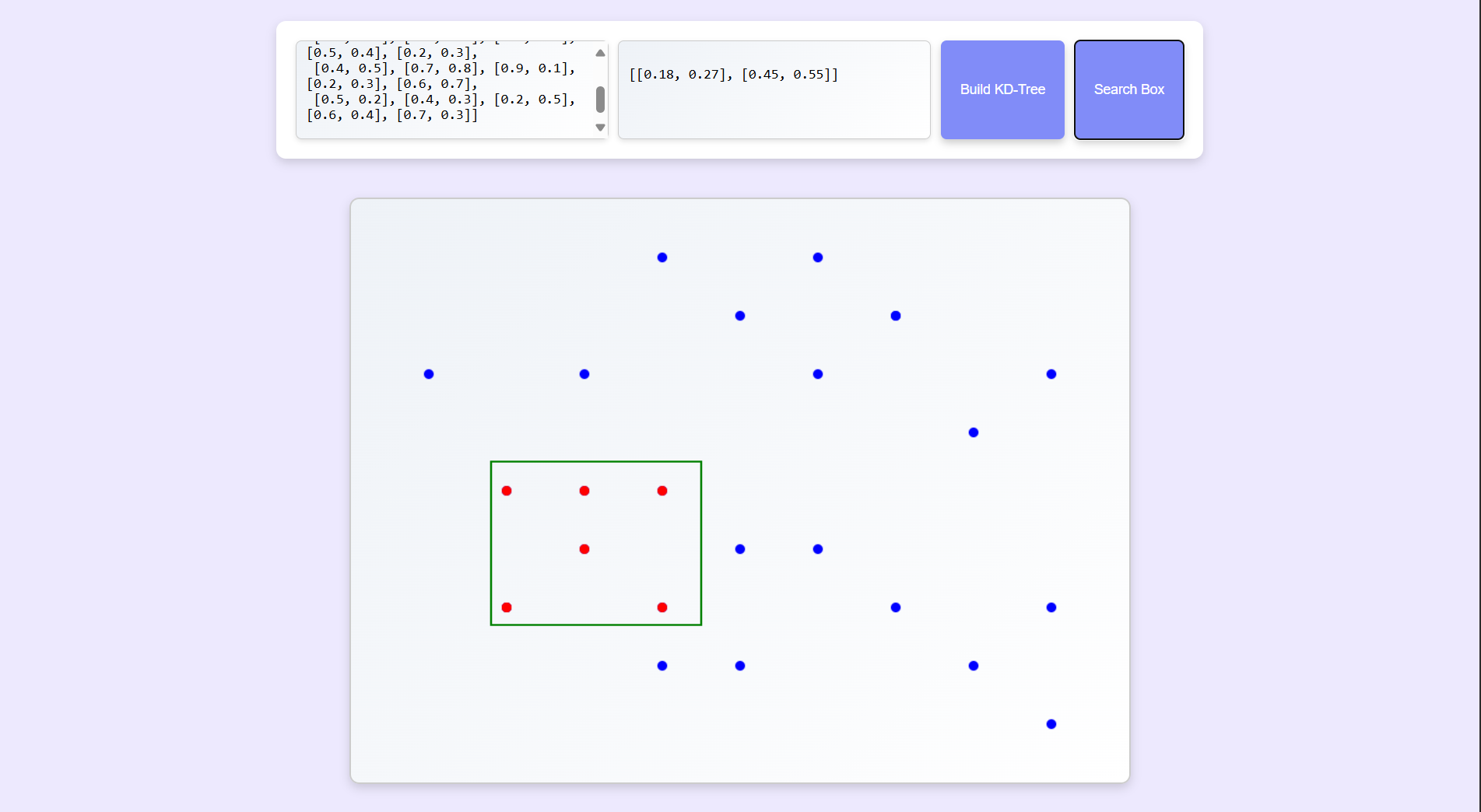
****

Fig 4. KD Tree Search

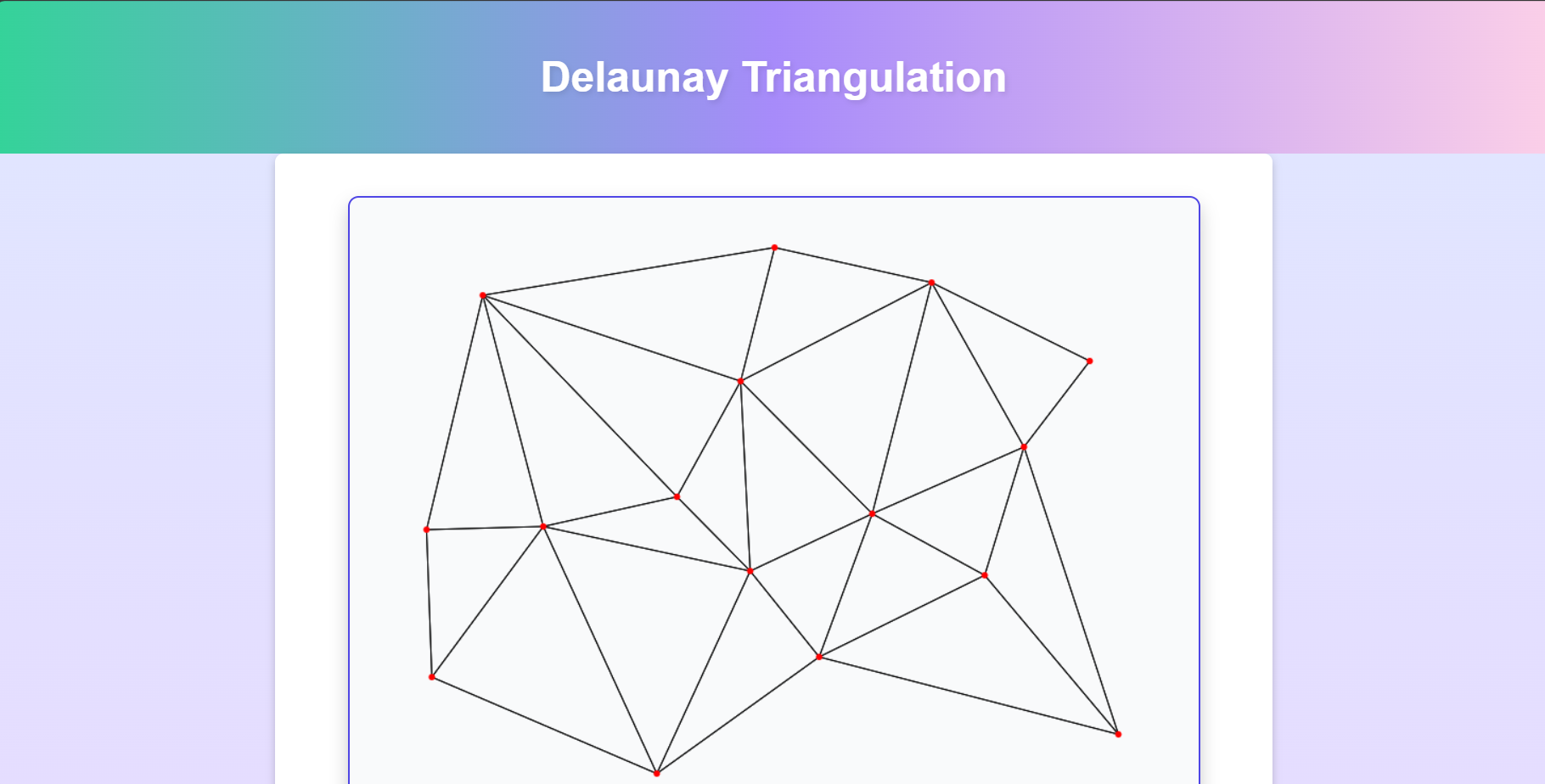
****

Fig 5. Delaunay Triangulation

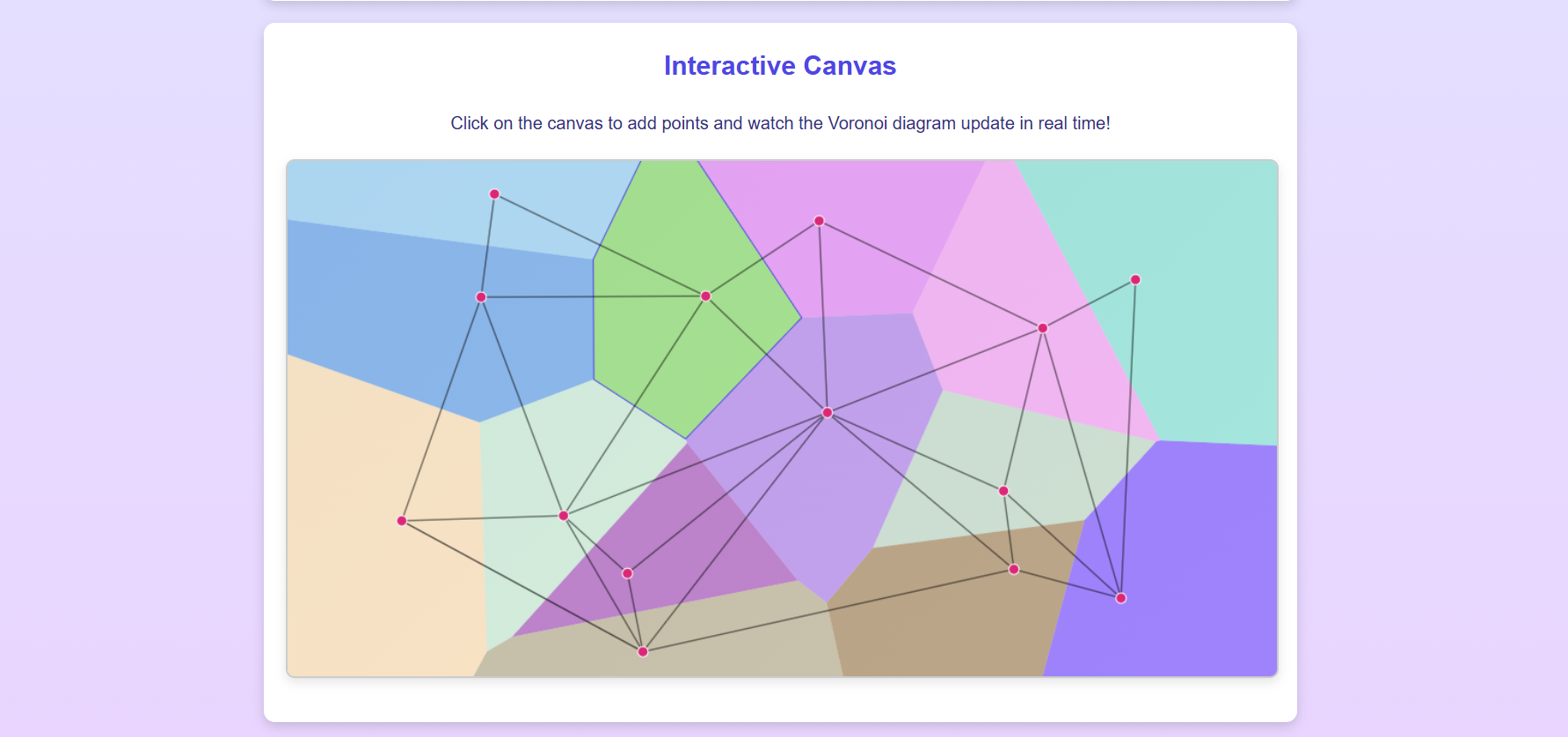
****

Fig 6. Voronoi Diagram

**7. Conclusion**

This project successfully demonstrates the practical implementation of fundamental geometric algorithms through interactive visualization. The implementations provide valuable insights into algorithm behaviour while maintaining computational efficiency and user accessibility. Through careful attention to user interface design and algorithm implementation, the project achieves its goal of making complex geometric concepts more approachable and understandable.

The interactive visualizations serve both educational and practical purposes, enabling users to experiment with different inputs and observe algorithmic behaviour in real-time. The implementations demonstrate the versatility of these algorithms across various technical fields, from computer graphics to scientific research. As computational geometry continues to evolve, these visualizations provide a solid foundation for understanding and exploring geometric algorithms in practical applications.

Future development could focus on performance optimization for larger datasets, integration with real-world data sources, and expansion to include additional geometric algorithms. The project stands as a testament to the power of visual learning in understanding complex computational concepts, while providing a practical tool for exploring geometric algorithms in action.