Final Project Presentation

Delaunay Triangulation and Voronoi Diagrams

Key Reference

- This project is inspired and informed by two pivotal papers, reflecting the integration of theoretical and practical aspects:

  i. **Dynamic Fusion Pathfinding in Robotics**

     - **Paper**: A Dynamic Fusion Pathfinding Algorithm Using Delaunay Triangulation and Improved A-Star for Mobile Robots
     - **Insight**: Demonstrates the application of Delaunay Triangulation in robotic pathfinding, guiding the practical aspect of the project.

  ii. **Fundamentals in Computational Geometry**

     - **Paper**: *Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams*
     - **Impact**: Provides foundational algorithms and concepts essential to the theoretical framework of the project.

Introduction to Computational Geometry

- **Computational Geometry**: A key area in computer science.
- **Focuses On**: Algorithms for geometric problems.
- **Applications**: Robotics, computer graphics, geographic information systems (GIS).

Delaunay Triangulation

- **What is it?**
  - A triangulation method for a set of points in a plane.
  - Maximizes the minimum angle of all triangles.
- **Properties**:
  - No point is inside the circumcircle of any triangle.
  - Helps in avoiding skinny triangles.
- **Applications**:
  - Mesh generation, numerical simulations, and pathfinding algorithms.

Voronoi Diagrams

- **What are they?**
  - Partition of a plane into regions based on distance to points in a specific subset.
- **Duality**:
  - Voronoi diagrams are the dual of Delaunay triangulation.
- **Applications**:
  - Used in meteorology, urban planning, and aviation for proximity-based problem-solving.

Project Objective

- **Primary Goal**:
  - Develop an efficient algorithm for generating Delaunay triangulation.
- **Secondary Objective**:
  - Constructing a Voronoi diagram based on the triangulation.
- **Method**:
  - Using divide-and-conquer approach for efficiency with large datasets.

Real-World Implications

- **Application in Dynamic Environments**:
  - Autonomous vehicles, drones, underwater exploration.
- **Importance**:
  - Essential for path optimization and obstacle avoidance.
- **Impact**:
  - Enhances safety and efficiency in navigation systems.

Key Observations by the Authors

- **Efficient Voronoi Diagram Computation**:
  - Algorithms for constructing Voronoi diagrams using Delaunay triangulation.
  - Achieving O(n log n) time complexity.
- **Separation of Geometrical and Topological Aspects**:
  - Importance of distinguishing between geometric calculations and topological manipulations.
- **New Data Structure: Quad Edge**:
  - A novel data structure for representing diagrams, their duals, and mirror images.

Problem Statement: Delaunay Triangulation

- **Input Specification**:
  - A list of 2D sites, each represented as a `Site` object.
- **Output Specification**:
  - A set of edges forming the Delaunay triangulation.
- **Key Challenges**:
  - Handling collinear sites and ensuring a comprehensive mesh of triangles.

Delaunay Algorithm Details

- **Divide-and-Conquer Technique**:
  - Recursively divide the set of points and merge the solutions.
- **Handling Edge Cases**:
  - Special emphasis on identifying convex hull edges.
- **Data Structure**:
  - Using a dictionary for efficient edge tracking and manipulation.

Algorithm Pseudocode: Delaunay Triangulation

- **Function `__triangulate`**:
  - Base cases for 2 and 3 points.
  - Recursive division for larger sets.
  - Merge step involving edge adjustments and validations.
- **Key Operations**:
  - Finding lower common tangent.
  - Restoring Delaunay condition with edge flipping.

Performance Analysis: Delaunay Triangulation

- **Complexity Goal**: O(n log n).
- **Divide Step**:
  - Linear time operation, O(n).
- **Conquer and Merge Steps**:
  - Constant time for small subsets.
  - Linear time for merging.
- **Overall Complexity**:
  - Product of recursion depth (O(log n)) and work per level (O(n)).

Problem Statement: Voronoi Diagrams

- **Objective**:
  - Convert Delaunay triangulation into a Voronoi diagram.
- **Input**:
  - The output edges of the Delaunay triangulation.
- **Output**:
  - A list of Voronoi edges defining cell boundaries.

Voronoi Algorithm Details

- **Efficient Conversion**:
  - Leverages Delaunay output to define Voronoi edges.
- **Handling Special Cases**:
  - Managing edges at the triangulation boundary.
- **Seamless Integration**:
  - Utilizing existing data structures from triangulation.

Algorithm Pseudocode: Voronoi Diagram

- **Function `voronoi`**:
  - Calculation of circumcenters for each triangle.
  - Mapping edges to triangles.
  - Constructing Voronoi edges between circumcenters.
- **Key Steps**:
  - Identifying adjacent triangles.
  - Defining Voronoi edges based on circumcenters.

Performance Analysis: Voronoi Diagram

- **Complexity Goal**: Aligned with Delaunay triangulation, O(n log n).
- **Key Steps**:
  - Circumcenter calculation: O(n).
  - Edge-to-triangle mapping: O(n).
  - Voronoi edge construction: O(n).
- **Overall Complexity**:
  - Combined with Delaunay triangulation, remains O(n log n).

Conclusion

- **Summary of Contributions**:
  - Development of efficient algorithms for Delaunay triangulation and Voronoi diagram generation.
  - Incorporation of innovative approaches and data structures.
- **Potential Impact**:
  - Applications in computational geometry, navigation systems, and spatial analysis.