Spherical Voronoi Diagram Plotter: Theoretical Description, Algorithm, and Complexity

Theoretical Description

A **Spherical Voronoi diagram** partitions the surface of a sphere into regions based on proximity to a set of seed points (sites) on the sphere. Each region contains all points on the sphere that are closer (in geodesic distance) to its corresponding seed than to any other. The boundaries between regions are segments of great circles.

Key Concepts

- **Seed Points**: Points on the sphere that define the Voronoi regions.
- **Voronoi Region**: For each seed, the set of points on the sphere closer to it than to any other seed.
- **Voronoi Vertex**: A point equidistant (on the sphere) to three or more seeds; forms the corners of Voronoi regions.
- **Great-Circle Arcs**: The boundaries between regions are segments of great circles, the shortest path between points on a sphere.

Spherical Voronoi diagrams are widely used in geosciences, astronomy, and any field requiring partitioning of spherical surfaces.

Algorithm Description

The project implements a Spherical Voronoi Diagram Plotter with the following core steps:

1. **Seed Point Generation**
   - *Random Generation*: Points are distributed uniformly on the sphere using random azimuthal (longitude) and polar (latitude) angles, then converted to Cartesian coordinates.
   - *Manual Addition*: Users can add points by specifying latitude and longitude, which are also converted to Cartesian coordinates.
2. **Spherical Voronoi Construction**
   - The algorithm uses the `SphericalVoronoi` class from `scipy.spatial`, which:
     - Takes as input the Cartesian coordinates of the seed points, the sphere's radius, and center.
     - Computes the Voronoi regions by determining, for each seed, the set of points on the sphere closer to it than to any other seed.
     - Calculates Voronoi vertices and the ordering of region boundaries for proper visualization.
3. **Visualization**
   - The sphere is rendered as a 3D wireframe.

  - Seed points are plotted as red dots.
  - Voronoi vertices are plotted as blue dots.
  - Region boundaries are drawn as colored arcs connecting Voronoi vertices, outlining each region.

4. **User Interface**
   - The GUI (built with Tkinter) allows users to:
     - Specify the number of seed points.
     - Generate random points or add/remove points manually.
     - Plot or clear the Voronoi diagram interactively.

Algorithmic Steps (Pseudocode)

```
Input: List of n seed points on the sphere (as Cartesian coordinates)

1. If n < 4, abort (at least 4 points are required).
2. Compute the Spherical Voronoi diagram:
    a. For each seed, determine its Voronoi region by finding all points on the sphere closer to it tha
    b. Compute Voronoi vertices (intersection points of three or more regions).
    c. For each region, sort its vertices for correct boundary plotting.
3. Plot:
    a. Draw the sphere as a wireframe.
    b. Plot seed points and Voronoi vertices.
    c. For each region, plot the boundary arcs connecting its vertices.
```

Complexity Analysis

- **Generating random points**: $O(n)$, for n seeds.
- **Spherical Voronoi construction**: $O(n \log n)$, uses convex hull on the sphere and region assignment.
- **Sorting region vertices**: $O(n)$, performed for each region.
- **Visualization (plotting)**: $O(n)$, each point, vertex, and edge plotted once.

Overall Complexity

- **Diagram Construction**: The most computationally intensive step is the construction of the Spherical Voronoi diagram, which is $O(n \log n)$ due to the convex hull computation on the sphere (as used internally by `scipy.spatial.SphericalVoronoi`).
- **Visualization and GUI**: These are linear in the number of points and regions, i.e., $O(n)$.

Space Complexity

- **Seed Points**: $O(n)$
- **Voronoi Vertices and Regions**: $O(n)$
- **Plotting Data**: $O(n)$

Summary Table

Component	**Description**	**Complexity**
Seed Generation	Random/manual placement on sphere	$O(n)$
Voronoi Construction	Spherical Voronoi via convex hull	$O(n \log n)$
Region Sorting	Ordering vertices for plotting	$O(n)$
Visualization	Drawing sphere, points, edges	$O(n)$

References

- Boissonnat, J.-D., & Cazals, F. (2002). Computing Voronoi diagrams for spheres. *Journal of Computational Geometry.*
- SciPy Documentation: `scipy.spatial.SphericalVoronoi`
- de Berg, M., van Kreveld, M., Overmars, M., & Schwarzkopf, O. (2008). *Computational Geometry: Algorithms and Applications.*