SphereStats: A Python Library for Spherical Geometry and Geospatial Analysis

Ghulam Abbas Zafari
ghulamabbas.zafari@mail.polimi.it

August 1, 2025

Abstract

SphereStats is a comprehensive Python library designed for advanced geospatial analysis and spherical geometry operations. It provides robust tools for calculating distances, routing, proximity analysis, clustering, and visualization on spherical surfaces with particular emphasis on Earth-based applications. The library is especially valuable for navigation systems, geographic data science, and spatial analytics applications requiring high precision spherical calculations.

1 Features

1.1 Distance Calculations

- Accurate great-circle distances using the Haversine formula
- Efficient point-to-line and point-to-polygon distance evaluations
- Precise midpoint computation for optimal route planning

1.2 Routing and Network Analysis

- Optimized shortest path computation using Dijkstra's algorithm
- Flexible intermediate waypoint generation
- Accurate travel distance estimation along spherical arcs

1.3 Geometric and Statistical Tools

- Centroid and bounding circle computation for multiple spherical points
- Efficient convex hull computation algorithms
- Advanced point clustering based on geodesic thresholds

1.4 Proximity Analysis

- Dynamic buffer zone generation around geographic features
- High-performance distance threshold queries for geofencing applications

1.5 Visualization Capabilities

- High-quality plotting of routes, convex hulls, and heatmaps
- Great-circle arcs and isochrone maps rendering with customizable styles

1.6 Advanced Metrics

- Spherical triangle area and internal angle computations
- Isochrone map generation for time-accessibility modeling

2 Visual Examples

Figure 1: Newyork$_{r}eachableareas$

Figure 2: Isochrones around New York City showing travel time contours

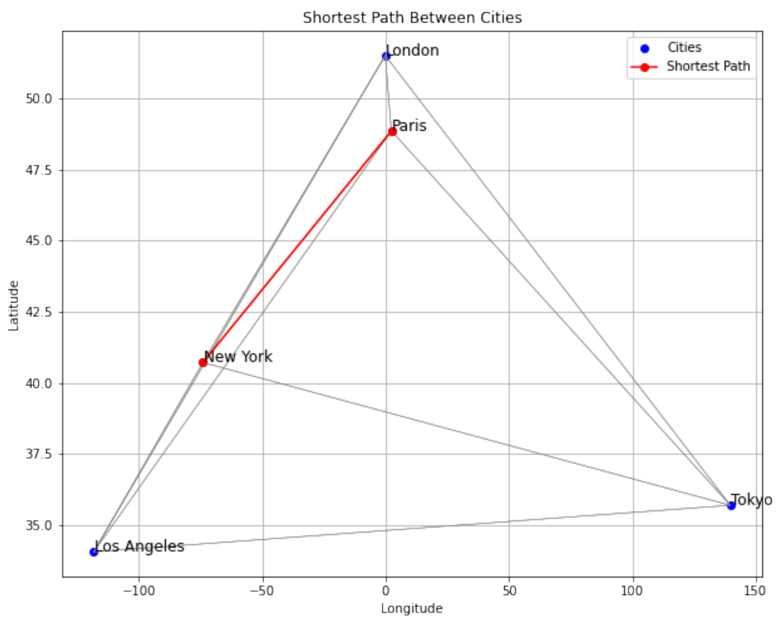

Figure 6: Shortest$_{p}ath$

Figure 7: Optimal spherical path between New York and Paris

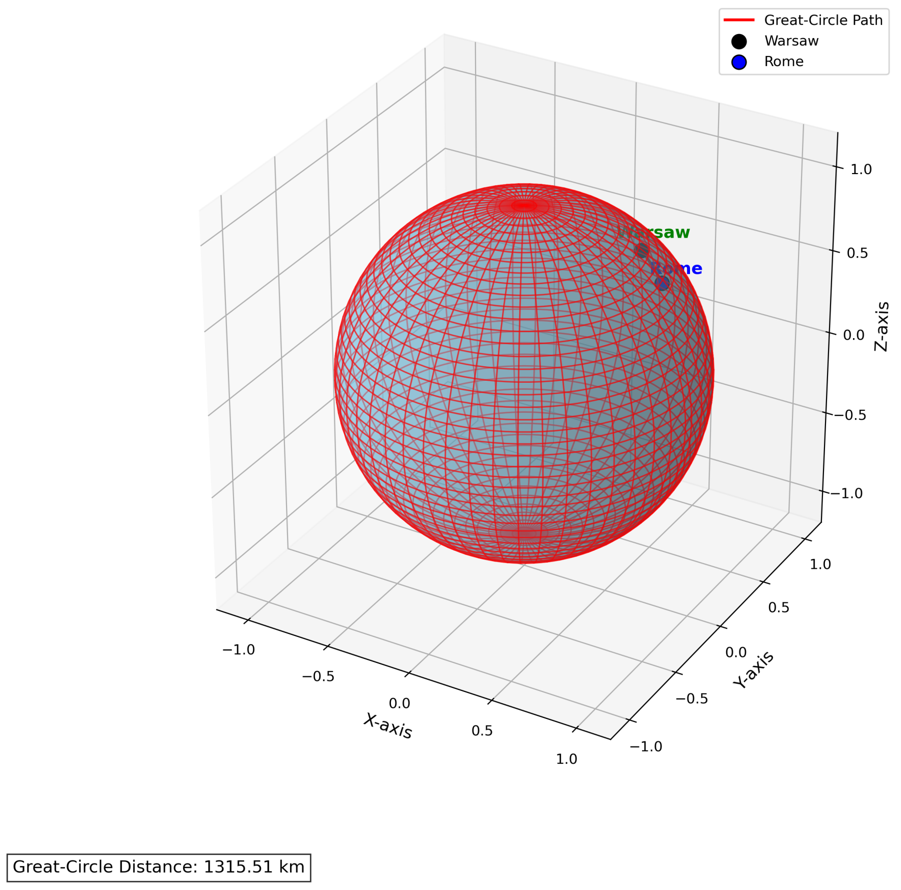

Figure 8: Great$_{c}ircle_{p}ath$

Figure 9: Comparative flight travel times between major global cities

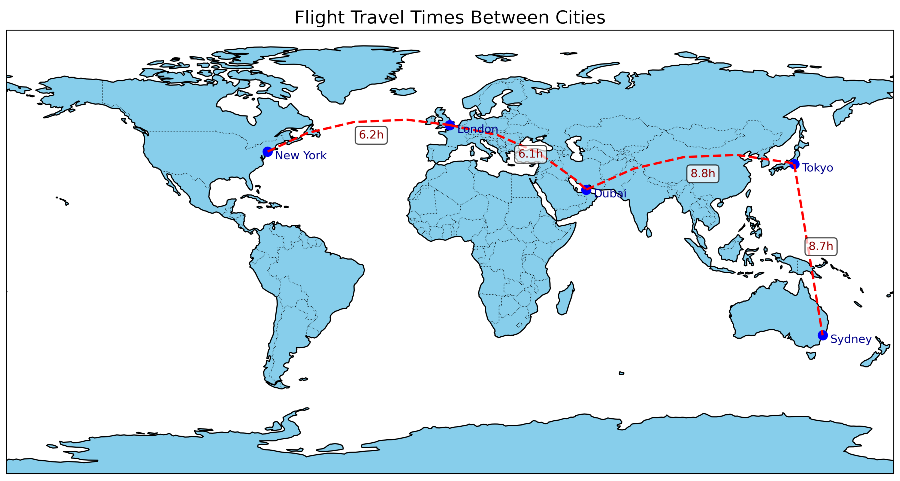

Figure 10: Flight travel time and path

Figure 11: Great-circle path visualization on 3D sphere between Warsaw and Rome

3 Example Code Usage

3.1 Distance Calculation

Listing 1: Calculating great-circle distance between two points

```python
from SphereStats.distance import calculate_great_circle_distance

# Coordinates in (latitude, longitude) format
point1 = (40.748817, -73.985428)  # New York
point2 = (34.052235, -118.243683)  # Los Angeles

# Calculate distance in kilometers
distance_km = calculate_great_circle_distance(point1, point2)
print(f"Distance: {distance_km:.2f} km")
```

3.2 Routing with Waypoints

Listing 2: Generating optimal spherical route

```python
from SphereStats.routing import generate_route

# Generate route with intermediate waypoints
route = generate_route(
    origin=(40.748817, -73.985428),  # New York
    destination=(34.052235, -118.243683),  # Los Angeles
    num_waypoints=5
)

print(f"Generated route with {len(route.waypoints)} waypoints")
```

3.3 3D Convex Hull Visualization

Listing 3: Computing and visualizing convex hull on sphere

```python
from SphereStats.convex_hull import convex_hull_on_sphere, plot_convex_hull_3d

# Global city coordinates
points = [
    [40.748817, -73.985428],  # New York
    [34.052235, -118.243683],  # Los Angeles
    [48.8566, 2.3522],         # Paris
    [-33.8688, 151.2093],      # Sydney
    [35.6895, 139.6917],       # Tokyo
]

# Compute and visualize convex hull
hull, cartesian_points = convex_hull_on_sphere(points)
plot_convex_hull_3d(hull, cartesian_points, title="Global Cities Convex Hull")
```

4 Applications

- **Navigation and Routing**: Air or marine route planning, waypoint optimization for fuel efficiency
- **Geospatial Intelligence**: Proximity alerts, protected area analysis, hazard zone identification

- **Urban Planning**: Zone reachability analysis, route coverage optimization, accessibility modeling
- **Spatial Analytics**: Global resource allocation, geographic clustering of large datasets
- **Scientific Research**: Earth science applications, atmospheric modeling, planetary studies

5 Installation

The library can be installed via pip:

```
pip install SphereStats
```

For development installations:

```
pip install -e git+https://github.com/username/SphereStats.git#egg=SphereStats
```

6 System Requirements

- Python $\geq$ 3.8 (64-bit recommended)
- Core dependencies:
  - numpy $\geq$ 1.20.0
  - scipy $\geq$ 1.7.0
  - matplotlib $\geq$ 3.4.0
  - cartopy $\geq$ 0.20.0
  - pyproj $\geq$ 3.1.0

7 Contributing

We welcome contributions from the community. The project follows these guidelines:

- Submit bug reports and feature requests through GitHub Issues
- Follow PEP 8 style guidelines for code contributions
- Include comprehensive unit tests for new features
- Document all new functionality

8 License

This project is licensed under the MIT License - see the LICENSE file for complete details. The MIT License permits free use, modification, and distribution of the software for any purpose.

9 Contact Information

For questions, support, or collaboration opportunities:

- **Primary Maintainer**: Ghulam Abbas Zafari
- **Email**: ghulamabbas.zafari@mail.polimi.it
- **GitHub Repository**: `https://github.com/zafariabbas68/SphereStats`