Python Visualization, Analysis, and Jupyter Notebook Development for Unstructured Data

Philip Chmielowiec[1,2], Orhan Eroglu[1], Alea Kootz[1], Anissa Zacharias[1], Michaela Sizemore[1]

[1] National Center for Atmospheric Research
[2] University of Illinois at Urbana-Champaign

Abstract

Background
The most widely-used approach for visualizing unstructured data is by rendering the triangular mesh produced by running Delaunay Triangulation on our set of points. This method is computationally expensive and produces an approximation of our mesh. Modern unstructured meshes often come with connectivity information in the form of face nodes, which Delaunay does not utilize.

Methods
By utilizing the face node connectivity information, we are able to construct a mesh of polygons that is suitable for rendering with libraries such as Datashader.

Results
This implementation produces results that showcase the true structure of our mesh. The overall performance compared to Delaunay Triangulation is around 3-4x faster. Rendering with Datashader allows for us to visualize millions of polygons in under a few seconds.

Conclusion
Visualizing our unstructured data as a mesh of polygons offers significant performance and visual improvements. The main bottleneck is converting our raw data to polygons in Python, which is an active area of development in the computational geometry community (Shapely 2.0)

Mesh Construction

1. Initial Polygon Array
- Node Coordinates are indexed using Face Node values
- Polygon: $[x_1, y_1, x_2, y_2, x_3, y_3, ..., x_n, y_n]$, where $n = n_{face\_nodes}$
- Total # of Polygons = Total # of Faces

2. Cyclic Polygon Search
- Locate any polygons that wrap around the globe
- Check longitude values (crossing between ± 180)
- Store indices

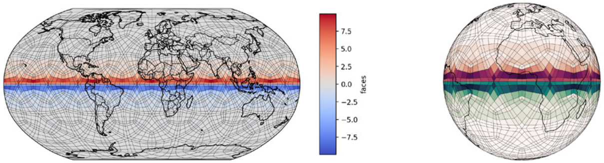
3. Convert to GeoDataFrame
- Initial Polygon Array converted to Polygon Objects through pyGEOS
- Loaded into a Spatial Pandas GeoDataFrame as “geometry”

4. Face Value Calculation
- Face Node Mean or Direct Face Value
- Loaded into our GeoDataFrame as “faces”

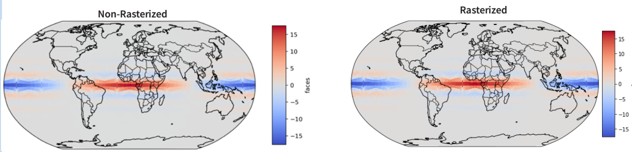
5. Removing Cyclic Polygons
- Mask Cyclic Polygons using stored indices

Visualizations

Unstructured Meshes



Rasterization

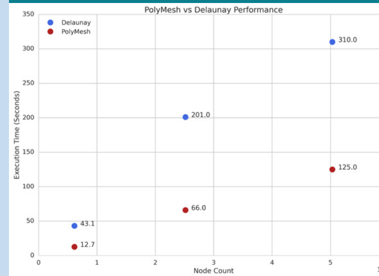


Cyclic Polygon Correction

Core Packages

Data	Mesh Representation	Visualization
uxarray	SpatialPandas	hvPlot
	GeoPandas	Datashader
	GEOS	GeoViews

Performance



Workflow

Data Loading

```
base_path = "/glade/p/cisl/vast/vapor/data/Source/UGRID/NOAA-geoflow/large/"

ugrid_large = ux.open_dataset(base_path + "grid.nc",
                              base_path + "v1.000000.nc",
                              base_path + "v2.000000.nc",
                              base_path + "v3.000000.nc")
```

Mesh Representation

```
projection = ccrs.Robinson()
geoflow_small = Polymesh(ugrid=ugrid_large, projection=projection)
geoflow_small.construct_mesh()
```

Visualization

```
df = mesh.data_mesh(name="Example Var", dims={"time" : 0}, fill='faces')
plot = df.hvplot.polygons(rasterize=True,aggregator='mean', c='faces', cmap=cmap)
plot * gf.coastline(projection=projection) * gf.borders(projection=projection)
```

Future

Performance Improvements
- pyGEOS + Shapely 2.0 merger
- Faster Polygon Calculations
- Improved render with Datashader

NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH