Optimized Visualization of Large Scale Ocean Circulation Simulations II.

Thomas Connolly (CMU), Brian Wen (JHU), and the Poseidon Group

Abstract

The purpose of this project was to apply image interpolation techniques to develop a dynamic tile-server visualization for a large oceanographic dataset. The method of rendering images directly from the dataset relies on building interpolator objects that track which points in the dataset are relevant to the interpolation of pixel values at each resolution. Interpolators are built by transforming data points into the visualization's Web Mercator projection and resampling at each resolution to calculate pixel interpolation weights. There are three stages to the optimization of this technique. First, data points not used in the interpolation of ocean pixel values, e.g. interior land data points, are masked out of the interpolator objects. Second, interpolators are broken into tiles at each resolution to allow single image tiles to be dynamically computed at each resolution. Third, the data point indices in each interpolator are reindexed into the coordinates of the original dataset to allow data points to be accessed directly without intervening coordinate transformations. Applying these interpolators to a chunked zarr data structure allows for very efficient dynamic image rendering, especially for low level tiles.

Introduction

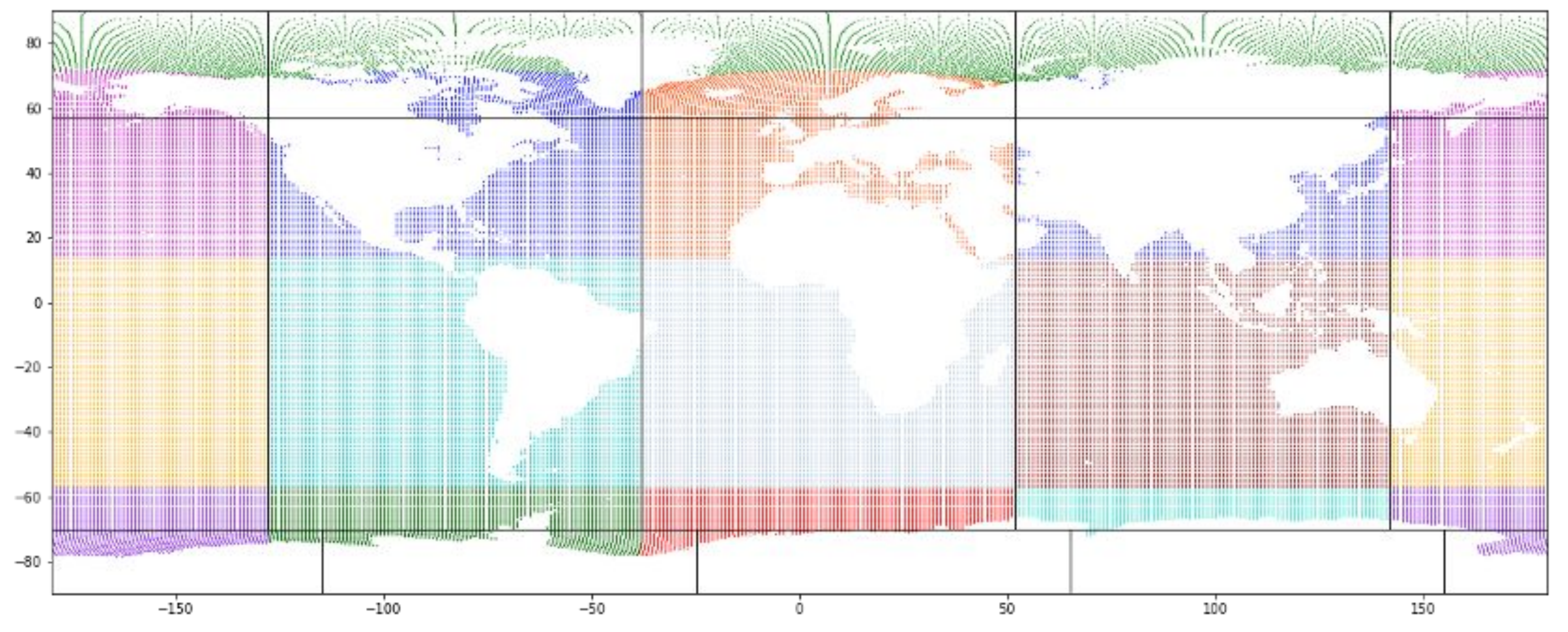
Fig 1: The 13 tile coordinate system of the Poseidon dataset

Dynamic visualization of data is accomplished by using 'interpolator' objects to store the information needed to compute an image of the data.

An interpolator object stores the set of all indices to the data points in the full dataset necessary to build an individual tile at a specific resolution. It also stores, for each image pixel in the tile, which data points are needed to calculate that pixel's value and what the relative weighting of each data point is.

Methods

STAGE I:

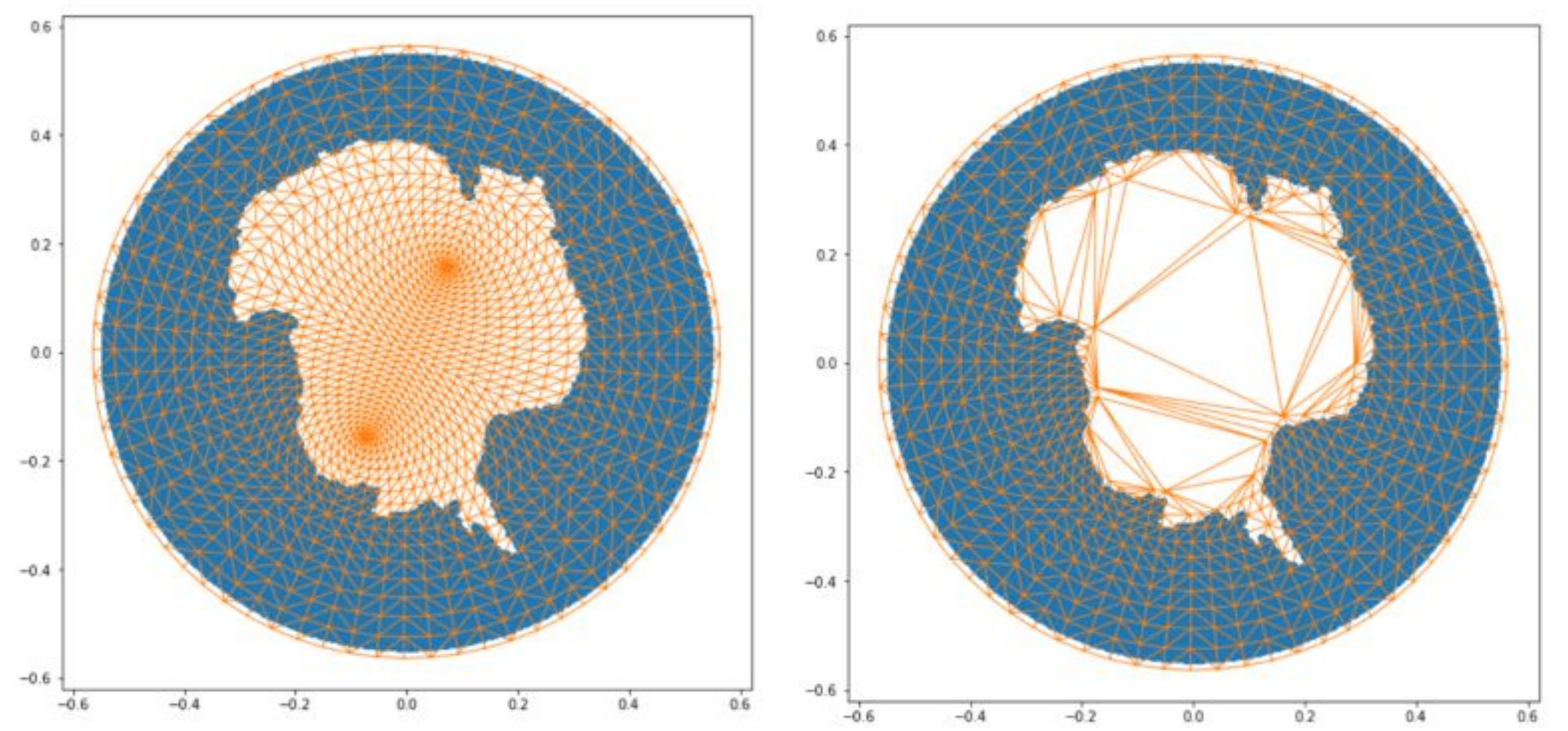
Fig 2: Untrimmed vs trimmed Delaunay triangulation

We begin with an unoptimized interpolator object that inefficiently indexes data from the whole earth to build images.

We apply a mask of land values to the image pixels to exclude pixels over land. Then, for each remaining non-land interpolator pixel we determine which Delaunay triangle it falls in. Data points not part of any of these triangles are masked out of the interpolator object.

This significantly reduces the amount of calculation necessary to build an image by avoiding calculating values for pixels on land.

STAGE II:

Z-Level	Dimensions	Total Tiles
0	1x1	1
1	2x2	4
2	4x4	16
3	8x8	64
4	16x16	256
5	32x32	1024
6	64x64	4096

Fig 3: Visualization tiling dimensions at each zoom level

Full earth interpolators are broken into 256x256 pixel image tiles for efficient rendering in the viewer's Web Mercator projection.

Each interpolator object now contains only the pixel values and weights needed to interpolate a single image tile.

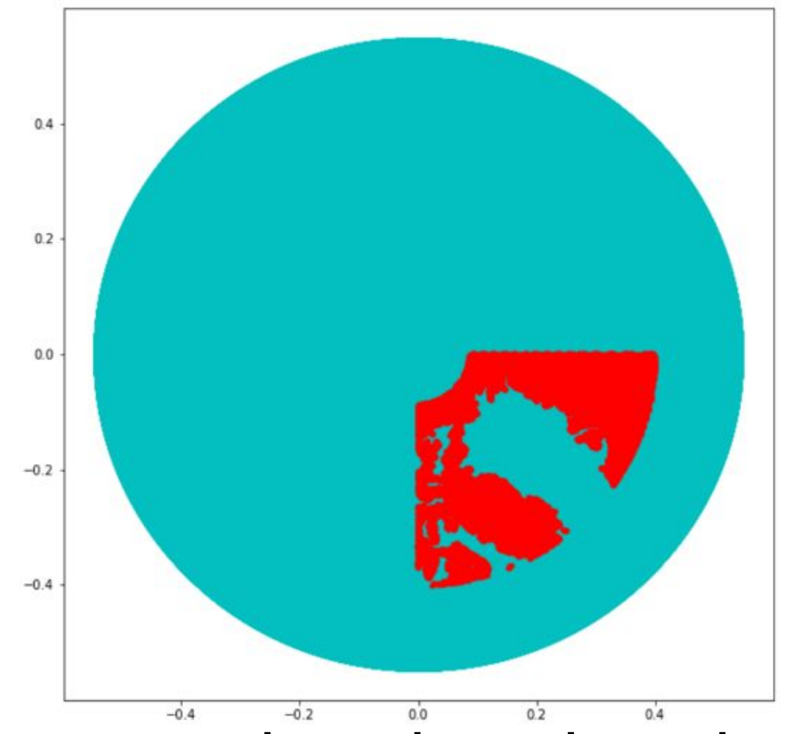
Fig 4: Interpolator tile pixels overlayed on lat-long coordinate system

STAGE III:

Although the interpolator objects now hold much less pixel data, building an image still involves handling the a full earth slice of the dataset since it must be first projected from the original coordinate system to visualization's coordinate system.

This can be avoided by re-indexing the interpolator indices array so the indices refer to data points in the original LLC coordinate system rather than the visualization coordinate system.

After re-indexing the indices we can reduce each interpolator's indices array to only the unique set of points relevant to the calculation of that tile's images. This dramatically reduces the amount of data in the dataset that each interpolator needs to handle, increasing the speed of dynamic image rendering.

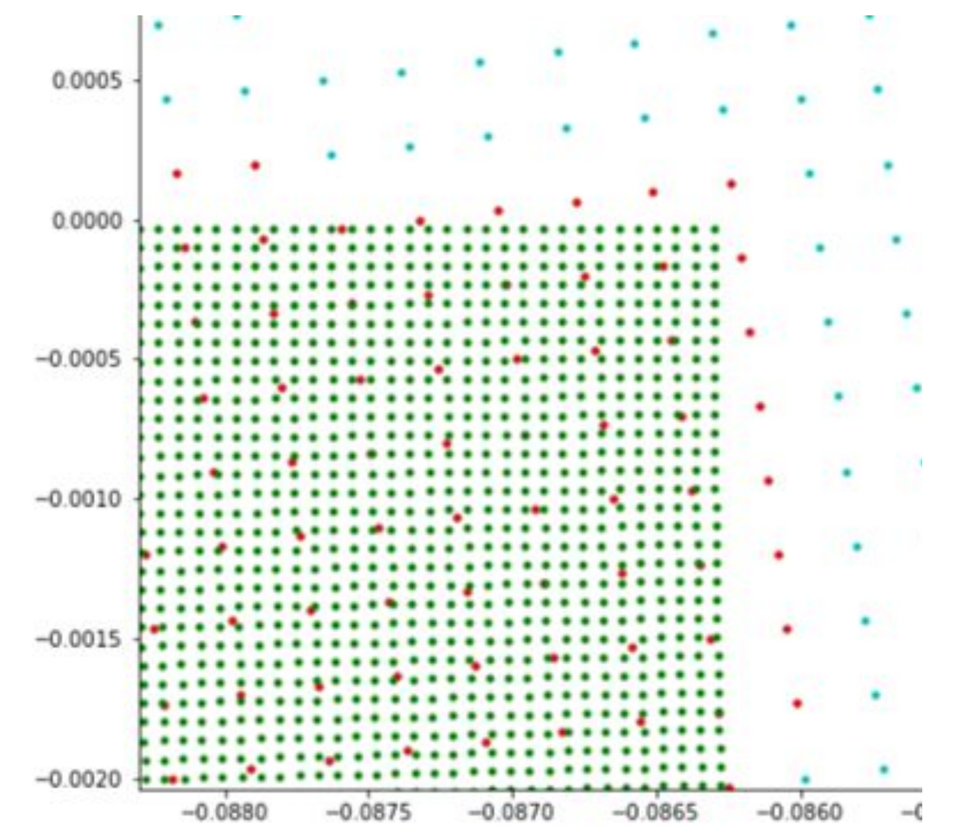
Fig 5: Interpolator pixel locations overlayed on dataset data points

Results

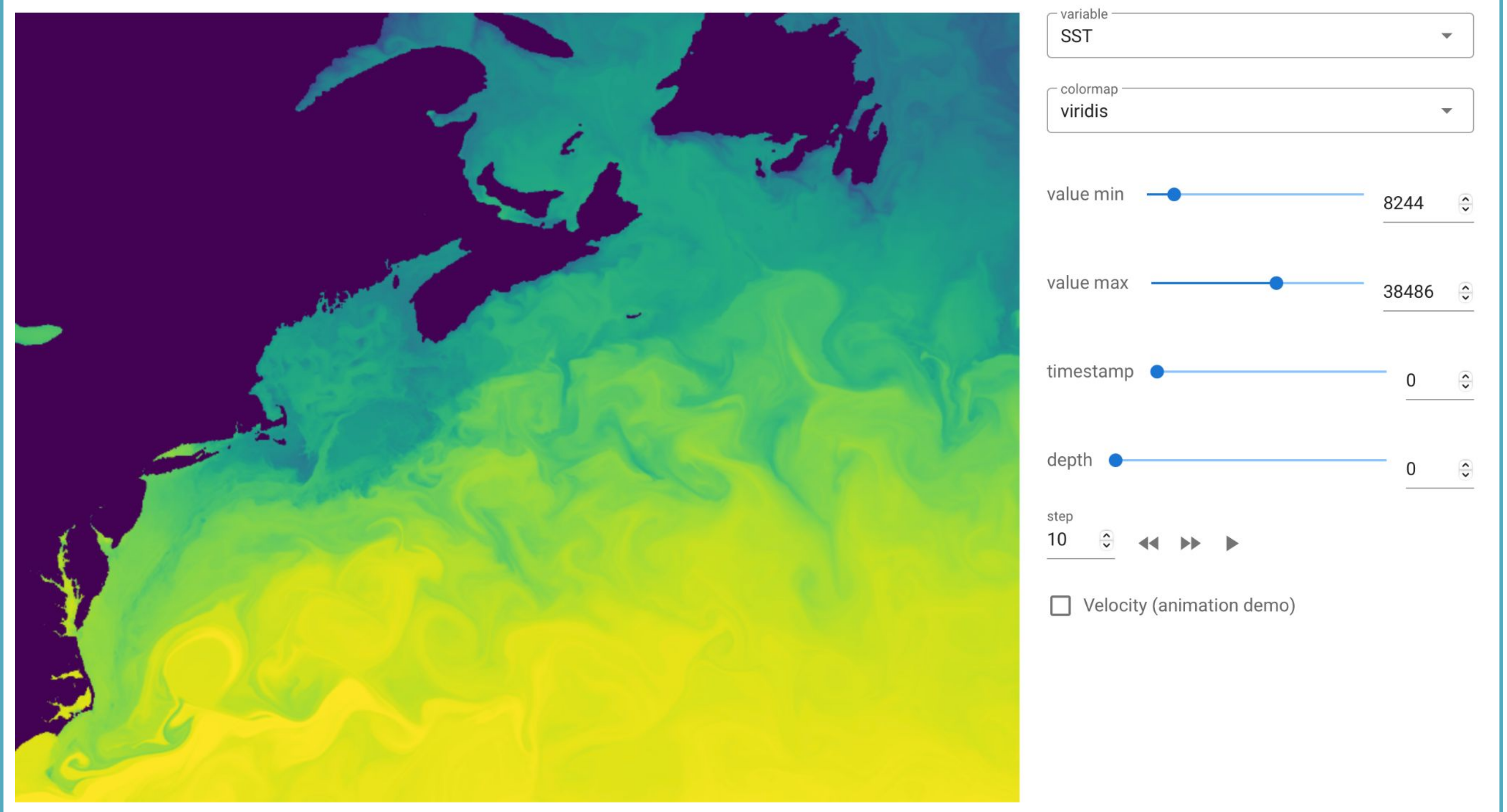
variable: SST
colormap: viridis
value min: 8244
value max: 38486
timestamp: 0
depth: 0
step: 10
☐ Velocity (animation demo)

Fig 6: Prototype dynamic data visualizer

The viewer allows a user to select an ocean parameter to view and then step through timesteps and depth levels at a desired zoom resolution.

Zoom Level	Unoptimized Full Image Render Time (s)	Optimized Tile Render Time (ms)
1	2.61	654 ± 29.9
2	2.75	349 ± 16.6
3	3.31	215 ± 12.3
4	5.62	133 ± 7.58
5		126 ± 3.08
6		120 ± 2.32

Fig 7: Timing results pre and post optimization

The optimized dynamic rendering method shows significant improvement over unoptimized rendering time, particularly for low level tiles. The slow rendering of higher level tiles can avoided by precomputing the top few levels and serving the images directly to the viewer.

Viewer URL: http://scitest09.pha.jhu.edu/poseidon-viewer/viewer/index.html

Acknowledgments

Thanks to Professor Alex Szalay and Dr. Dmitry Medvedev for guiding this project and making it possible.