ELEC4848 Senior Design Project 2021-2022

Lai Hong Kit (3035470773)

3D Tidal and Cloud Visualization System (Extension on HKO ISWP Project)

Supervisor: Dr. Choi, Yi King
Second Examiner: Dr. Kenneth KY Wong

Abstract

As WebGL and web-based rendering technology matured, new possibilities were enabled to visualize high dimensional meteorological data in real-time and in a portable web-based environment. Amused by the advancements, I want to take the opportunity to try and develop something using these cutting edge technologies. To test and prototype what these new technologies can possibly bring to the table in terms of visualization and extracting insights from these huge, complex datasets.

This project would act as an extension on top of the existing project that I have worked on and prototyped during my Integrated Study-Work Programme (ISWP) in Hong Kong Observatory (HKO) from 2020-2021. Using the already built backend sharded MongoDB cluster and frontend visualization framework, Deck.gl, a different visualization project has been proposed and adopted as my FYP.

This project aims to visualize and analyze the possible impacted areas of rain and tides with real-time or numerical model data provided by HKO on a web-based platform. A total of six datasets were provided, including detailed Digital Terrain Model (DTM) and Digital Surface Model (DSM), tidal model data generated by Simulating WAves Nearshore (SWAN) and Wave Watch III (WW3) and semi real-time reflectivity data from Tai Mo Shan (TMS) and Tate's Cairn (TCR) radars.

A prototype visualizing tiled 3D DTM terrain, tidal data and reflectivity data have been successfully implemented. Users can smoothly select and view his / her desirable datasets with the corresponding metadata and descriptions of the dataset provided in detail.

Acknowledgements

I would like to acknowledge and express my sincerest gratitude to my supervisor Prof. Loretta Choi and Mr. Wong Wai Kin from HKO who made this work possible. Their guidance and advice carried me through all the stages of my project.

I would also like to thank Mr. Au Yeung Kin Chung (HKO) for providing all the relevant data and technical advice along the way.

Finally, I would like to thank all the kind developers who help answered my questions along the way in forums like Stack Overflow, GitHub etc. Without these people, my project will not have been possible in the first place.

Table of Contents

Abbreviations

Introduction

With the advancement of computer technology, numerical simulation models and sensor technologies, more and more datasets and datapoints of our planet and atmosphere are being recorded and documented and predicted. From Vilhelm Bjerknes, who first introduced the idea and outlined the basic factors, equations, requirements and difficulties of a numerical weather prediction (NWP) system in a paper entitled “Das Problem der Wettervorhersage, betra-chtet von Standpunkt der Mechanik und Physik” in 1904, (Gramelsberger, 2009), to advanced modern NWP models which can formulate dozens of meteorological elements in a very fine spatial and temporal resolutions provided by centers like National Center for Environmental Prediction (NCEP) and European Center for Medium-Range Weather Forecasts (ECMWF) (Observatory, Numerical Weather Prediction Models, 2021). The huge amount of data collected contains the status of our atmosphere and oceans and to be able to understand the meaning of the data, processing and visualization systems to extract such insights.

Current generation visualization systems mostly use 2D cross-sections to display the actual multi-dimensional datasets, examples can be seen by visiting websites of NCEP, ECMWF and HKO. And pre-rendered methods or specialized software must be used if the whole picture of the desirable element were to be displayed. For example, the 3D render showing the intensity of Hurricane Katrina by NASA (Francis Reddy, 2015) or the 3D tidal visualization by Matthew and Stephen showcasing how coastal areas would be affected by floods caused by extreme weather events like Hurricane Michael in 2018 (Matthew Bilskie, 2019).

With the introduction and maturity of web-based 3D rendering technologies like WebGL, high performance interactive 2D and 3D graphics are available to any modern browsers that we use daily. This project aims to prototype and create a complete pipeline for processing and visualizing multi-dimensional tidal and radar data near Hong Kong in semi real-time as a showcase of the probabilities that this new technology might bring for meteorological data visualizations.

Methodology

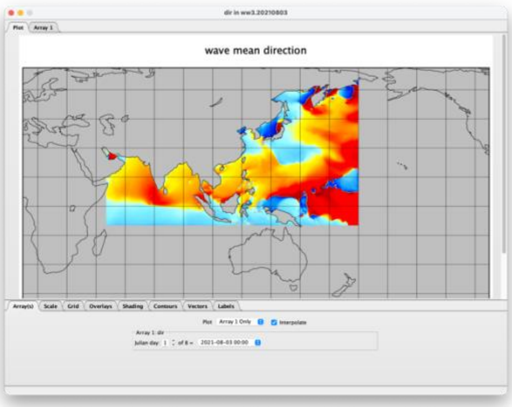
Datasets

Digital Terrain Model & Digital Surface Model. A set of DTM and DSM captured by the Lands Department of the HKSAR government in 2021 using Lidar scanning technology are provided. DTM only captures the bare Earth surface, by attempting to remove all natural and artificial features like trees and buildings. While DSM captures both natural and man-made features of the environment. The data comes in raw text files, sliced into regions according to their scanning sequences containing data for each point scanned. In each row, its coordinates in HK1980 Easting and HK1980 Northing, height in meters and other metadata were recorded. In its raw format, DTM comes in at around 500GB while DSM comes in at around 1TB.

Simulating WAves Nearshore Model Data. SWAN is the third generation wave model developed by Delft University of Technology that computes random, short-crested wind-generated waves in coastal regions and inland waters (SourceForge, 2021). The model were being ran by HKO once per 12 hours and data were generated and exported at 0000 and 1200 every day. They are formatted in CSV and the generated variables are as follows:

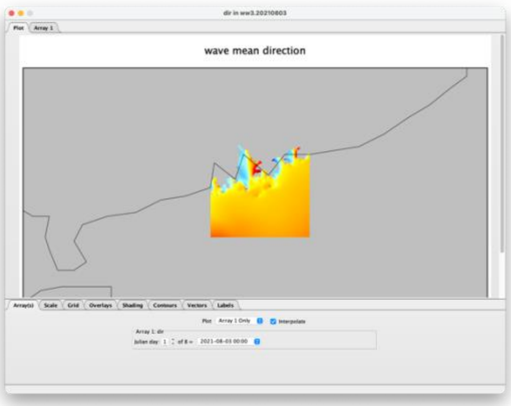
- Hsig: Significant wave height
- Hswell: Swell height
- Dir: Average wave direction
- X-Windv: X component of wind velocity at 10m sea level
- Y-Windv: Y component of wind velocity at 10m sea level
- PkDir: Direction of the peak spectrum
- Period: Wave period
- DrPT01 to DrPT06: Average wave direction of partition 01 to 06
- TpPT01 to TpPT06: Relative peak period of partition 01 to 06
- HsPT01 to HsPT06: Wave height of partition 01 to 06

Wave Watch III Model Data. WW3 is also a third-generation wave model developed by National Oceanic and Atmospheric Administration (NOAA) NCEP inspired by the WAM model, which solves the random phase spectral action density balance equation for wavenumber-direction spectra (National Weather Service (NWS), 2009). The model was being ran once per day on four different areas with different zoom level. With the largest covering the whole Asian Pacific Region (Figure 1) and the smallest dataset covering only the near waters of Hong Kong with an accuracy of 0.025 latitude and 0.025 longitude per pixel (Figure 2). The raw dataset are formatted in NetCDF, which is a common file format to store raw meteorological data. This model are also being ran twice a day at 0000 and 1200, with each run generating 8 timestamps with a prediction time of 1 day ahead and data of all the elements below.



Plot: Array 1 Only ☑ Interpolate
Array 1: dir
Julian day: 1 of 8 = 2021-08-03 00:00

Figure 1 (Panoply)



Plot: Array 1 Only ☑ Interpolate
Array 1: dir
Julian day: 1 of 8 = 2021-08-03 00:00

Figure 2 (Panoply)

A few generated variables from the WW3 dataset that are of particular use are as follows:

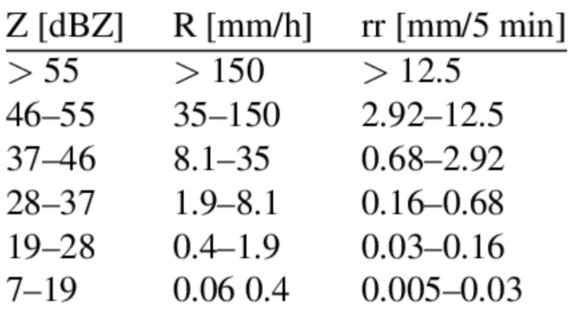
- Dir: Wave mean direction
- Dp: Peak direction
- Fp: Wave peak frequency
- Hs: Significant height of wind and swell waves
- Latitude: Latitude of the datapoint
- Longitude: Longitude of the datapoint
- Lm: Mean wave length
- Uwnd: Eastward wind
- Vwnd: Northward wind

Tai Mo Shan & Tate's Cairn Weather Radar Data. Both TMS and TCR radars are Doppler weather radars having a detection range of up to 500km, a scan period of 6 to 12 minutes and are actively looking for rainfall intensity in terms of reflectivity and Doppler wind field (Observatory, Long-range Weather Radars, 2021).

Due to the scope of the project, the finest and smallest range of 64 km data were provided. The raw datasets come in raw NetCDF files at 6 to 12 minutes intervals with each dataset containing the following elements.

- __xarray_dataarray_variable: raw reflectivity data in a 31x480x480 shape
- Height: actual height of the 31 vertical layers of the reflectivity data, in increments of 500, ranging from 0 meters to 15000 meters
- Time: timestamp of the current dataset
- X: x value in specially defined grid
- Y: y value in specially defined grid

Reflectivity is the amount of power returned to the radar receiver after each round of scans. It can be roughly translated to actual rainfall rates using the table shown in Figure 3 (Weusthoff, 2008).



Z [dBZ]	R [mm/h]	rr [mm/5 min]
> 55	> 150	> 12.5
46–55	35–150	2.92–12.5
37–46	8.1–35	0.68–2.92
28–37	1.9–8.1	0.16–0.68
19–28	0.4–1.9	0.03–0.16
7–19	0.06 0.4	0.005–0.03

Figure 3

Additional information regarding the coordinates of the datapoints in WGS84 were provided to simplify the data translation process.

Project Structure

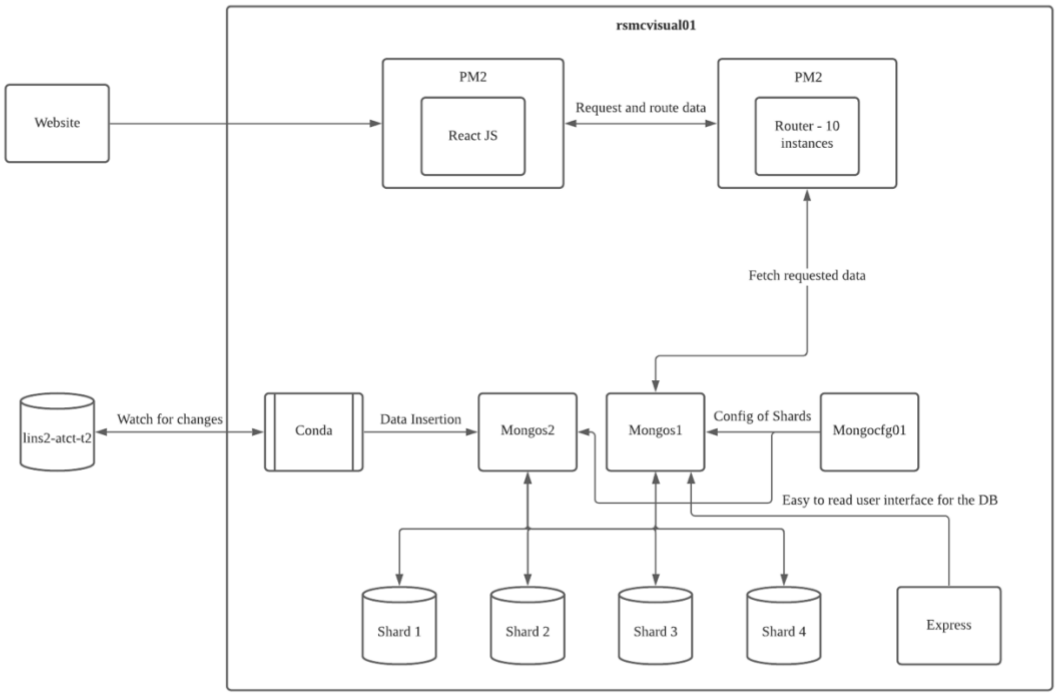


Figure 4

Overall structure. Figure 4 shows the current structure of the project in detail. Which mostly remained the same as the one inherited from the ISWP project in HKO.

Each individual rectangle on the right inside the rsmcvisual01 big rectangle denotes a Docker container containing one or more processes. Docker provides lightweight and loosely isolated environments for applications to be ran individually. It provides great flexibility and a lot of customization options, which is perfect for prototyping and development work. Also, Docker Compose scripts have been written for quick deployment of these containers in any given environment with the appropriate environment variables.

Backend. The backend structure is denoted by the lower half cluster of nodes in Figure 4. Lins2- atct-t2 is the server where the raw data is hosted. It can be any server as long it hosts the required data. It updates on various intervals with new files available for download at various intervals depending on the data source. Typically, around 6 to 12 minutes for semi real-time data and once or twice per day for model or NWP data. Conda is a miniconda3 (Python) process responsible for data fetching, data processing and data insertion to the database. The Conda instance watches for changes in the file server and processes the newly added files when they were being added into the data server. Both Mongos01 and Mongos02 are mongos instances where Mongos02 is responsible for routing write operations fired from Conda and write the data into the database while Mongos1 is responsible for routing the query operations, getting the required data and returning it to the router instances. Mongocfg is the configuration server that stores the configurations of the shards, it dictates how the data that are going to be stored will be partitioned and directs the mongos instances to the correct shard to find the correct partitions when data is being queried. A distributed and sharded NoSQL database were chosen as the database of choice as NoSQL allows for flexible schema designs which is perfect for the complex and unpredictable structures of all the meteorological datasets that might be required to be stored and visualized. A sharded approach also allows for faster data storage and retrieval as partitions of the dataset and be queried at the same time and then combined before serving to the router. Moreover, duplication can also be easily implemented to increase data security.

Finally, Express is a service that provides a human readable interface for easier monitoring the database performance and storage statistics, which was very helpful for debugging and the development process in general.

Frontend. The frontend structure, which is denoted by the two bigger nodes at the upper half in Figure 4. The website is being developed by ReactJS which is a JavaScript library for building user interfaces, with declarative design and a component-based approach, with lots of libraries to extend its capabilities (Meta Platforms, 2022). In addition to ReactJS, multiple other frameworks were used to visualize the metrological datasets, namely Deck.gl, Loaders.gl and React-map-gl, all lives under the ReactJS framework and were used extensively for rendering the necessary data.

Deck.gl is a WebGL-powered framework for a visual exploratory data analysis of large datasets. Unlike similar visualization frameworks such as Kepler.gl that provides a drag and drop simple experience to the user, Deck.gl uses a layered approach to data visualization, allowing complex visualizations to be constructed by composing existing layers and reusing them (Vis.gl, DECK.GL, 2021). The modularity nature brings great flexibility when it comes to building the visualizations. Each layer also comes with dozens of customization options which can be customized extensively. This framework provides the perfect balance between abstraction and customization and is perfect for my project. Loaders.gl is loader for big data visualization, includes parsers and encoders for many major 3D, geospatial and tabular formats (Vis.gl, Loaders.gl, 2021). React-map-gl is a React wrapper for Mapbox GL JS, which enables Mapbox maps to be shown on a browser using React (Vis.gl, react-map-gl, 2021).

Router. Express.js were used as routers of data between the frontend and backend. It is a minimal and flexible Node.js web application, with a myriad of HTTP utility methods and middleware available, enabling users to create a robust API quickly and easily (StrongLoop, 2017). PM2 were also used as a process manager to help watch for changes and rebuild the router, and help spawn clusters of the same router process for greater availability.

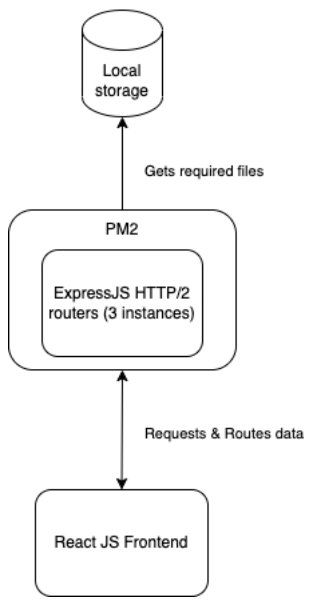


Figure 5

Local Environment. However, due to the pandemic and not being able to back to HKO and access the existing technology stack there. A more lightweight solution has to be found in order to develop the project locally. A simplified structure show in Figure 4 were adopted instead. With the local storage replacing the whole database itself and reducing instances of the ExpressJS router to only 3 instead of 10.

The HTTP protocol used by the router, however, is upgraded. HTTP/2 were used instead of HTTP/1.1. HTTP/2 is the newest HTTP standards to be published in May 2015 with the code RFC 7540 along with RFC 7541 (HPACK) (Ilya Grigorik, 2019). HTTP/2 has a lot of advantages over HTTP/1.1, it heavily optimizes the transfer of the HTTP/1.1 protocol, with the introduction of compression, multiplexing and prioritization. With benchmarks done by Romual, Emile and Nathalie in a paper in 2016, HTTP/2 always performs better than HTTP/1 standards with faster page download times with the addition of additional safety features as well (Romuald Corbel, 2016). With the huge volume of data that are required to visualize at the frontend, a speed up in routing speed between the client and server drastically improved the user experience. And in my project's experience, the data loading performance of my webpage improved by around 15% as well.

Data Processing

DTM & DSM. The goal of processing this dataset is a tiled height maps encoded in PNG to serve as the input of the Deck.gl terrain layer that are used to render the terrain. First of all, Python scripts were written to translate the coordinate system of each datapoint in the raw text data from HK1980 coordinate system to a more common WGS84 coordinate system which is the coordinate system used by the Deck.gl rendering framework. The Transformer method from the PyProj were used for the coordinate system transformation (Jeffrey Whitaker, 2021). Then, custom VRT files were constructed for each text file, which provides descriptions for the dataset and can be used for translating and GDAL datasets (Frank Warmerdam E. R., 2022). Custom GeoTIFFs were constructed storing metadata and describing the geographic image data (Open Geospatial Consortium, 2022). In the meantime, the Grid operation from GDAL were performed to create a gridded data from the possibly scattered source data (Frank Warmerdam E. R., gdal_grid, 2022), as images are pixels which must be stored in a regular grid. Then, all GeoTIFF files were combined into one huge GeoTIFF containing all data using the gdal_merge.py script included in the GDAL suite of tools (Frank Warmerdam E. R., gdal_merge.py, 2022). The purpose of combining all the existing GeoTIFFs is for the convenience of slicing them into tiles according to the OpenGIS Web Map Tile Service Implementation Standard. Figure 6 shows how the tiles will be sliced (Open Geospatial Consortium Inc., 2010). The whole world map will be divided into various zoom levels and with each zoom level increase, one tile will be sliced into four tiles while the resolution remains the same, thus rendering more detailed data as the user zooms into the map. The gdal2tiles.py Python script were used to slice the huge GeoTIFF file into appropriate zoom level and details. The finest terrain will be seen at zoom level 15 with an accuracy of 2 meters per pixel of the input data and a tile resolution of 512 pixels by 512 pixels. The tile resolution and zoom level were chosen as a compromise and a balance point to details visible by the user and the performance of the website in general. As for how the height data were encoded into the sliced PNG tiles, the Terrarium Standard were used. To encode the height in meters to the red, green, blue and offset channels. The following pseudo code can be used (Nvkelso, 2017):

```
v = data + 32768
r = np.floor(v/256)
g = np.floor(v % 256)
b = np.floor((v - np.floor(v)) * 256)
```

The processing time for both DTM and DSM are quite long, requiring dozens of minutes to process the raw data to the end result tiled PNG height maps. However, as this is a one-time process, and only the tiled PNGs will need to be accessed by frontend later, processing times for this dataset is not a concern.

The detailed codebase can be accessed at https://github.com/laijackylai/hkdsm.

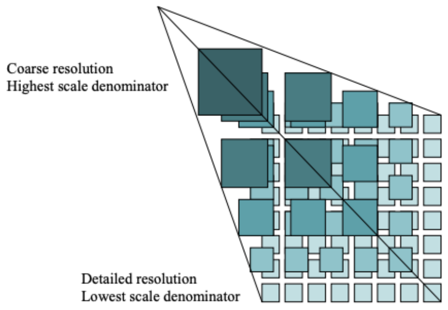


Figure 6

The result of the sliced tiles can be seen in Figure 7 and Figure 8 showing a sample of a PNG tile at zoom level 10 and zoom level 15. The size of the resulting PNGs ranges from 8KB up to 512KB.

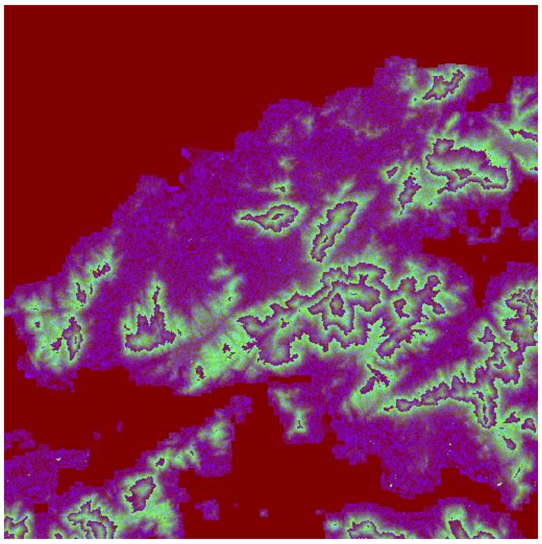


Figure 7: Zoom Level 10 (512x512

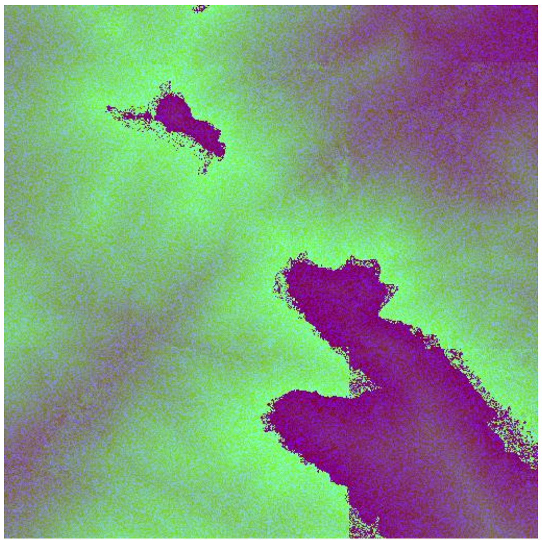


Figure 8: Zoom Level 15 (512x512)

Wave Watch III. The goal of processing this dataset is identical to the similar to DTM and DSM, the significant wave height data of the dataset were to be encoded into a height map for 3D mesh construction and overlayed on top of the terrain to show the areas affected by tides or ocean waves. The process involves reading the appropriate elements from the raw NetCDF file using the Python PyNio library. The PyNio library allows for read and write access to the NetCDF formats which is based on the the original NCL language required to read and write such datasets (NCAR, 2022). With the right element accessed, black and white PNG files encoded only to the 8-bit black and white channels were created using the Python Pillow Image library (Fredrik Lundh, 2022). All, timestamps of each incoming datasets will be processed in real-time and uploaded to the database. The processing time for this dataset is very short, only around a few seconds as the resolution of the source data is quite course, only having a resolution of 81x81 covering the area of Hong Kong. Moreover, the datasets only come in twice a day, slightly after 0000 and 1200 where the Wave Watch model were being ran. Therefore, data processing time is not a concern. Figure 9 below shows a sample generated height map, a light color represents a higher value which the wave significant height is taller, while darker color represents a smaller wave height value or 0 at land, slight trace of Hong Kong's southern coastline can be seen in the middle top part of the figure. The resulting height maps are all 4KB in size.

Figure 9 HSig (81x81)

Simulating WAves Nearshore. The goal of processing this dataset is the same as WW3 above, generating a height map showing the significant wave height by contructing a mesh to overlay with the terrain layer. However, even with the abundant variables available mentioned in the Dataset section above, the domain range for this dataset cannot be found. Thus, even if this dataset is much more fine with more datapoints available, it cannot be processed and visualized.

Detailed codebase and other generated height maps can be seen at https://github.com/laijackylai/hktides.

Tai Mo Shan & Tate's Cairn Radar. The goal of processing these two datasets is to construct custom PLY files for point cloud visualizations on the frontend. PLY, also known as the Polygon File Format or the Stanford Triangle Format, stores graphical objects that are described as collections of polygons or in point cloud's state, a collection of points with its corresponding attributes. The structure of a typical PLY files are as follows:

```
Header
Vertex List
Face List
(lists of other elements)
```

The header stores metadata that describes each element type, specifying its type and number of elements in the object. Also it indicates whether the file is being stored in a smaller binary form or the human-readable ASCII form (Bourke, 2021).

Normally, PLY files allow for any type and name of the variables to be used. However, when going through the source code of the PLYLoader.js in GitHub used by Loaders.gl, which is forked from three.js (mrdoob, 2022). It is discovered that only specified attributes will be read and parsed. Those elements include 'x', 'y', 'z' positions, 'nx', 'ny', 'nz' normal, 'u', 't' textures and 'red', 'green', 'blue' values. The element name must also either be 'vertex' or 'face'. If the processed data will be stored in ASCII, which is a human-readable text based storage format, changing the source code to match my own specifications would be manageable. However, since the final processed product would be a binary PLY file, due to storage, bandwidth and frontend performance considerations. Also, Updating the binary read part of PLYLoaders.js is quite difficult and given the project timeframe, its decided that it would not be modified. Instead, custom variables that I need, namely, the original reflectivity value dBZ of the datapoint will be mapped to the 'u' and 't' attribute. The PyNio library are used to read and extract the appropriate variables in the individual datasets, then using the Numpy module to construct a structured array with all the data for each data point, with each entry containing longitude, latitude, height, red, green, blue, dBZ, dBZ. Not all data pointers were stored, negative values where no rain were present were filtered out of the dataset as they do not indicate any rain and to save storage spaces and bandwidth. Last but not least, the plyfile module from Python were used to write to both the binary and ASCII versions of the particular dataset (dranjan, 2021). The result PLY dataset contains the following headers:

```
ply
format ascii 1.0
element vertex 88763
comment contains radar CAPPI data
comment format in ((lon, lat, height), (r, g, b), (dBZ, dBZ))
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
property float s
property float t
end_header
```

The radar data is color coded according to the NOAA color chart shown in Figure.

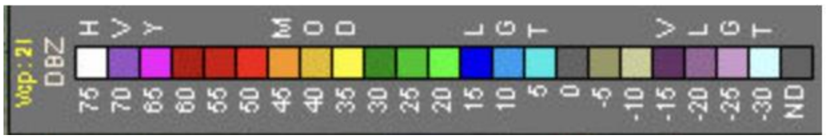


Figure 10

The resulted binary files will be used to serve to the frontend while ASCII versions were used for data verification. Sizes of binary PLY files ranges from 1.9MB to 36MB whilst source NetCDFs are all 57.2MB. As the original dataset comes into the storage server at 6 to 12 minutes intervals. Processing of the two NetCDF files must be done within this interval. Currently, each file takes around 1minute to process using 1 CPU core. Parallel processing are be utilized, such that both datasets are processed and uploaded and stored to the database at the same time.

The full codebase can be accessed at https://github.com/laijackylai/hkradar.

Routing

In the scope of this project, a total of three processed datasets and other metadata will have to be served from either the local storage or database to the frontend. They are the generated tiled PNG height maps, tides height maps, and radar PLY files. Moreover, metadata information such as what datasets are available will have to be served to the frontend to allow uses to select and view their desired dataset. Other required data can be directly queried from the frontend. Full codebase can be referenced at https://github.com/laijackylai/hkdsm.

Terrain Data. As the frontend sends query requests for tiled height maps of its corresponding zoom level, x label and y label. The router uses the File System module to look through the local storage or database to check whether the desired detailed height map is available. If yes, that height map would be sent. If not available, Mapbox RGB Terrain heightmap data would be sent to the frontend instead. This tileset however, its height values are encoded in a different standard, the Mapbox standard. The following equation can be used to decode the pixel values to the actual height values in meters (Mapbox, 2022):

```
elevation = -10000 + (({R} * 256 * 256 + {G} * 256 + {B}) * 0.1)
```

Radar Data. With the specified PlY file queried, the router looks for whether the file exists in the file system and sends the requested file back to the frontend. Theoretically, this operation should always be successful as the user can only select from available files returned by the router to the frontend when selecting the desired data.

Tidal Data. Same as radar data, search for the specified tidal height map and returns the PNG to the frontend.

Available Timeslots. With the specified element or type and date in the route parameters, the router will look through the file system or database to list the currently available dataset at the specified date and return a sorted list of names of files available to the frontend.

Results

Backend

All the datasets regarding the visualization of tides and radar reflectivity except SWAN due to technically difficulties were successfully processed and stored to local storage. The sizes of the resulted deliverables are also much smaller than the raw dataset itself, with optimizations and reducing redundant data in general. The following table summarizes the statistics of the processed datasets.

Name of Dataset	Raw Input Data Size (per unit of data)	Processed Size (per unit of data)	Processing time	Used
DTM	500GB (all)	141.6MB	~15 minutes	Yes
DSM	1TB (all)	150MB	~15 minutes	No
SWAN	5GB / timestamp	/	/	No
WWIII	4MB / run	4KB	< 1 minute	Yes
TMS	57.2MB / scan	8KB – 500KB	~ 1 minute	Yes
TCR	57.2MB / scan	8KB – 500KB	~ 1 minute	Yes

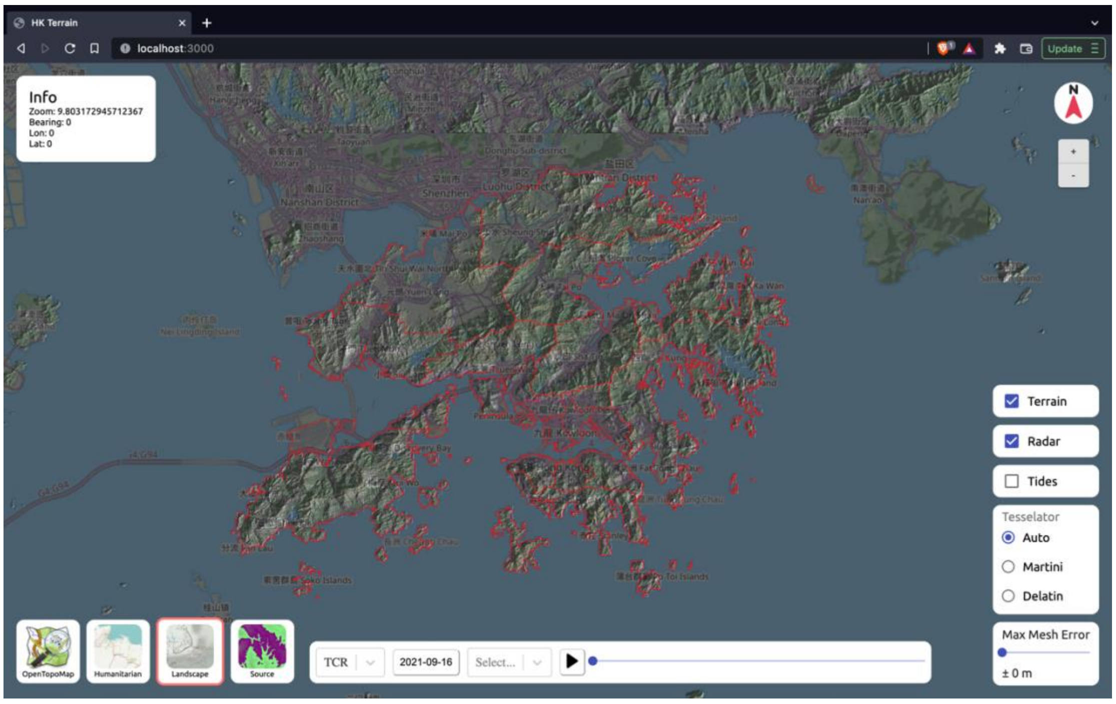
Frontend

Design & Controls. Figure 11 below shows what the webpage looks like after initial loading. I opted for a cleaner look referencing the rounded and white styles of Google Maps.

On the top left hand corner, an Info Box is present to show the user some metadata including the current zoom level of the map, the bearing of the map and the curser's latitude and longitude positions. And on the top right hand corner, a compass pointing to the North, and plus and minus controls for click zoom of the map.

Controls were positioned on the bottom of the webpage. On the left hand side, textures can be selected to overlay to the tiled 3D terrain. The default texture is the landscape texture hosted at tile.tunderforest.com. The long bar in the middle allows users to specify the datasets that they would like to view, with the first box selecting the sub-types of data if any, and the second and third box selecting the date and timestamp of the dataset respectively. And the slider allows user to scrub through the available datasets in the selected date freely, while the play button allows play back of animations of the selected dataset through time. On the right hand side, shows controls for individual elements, and toggles for the visibility of the three features, terrain, radar and tides. Tesselactor is an option to select the terrain generation algorithm for the 3D terrain, which will be explained in later sections, and the Max Mesh Error is also an attribute that constitutes to 3D terrain generation.

The map can be navigated through drag controls of the map, zoomed in and out with scroll wheels, and control + drag to change the viewing angle, i.e. pitch of the map.



Info
Zoom: 9.803172945712367
Bearing: 0
Lon: 0
Lat: 0

[✓ Terrain]
[✓ Radar]
Tides

Tesselator
[✓ Auto]
Martini
Delatin

Max Mesh Error
± 0 m

2021-09-16

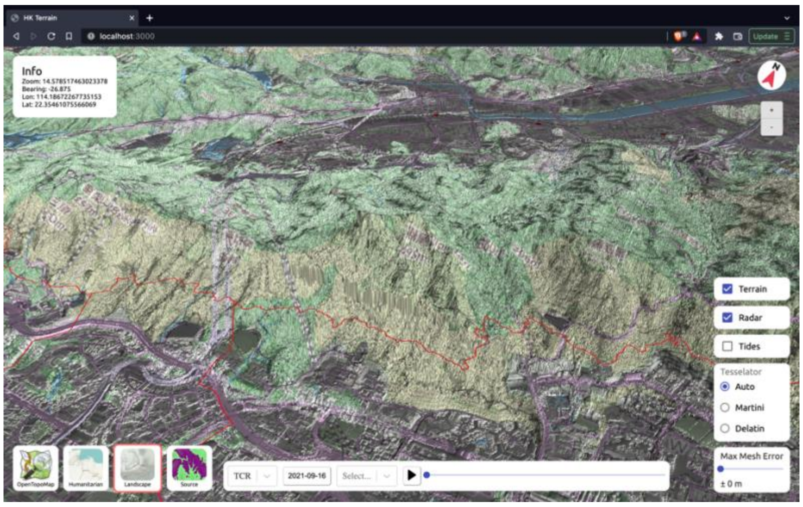
Figure 11: Landing Page

3D Terrain. Tiled 3D terrain were generated on the fly with the height maps routed by the backend system on the fly using the terrain layer by Deck.gl. The terrain layer takes an image as in input, in this case detailed DTM of Hong Kong or height maps served by Mapbox, and using the rgb values for each pixel and the decoder to decode the height value of each pixels. Then generates a 3D mesh using either Martini or Delatin. And renders the terrain on screen.

Martini stands for Mapbox's Awesome Right-Triangled Irregular Networks, Improved. It is a Javascript library for real-time terrain mesh generation (Barron, 2021). The library is based on the paper "Right-Triangulated Irregular Networks" by Will Evans et. al. (1997). It uses right triangles to deliver a very fast mesh surface approximation (William Evans, 1997). It requires an input of a (2^k + 1) x (2^k + 1) image and generates a terrain mesh of varying detail levels. The result is quite stunning as all the details of the terrain can be seen with a manageable FPS visualized in real-time on the web.

Delatin is another Javascript 3D terrain mesh generation tool. It uses Delaunay triangulation to approximate the terrain mesh and it does not have any input limitations like in Martini (mourner, 2021). Delatin is a port of the hmm (C++) library by Michael Fogleman which in turn based on the paper titled "Fast Polygonal Approximation of Terrains and Height Fields" in 1995 by Michael Garland and Paul Heckbert. Though Delatin does not share the limitations of Martini, the performance is yet to be desired for rendering large scale detailed 3D terrain in this project's use case. If Delatin were used, the viewport froze 9 times out of 10 thus rendering the webpage unusable. Therefore, during the data-processing stage, input specifications of Martini will have to be met in order to use Martini to render the terrain on frontend to achieve reasonable performance.

Figure 12 below shows the 3D terrain of the Lion Rock in the middle with landscape textures. The Shatin river can also be seen at the background. District section lines and coastlines can also be seen in red.


Info
Zoom: 14.578517463023378
Bearing: -26.875
Lon: 114.18672267735153
Lat: 22.35461075566069

[✓ Terrain]
[✓ Radar]
Tides

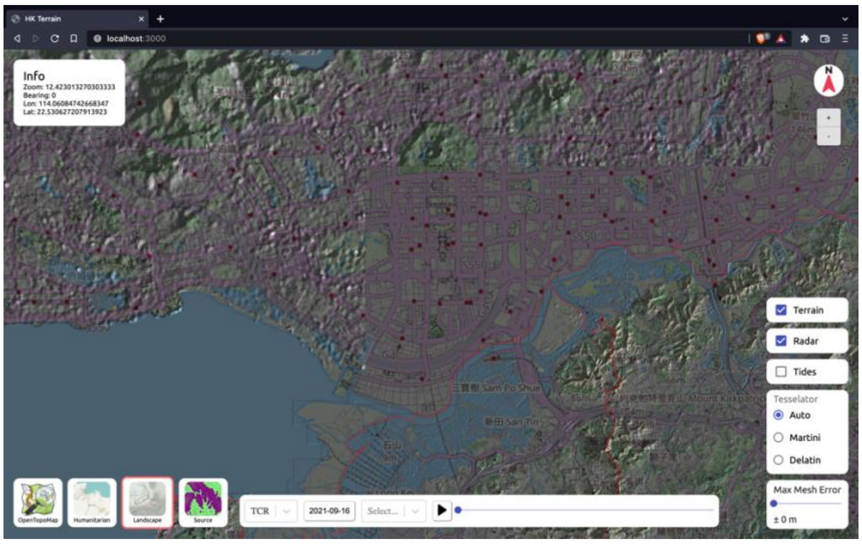
Tesselator
[✓ Auto]
Martini
Delatin

Max Mesh Error
± 0 m

TCR
2021-09-16
Select...

Figure 12: Lion Rock

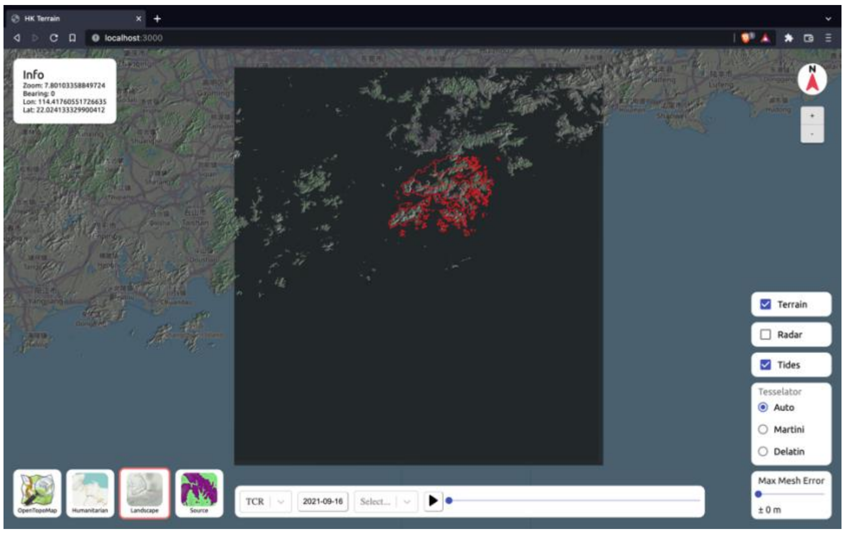
The detailed DTM provided by HKO though, only covers areas in Hong Kong. So upon close inspection of the northern areas of Hong Kong, shown in Figure 13, a big gap can be seen as there is no data available for areas in Shenzhen but it is included in the areas of the tile shown. Also, a great comparison for how detailed the DSM data provided by HKO is compared to open source data.


Info
Zoom: 12.423013270303333
Bearing: 0
Lon: 114.06084742668347
Lat: 22.530627207913923

[✓ Terrain] [✓ Radar] Tides
Tesselator: [✓ Auto] Martini Delatin
Max Mesh Error ± 0 m
TCR 2021-09-16 Select...

Figure 13

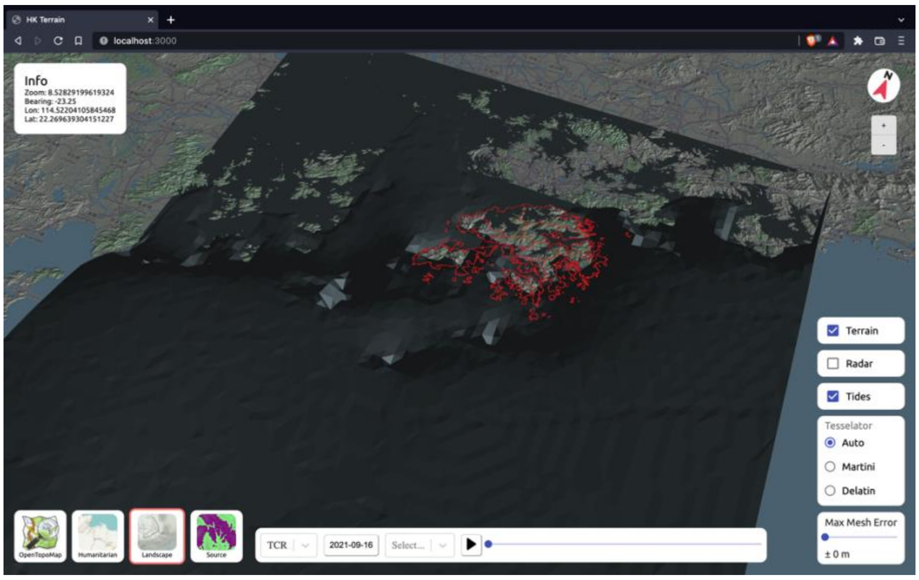
Tides. Tidal data were also rendered as mesh and overlayed on top of the map. Figure 14 showed the zoomed out version of the dataset. A deep dark blue texture have been applied to the tidal texture to contrast the green looks of land also to separate with the normal light blue texture of the sea. However, as the input data is quite course, only with a resolution of 81x81 over such a large area and a very low variance in tidal height, not much can be seen.


Info
Zoom: 7.80103358849724
Bearing: 0
Lon: 114.41760551726635
Lat: 22.024133329900412

[✓ Terrain] Radar [✓ Tides]
Tesselator: [✓ Auto] Martini Delatin
Max Mesh Error ± 0 m
TCR 2021-09-16 Select...

Figure 14 Tides

To exaggerate the effect and really show the significant height of the tidal data, it can be exaggerated by 100 times and the result can be seen in Figure 15 below. With the exaggeration, it can be concluded that the significant wave height decreases significantly when it is coming to closer the shore.

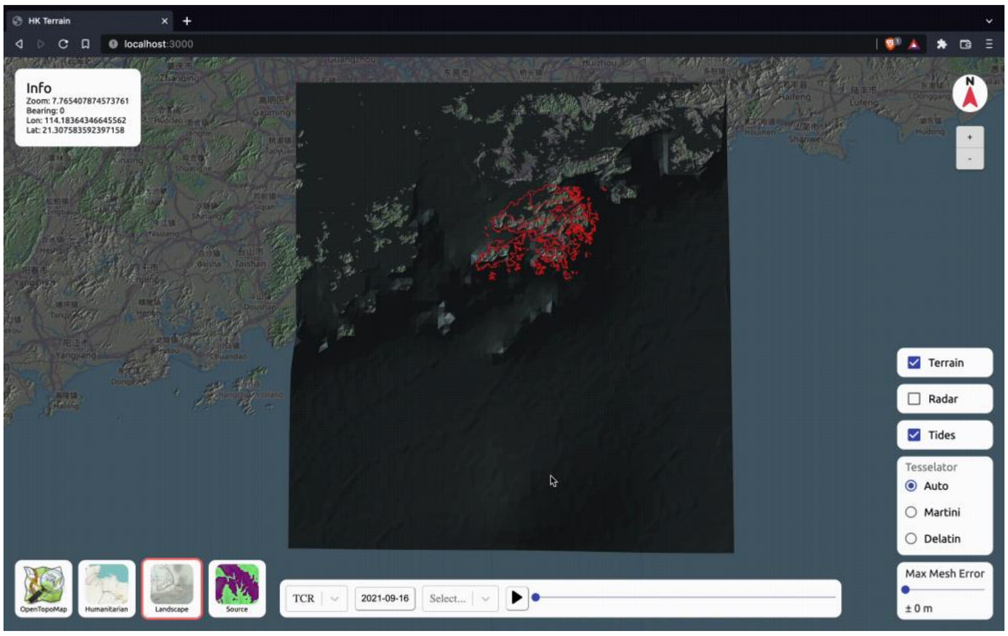


Info
Zoom: 8.52829199619324
Bearing: -23.25
Lon: 114.52204105845468
Lat: 22.269639304151227

Terrain [✓]
Radar [ ]
Tides [✓]
Tesselator: [✓ Auto] Martini Delatin

Figure 15

Figure 16 below shows an animated GIF showcasing the predicted significant wave height of near Hong Kong from 3rd August, 2021 0000 to 2100.

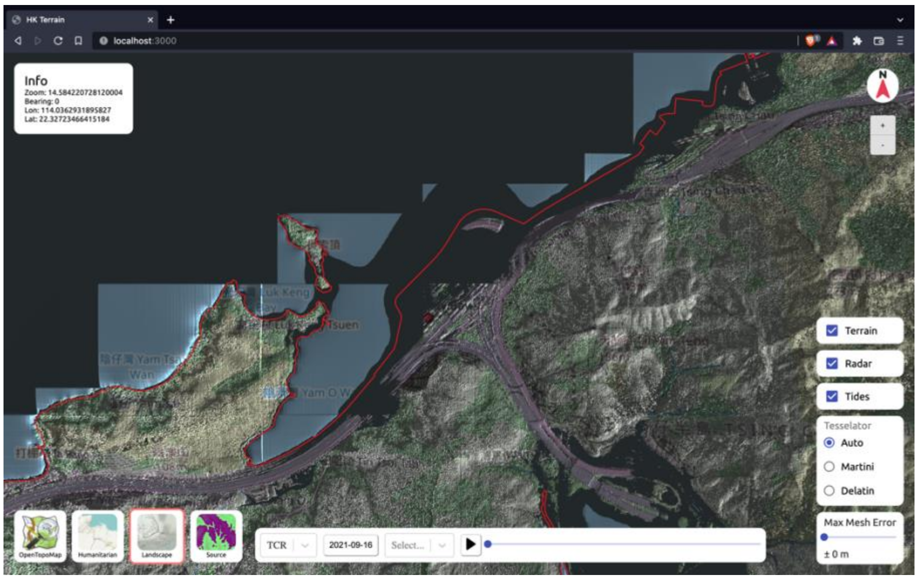


Info
Zoom: 7.765407874573761
Bearing: 0
Lon: 114.18364346645562
Lat: 21.307583592397158

Terrain [✓]
Radar [ ]
Tides [✓]
Tesselator: [✓ Auto] Martini Delatin

Figure 16

Upon close inspection, zooming to the shores of Hong Kong to take a closer look, however. A sizable section where the wave have gone over and covered the nearshore areas of Hong Kong. However, the fact is that on 3rd August, 2021, no major flooding or tsunamis have happened near Hong Kong. This is a result in the stark difference in detail levels between the two datasets. One with an accuracy of up to 2 meters, while another one has an accuracy in kilometres. Figure 18 illustrates how that might result in this undesirable visualization.


Info
Zoom: 14.584220728120004
Bearing: 0
Lon: 114.0362931895827
Lat: 22.32723466415184

Terrain ✓
Radar ✓
Tides ✓
Tesselator: [✓ Auto] Martini Delatin
Max Mesh Error ± 0 m
TCR 2021-09-16 Select...

Figure 17

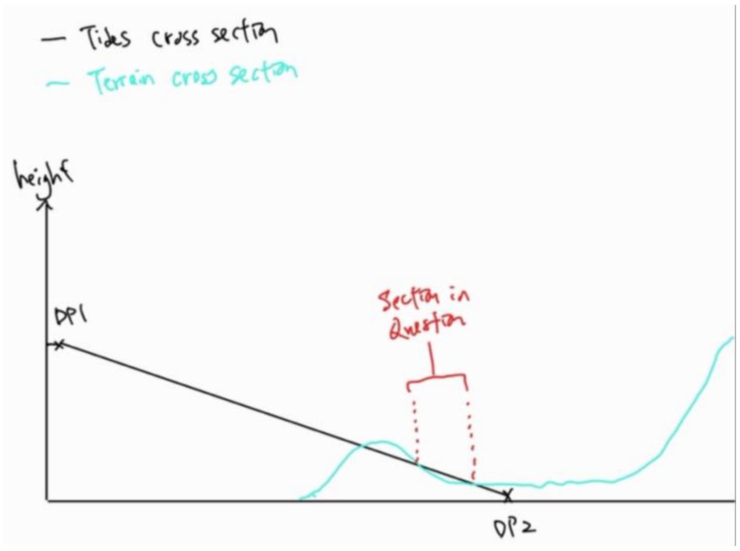
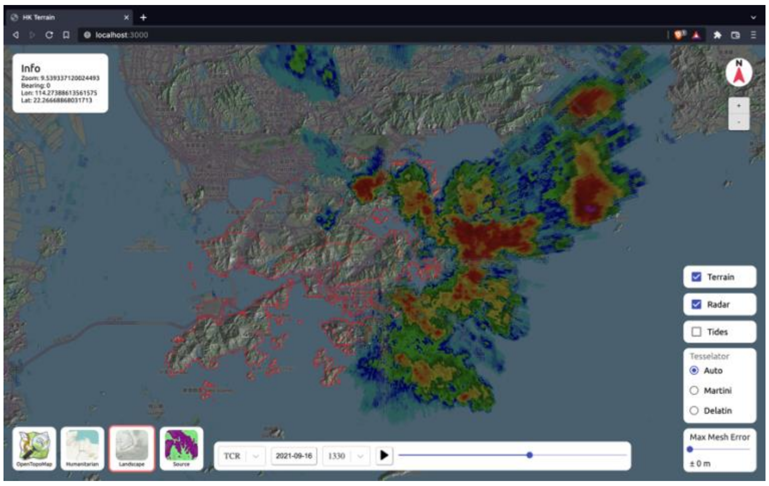


Figure 18

Figure 18 above illustrates in what situation the observation in Figure 17 might occur. The black line are the tidal datasets where the data points are very scattered. Only 2 data points are shown. The light blue line are the terrain layer where the data points are very dense. Highlighted in the red are area where the terrain there might not covered by the ocean but due to dataset accuracy differences, a bug in the visualization occurs.

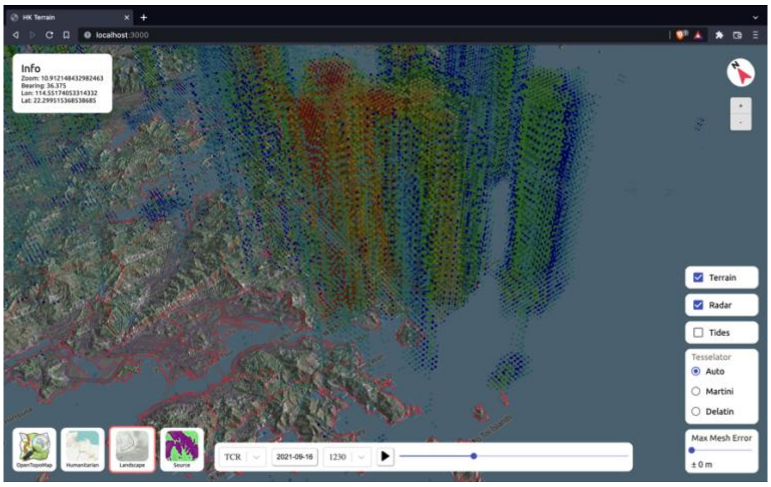
Radar. The processed PLY files are being used as the input to the Point Cloud layer of Deck.gl. The point cloud layer reads the x, y, z positions and color information and display the whole point cloud dataset. Figure 19 and 20 can be seen as an example with all the layers present. Figure 21 shows an animation of the radar data. The data being shown is the severe hail storm at 16th September, 2021.



Info
Zoom: 9.539337120024493
Bearing: 0
Lon: 114.27388613561575
Lat: 22.26668868031713

[✓ Terrain] [✓ Radar] Tides
Tesselator: [✓ Auto] Martini Delatin
Max Mesh Error ±0 m
TCR 2021-09-16 1330

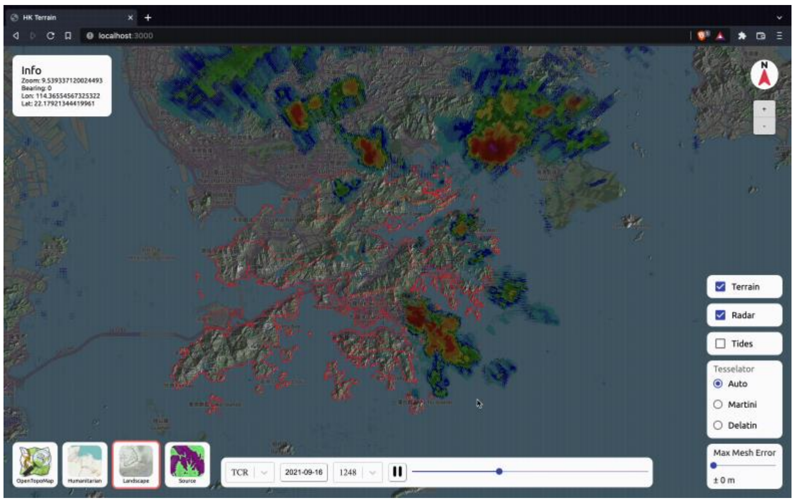
Figure 19: Top down view



Info
Zoom: 10.912148432982463
Bearing: 36.375
Lon: 114.55174053314332
Lat: 22.299515368538685

[✓ Terrain] [✓ Radar] Tides
Tesselator: [✓ Auto] Martini Delatin
Max Mesh Error ±0 m
TCR 2021-09-16 1230

Figure 20: Angled view



Info
Zoom: 9.539337120024493
Bearing: 0
Lon: 114.36554567325322
Lat: 22.17921344419961

[✓ Terrain] [✓ Radar] Tides
Tesselator: [✓ Auto] Martini Delatin
Max Mesh Error ±0 m
TCR 2021-09-16 1248

Figure 21: Radar GIF

Discussion

Network & Frontend Performance

Figure 22 shows the initial loading performance of the webpage with all things hosted locally on a 2020 13-inch M1 Macbook Pro with 16GB of United Memory following the local development structure mentioned in Project Structure above.

Before the 500 ms mark, bundled index.bundle.js were served from webpack. All the generated detailed tiled terrain are all served at the orange vertical line right after the 500 ms mark. Before the 1000 ms mark, local assets defined in the project that would be seen in later Figures were being loaded. And the long trails of assets loading up till around 4700 ms were all the responses from the external tile hosting services like Mapbox, OpenStreepMap, etc. in which the old HTTP/1.1 protocol were still being used.

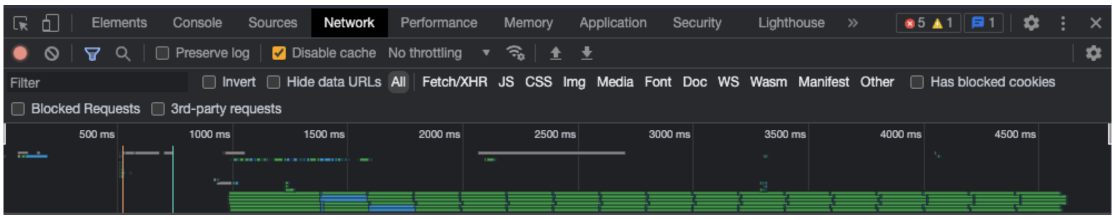
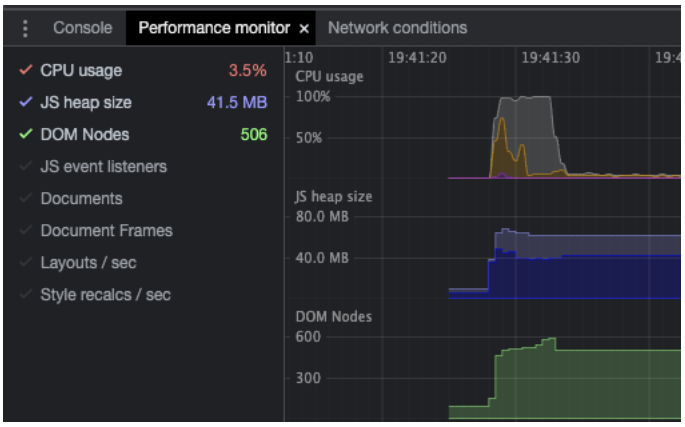


Figure 22: Network Performance Captured from Google Developer Console on a 2020 13" M1 Macbook Pro

As for other performance metrics, as shown in Figure 23 below. Upon initial load of the webpage, for a few seconds the CPU usage would be at 100% then with the web page finishing up loading, dropping back to 0%. And a reasonable JS heap size of 41.5 MB were used, compared to the 27.1 MB used when just loading in www.google.com.



CPU usage 3.5%
JS heap size 41.5 MB
DOM Nodes 506
JS event listeners
Documents
Document Frames
Layouts / sec
Style recalcs / sec

Figure 23: Performance Monitor on initial launch of the website

A wide range of other devices were also tested including Windows machines with dedicated graphics cards, Windows laptops, tablets and mobile phones. All of the device successfully loaded the webpage and were able visualize the datasets. The performance, however, is completely dependent on the device’s graphical power since all rendering is being done on device in real-time. Figure 24 showing the website being tested.

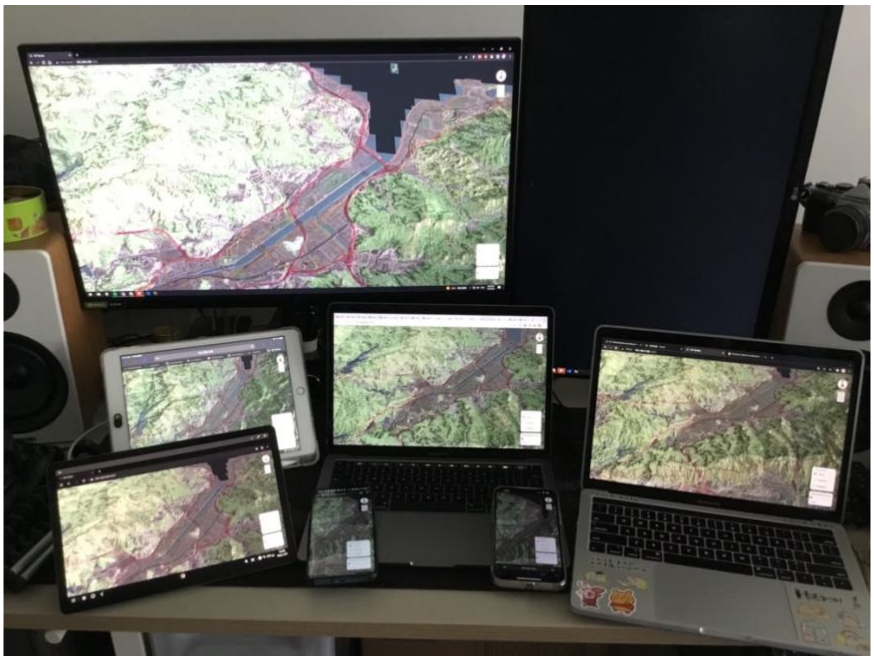


Figure 24

Limitations & Optimizations

Performance Optimizations. Further optimization can be made on the frontend by serving our own satellite textures with the new HTTP/2 protocol which would decrease the loading times of the page significantly.

Furthermore, as the Express framework does not currently support compression when using HTTP/2 (michael42, 2021), the files were all served with their full size. With compression support, served file sizes can be decrease further for faster routing times, thus faster loading times. However, compression does not work very well with image or binary files, as they are already compressed formats (Braun, 2017), but nevertheless it would result in smaller file sizes.

Mentioned in the data processing section above, construction and parsing of custom PLY files would also allow for smaller PLY files since a data field is duplicated currently. Smaller PLY file sizes also means faster loading times thus better performance.

More compact data storage formats can also be used to store the data. During the development process, Google's Protocol Buffer were explored and trialled to store the radar data. Protocol Buffer is a new binary data storage format that aims to replace XML, but smaller and faster (Google, 2022). However, it was not successful and PLY were used instead.

General Limitations. The general limitations of rendering complex multi-dimensional meteorological datasets in 3D is that sometimes the viewport will become too clustered with all the data present. As the screen is a 2D entity and features in the 3D space would be overlapped as seen on the screen. Sometimes, users might not be able to sense the 3D structure of the features at a glance or just looking at screenshots and data might be hidden away in the middle of data clusters. However, with interactivity features like zooming and panning with a smooth frame rate, it would be easier to understand the picture.

Performance Limitations. The performance of the webpage is completely dependent on the graphical rendering power of the device being used. Even with the all the optimizations and 0 loading time, rendering times for the multi-dimensional datasets in real-time would be the biggest bottleneck in performance.

Future Work.

In general, more elements in the existing datasets can be visualized. For example the wind vectors can be rendered alongside the significant wave height of the tidal datasets to better showcase the trend and overall picture of the ocean.

Frontend data processing tools will have to be added to allow user to select filter and select elements in specific layers of the dataset being displayed. For example, when visualizing 3D radar point clouds, users should be able to select and view individual layers, also filter out lower reflectivity values or vice versa such that the advantage of rendering datasets in 3D can be brought to the full potential.

Last but not least, the local storage data can be transferred to the fully fledged database in HKO in order to test the true performance of the whole prototype stack.

Conclusion

In this project, a full stack prototype showcasing 3D tiled terrain, tidal data and radar reflectivity data were successfully implemented, with a smooth supporting data processing pipeline and flexible database to back up the system. Cutting edge 3D rendering and web technologies were also utilized to show the potential of multi-dimensional meteorological data rendering in real-time in browser. With the flexibility of a web-based platform, data can be visualized in more advanced and innovative ways that might not be possible traditionally with proprietary software or in-house rendering software.

With a new dimension available for visualization, it would up to actual weather forecasters to identify whether it would be beneficial for them to see the meteorological elements rendered in the 3D viewport or not. And subsequently how such visualizations can benefit them when it comes to forecasting weather more accurately.

I sincerely hope this project is useful and look forward to spark and contribute to a new generation of meteorological data visualizations.

References

Shuman, F. G. (1989). History of Numerical Weather Prediction at the National Meteorological Center. *American Meteorological Society*, 286-296.

Gramelsberger, G. (2009). Conceiving Meteorology as the exact science of the atmosphere: Vilhelm Bjerknes's paper of 1904 as a milestone. *Meteorologische Zeitschrift*, 669-673.

Observatory, H. K. (2021). *Numerical Weather Prediction Models*. Retrieved from https://www.hko.gov.hk/en/aviat/amt/nwp.htm

Francis Reddy, U. o. (2015, April 29). *NASA 3D Resources*. Retrieved from https://nasa3d.arc.nasa.gov/detail/hurricane-katrina

Matthew Bilskie, L. S. (2019). *Tidal Visualization*. Retrieved from http://lantern.ncsa.illinois.edu/Vis/Tidal/

Observatory, H. K. (2021). *Long-range Weather Radars*. Retrieved from https://www.hko.gov.hk/en/aviat/amt/tms_radar.htm

National Weather Service (NWS), N. O. (2009). *WaveWatch III Model Description*. Retrieved from https://polar.ncep.noaa.gov/waves/wavewatch/

SourceForge. (2021). *SWAN*. Retrieved from https://swanmodel.sourceforge.io/

Meta Platforms, I. (2022). *React*. Retrieved from https://reactjs.org/

Vis.gl. (2021). *DECK.GL*. Retrieved from deck.gl

Vis.gl. (2021). Retrieved from LOADERS.GL: loaders.gl

Vis.gl. (2021). *LOADERS.GL*. Retrieved from loaders.gl

Vis.gl. (2021). Retrieved from Loaders.gl: loaders.gl

Vis.gl. (2021). *react-map-gl*. Retrieved from https://visgl.github.io/react-map-gl/

StrongLoop, I. a. (2017). *Express*. Retrieved from expressjs.com

Ilya Grigorik, S. (2019). *Introduction to HTTP/2*. Retrieved from https://developers.google.com/web/fundamentals/performance/http2

Romuald Corbel, E. S. (2016). HTTP/1.1 pipelining vs HTTP2 in-the-clear: performance comparison. *NOTERE.* Paris.

Jeffrey Whitaker, O. s. (2021). *Transformer*. Retrieved from https://pyproj4.github.io/pyproj/stable/api/transformer.html

Frank Warmerdam, E. R. (2022). *VRT - GDAL Virtual Format*. Retrieved from https://gdal.org/drivers/raster/vrt.html

Open Geospatial Consortium. (2022). *OGC GeoTIFF Standard*. Retrieved from https://www.ogc.org/standards/geotiff

Frank Warmerdam, E. R. (2022). *gdal_grid*. Retrieved from https://gdal.org/programs/gdal_grid.html

Frank Warmerdam, E. R. (2022). *gdal_merge.py*. Retrieved from https://gdal.org/programs/gdal_merge.html

Open Geospatial Consortium Inc. (2010). Retrieved from OpenGIS Web Map Tile Service Implementation Standard: file:///Users/laijackylai/Downloads/07-057r7_Web_Map_Tile_Service_Standard.pdf

Nvkelso, I. L. (2017). Retrieved from Types of Terrain Tiles: https://github.com/tilezen/joerd/blob/master/docs/formats.md

NCAR. (2022). *PyNIO.* Retrieved from https://www.pyngl.ucar.edu/Nio.shtml

Fredrik Lundh, A. C. (2022). *Pillow Image Module.* Retrieved from https://pillow.readthedocs.io/en/stable/reference/Image.html

Bourke, P. (2021). *PLY - Polygon File Format*. Retrieved from http://paulbourke.net/dataformats/ply/

mrdoob, M. a. (2022). *PLYLoader.js*. Retrieved from https://github.com/mrdoob/three.js/blob/dev/examples/js/loaders/PLYLoader.js

dranjan. (2021). *python-plyfile*. Retrieved from https://github.com/dranjan/python-plyfile

Weusthoff, T. a. (2008). Basic characteristis of post-frontal shower precipitation. *Meteorologische Zeitschrift*, 793-805.

Mapbox. (2022). *Access Elevation Data*. Retrieved from https://docs.mapbox.com/data/tilesets/guides/access-elevation-data/#decode-data

Barron, K. (2021). Retrieved from mapbox/martini: https://github.com/mapbox/martini

William Evans, D. K. (1997). Right-Triangled Irregular Networks.

mourner, p. (2021). *mapbox/delatin*. Retrieved from https://github.com/mapbox/delatin

michael42. (2021). *Support for Node.js 8 native http2*. Retrieved from expressjs/compression: https://github.com/expressjs/compression/issues/122

Braun, M. (2017). *why png size doesn't change after using http gzip compression*. Retrieved from stack overflow: https://stackoverflow.com/questions/11289369/why-png-size-doesnt-change-after-using-http-gzip-compression

Google. (2022). Retrieved from Protocol Buffers: https://developers.google.com/protocol-buffers

Survey and Mapping Office Lands Department. (2018). Retrieved from Explanatory Notes on Geodetic Datums in Hong Kong: https://www.geodetic.gov.hk/common/data/pdf/explanatorynotes.pdf

National Geospatial-Intelligence Agency. (2005). Retrieved from World Geodetic System 1984: https://www.unoosa.org/pdf/icg/2012/template/WGS_84.pdf

NASA. (2022). *Panoply netCDF, HDF and GRIB Data Viewer* . Retrieved from https://www.giss.nasa.gov/tools/panoply/

Appendix

Coordinate systems

HK1980. The HK1980 Grid system is a local rectangular grid system used extensively by the HKSAR government for measuring Hong Kong related meteorological datasets, like cadastral, engineering surveying and mapping. It is based on the HK80 Datum and Transverse Mercator projection. The origin of the projection was located at old Trig 2, which is now gone (Survey and Mapping Office Lands Department, 2018). Figure 25 below illustrates the HK80 Geodetic Datum.

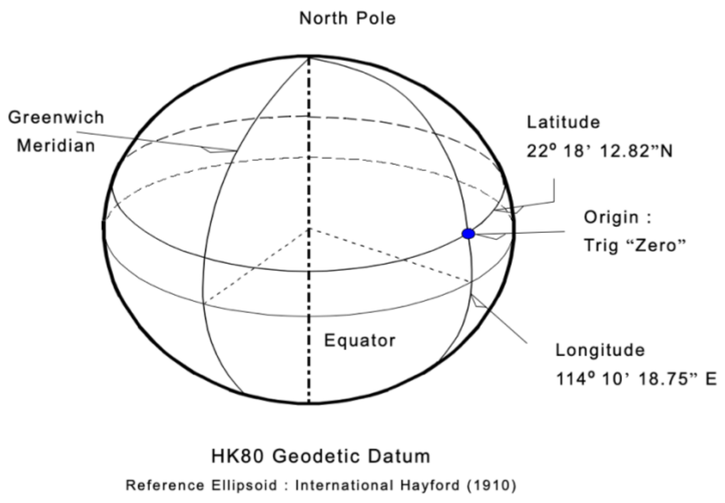


Figure 25

WGS84. WGS 84 is an earth-centred, earth-fixed Cartesian coordinate system and is the most commonly used coordinate system in mapping. WGS 83 is based on a consistent set of constants and model parameters that describe the Earth's size, shape, and gravity and geomagnetic fields. Its origin is located at the center of Earth's mass including all lands and oceans. The Z-axis corresponds to the direction of the BIH Conventional Terrestrial Pole, while the X-axis corresponds to the intersection of the IERS Reference Meridian and the plane passing through the origin and normal of the Z-axis and the Y-axis is a right-handed axis to the X-axis and completing the coordinate system (National Geospatial-Intelligence Agency, 2005). Figure 26 below shows the illustration of the WGS 84 datum.

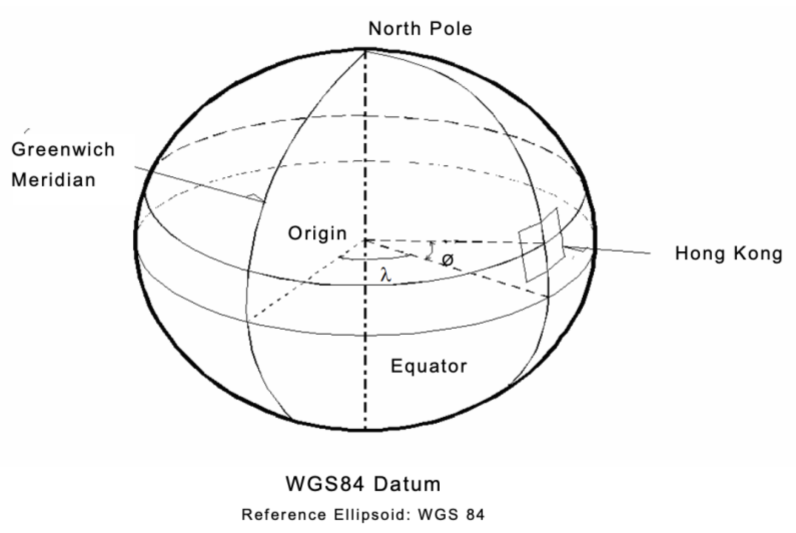


Figure 26

Panoply. Panoply is cross-platform tool that allows users to inspect various meteorological datasets like GRIB, NetCDF, HDF and other datasets (NASA, 2022). It allows users to explore elements in those datasets with complete metadata and descriptions available. Also, it can be used to plot 2D diagrams for user chosen elements.