User Guide

Morton Analytics AIS Vessel Tracking Dashboard

Prepared by:

Sixth Sense Solutions

Table of Contents

---

1. Set up

Prerequisites:

- Python 3.x installed on your system.
- Required Python packages listed in requirements.txt (for both repositories)

Clone the repositories

git clone https://github.com/tedtop/coast_guard_ais.git

git clone https://github.com/tedtop/ais_dashboard.git

From Terminal:

Navigate to Respective Project Directories

cd coast_guard_ais (or cd ais_dashboard)

Create/Activate a Virtual Environment

python -m venv ais_project (creation)

On Windows:

ais_project\Scripts\activate

On macOS or Linux:

source ais_project/bin/activate

Install Required Packages

pip install -r requirements.txt

Note: pip installing requirements.txt needs to be done for both the coast guard ais data processor *and* the ais dashboard as both use different packages.

Run the software

To run the data processor (assuming that the user is in the coast_guard_ais directory):

python zip2parquet.py

This will begin downloading and converting data from 2024 to parquet files and formatting in the hive partitioning style. Options for this are handled through the configuration section of the zip2parquet.py file. Once the data has been downloaded and partitioned on your system, the dashboard is able to be used.

A file containing all the data named “year=20xx” will have been created. Move this into an empty folder named “coast_guard_ais”. Further, move this entire data folder to the ais_dashboard directory.

If you’ve made it this far without problems, congratulations! Using the dashboard itself is much easier than all of the setup.

To run the dashboard

1) Navigate to the ais_dashboard folder.

   cd ais_dashboard

2) Activate your Python virtual environment.

   source ais_project

3) Run the dashboard.

   panel serve --autoreload --show main.py --static-dirs assets=assets

2. Project Purpose

The AIS Ship Traffic Dashboard is a prototype of a web-based geospatial visualization tool designed to help users explore maritime vessel activity using AIS (Automatic Identification System) data. It visualizes aggregated ship locations as heat maps over time, allowing analysts to observe traffic patterns, congestion, and trends across the globe.

In summary, the dashboard processes time-stamped parquet files, converts them into visual heatmaps using Datashader, and presents these images on an interactive map with playback and time-based filtering capabilities.

---

3. System Description

The dashboard is organized into the following core modules:

- **Data Renderer (renderer.py)**: Converts raw hourly AIS parquet data into 2D PNG visualizations.

- **Control Panel (control.py)**: Provides date and interval selectors for controlling the visualization range and triggers rendering.
- **Viewer Panel (viewer.py)**: Loads and displays the PNG visualizations as map overlays with animation controls.
- **Dashboard Wrapper (dashboard.py)**: Combines viewer and controls into a cohesive Panel-based interface.
- **Entry Script (main.py)**: Launches the dashboard server.
- **Configuration (config.py)**: Centralized setup of file paths, map ranges, and available intervals.

Similarly, the data processing system consists of:

- **Data Retrieval**: Extracts and downloads AIS ZIP files from NOAA's data repository.
- **Data Processing**: Reads and processes CSV files in chunks to manage memory usage effectively.
- **Data Conversion**: Transforms CSV data into Apache Parquet format for optimized storage and querying.
- **Data Organization**: Structures the processed data into a time-based hierarchical directory system.

---

4. Technologies Used

Component	Purpose
Panel	Interactive web UI framework
Holoviews	For geospatial overlays with zoom/pan capabilities
Datashader	Efficiently renders massive datasets into raster images
PyArrow/Dask	Efficient reading of columnar Parquet files
PIL / NumPy	Preprocessing image overlays for visualization

Param	Data-driven widget bindings and reactive programming
Colorcet	Color mapping for fire-style heatmaps

5. Components (Dashboard and Data Processor)

Data Processor

Extractor zip2parquet.py**:**

- Scrapes NOAA's AIS data page for available ZIP file URLs.
- Downloads each ZIP file while displaying progress.
- Extract CSV files from the downloaded ZIP archives.
- Process CSV data in chunks to handle large files efficiently.
- Convert each chunk into Parquet format.

Data Organization:

- Save the Parquet files into a structured directory (hive-partitioned) hierarchy based on timestamp

Example of data organization:

coast_guard_ais/year=2024/month=07/day=21/hour=11/AIS_2024_07_21_processed_hour11.parquet

Dashboard

Control Panel (AISRenderControl)

- **Date Range Selector**: Choose the time span for data rendering
- **Interval Selector**: Choose aggregation granularity (e.g., "1 Day", "7 Days")
- **Generate Button**: Launches PNG rendering across selected range
- **Progress Bar**: Displays rendering progress
- **Log Viewer**: Displays recent messages from the rendering engine

Viewer Panel (AISMapViewer)

- Dynamically loads available PNG frames
- Displays them over CartoDark basemap
- Controls include:
    - ▶ Play/Pause animation
    - ◀ Move backward
    - ▶ Move forward
- Timestamp display updates in real-time

Renderer (AISRenderer)

- Loops over a date range in N-hour increments
- Loads parquet data from:

  base_path/year=YYYY/month=MM/day=DD/hour=HH/

It is of utmost importance that the base_path is correctly set in AISRenderer, and that the user has a properly configured hive-style partitioned DFS of AIS data.

Further, the renderer transforms coordinates, rasterizes them, and saves:

output_root/interval/ais_<timestamp>.png

6. Training Materials

Launching the Dashboard
Assuming that the user has a Python 3 environment with all necessary packages installed, the user will run this from their terminal:

panel serve --autoreload --show main.py --static-dirs assets=assets

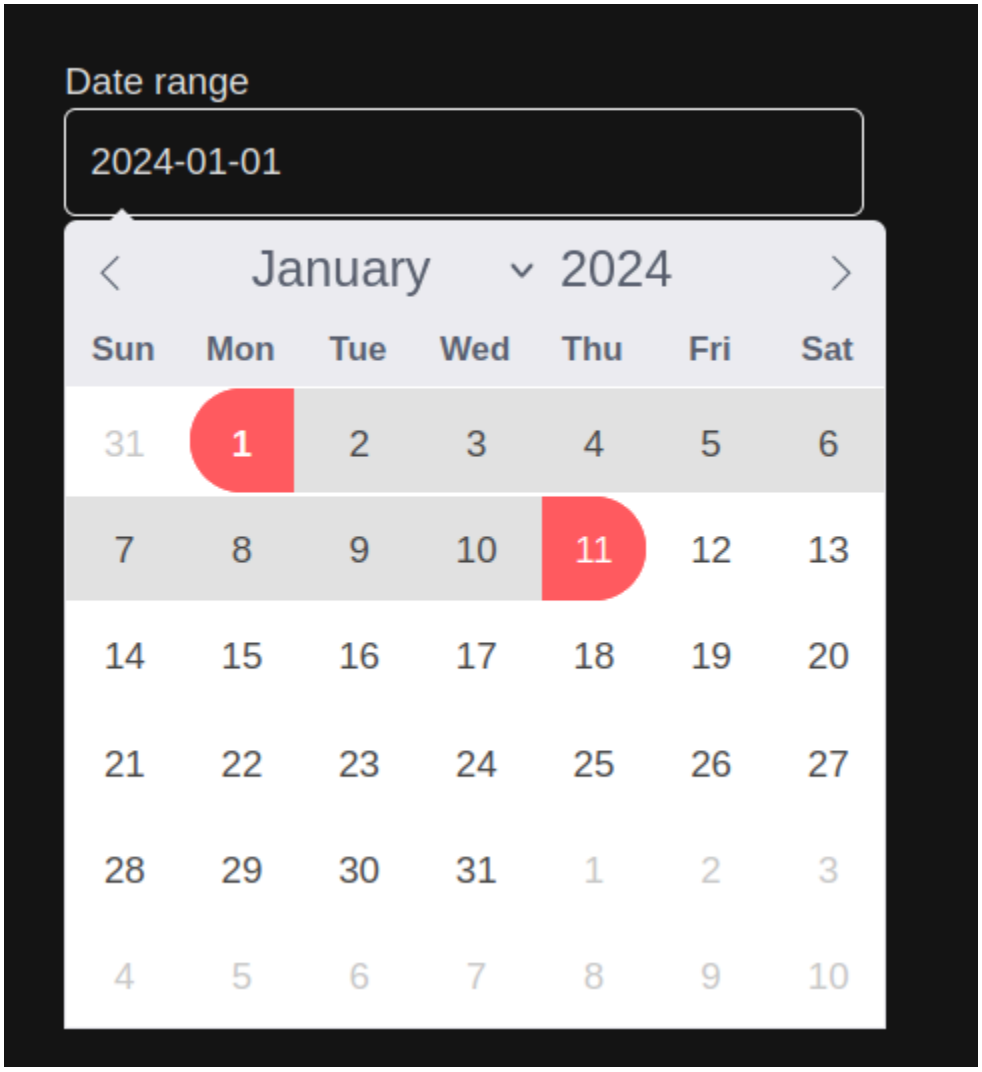
This will:

- Load the dashboard template
- Initialize the control and viewer components
- Open a browser window with the interface

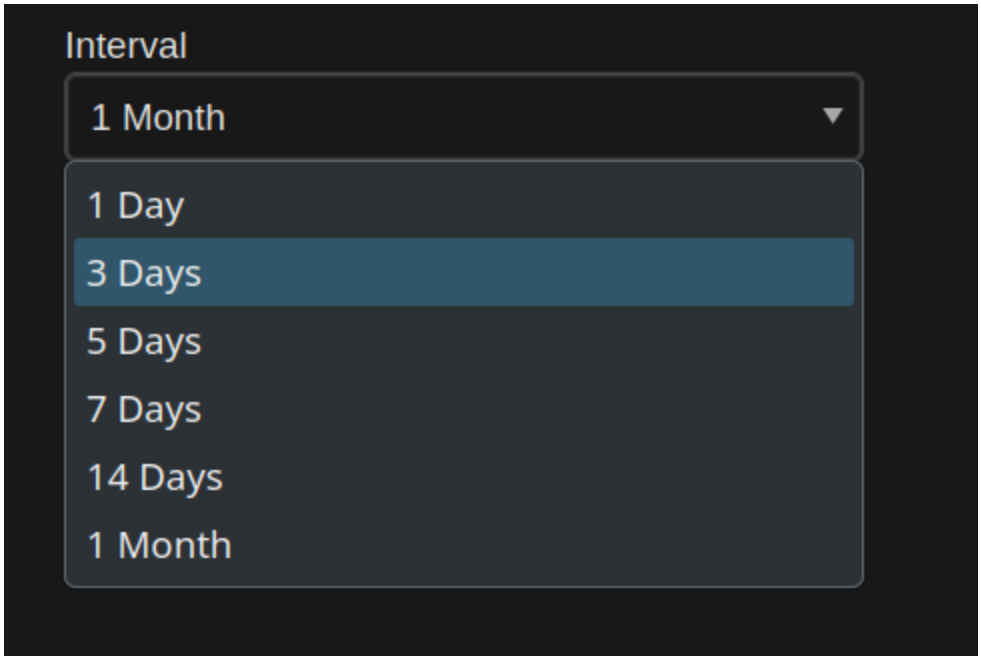
Of minor note: an internet connection is required to run this software properly, as it pulls a map from OpenStreetMap.com. Without an internet connection, no map will appear for the data to project onto.

Using the Control Panel

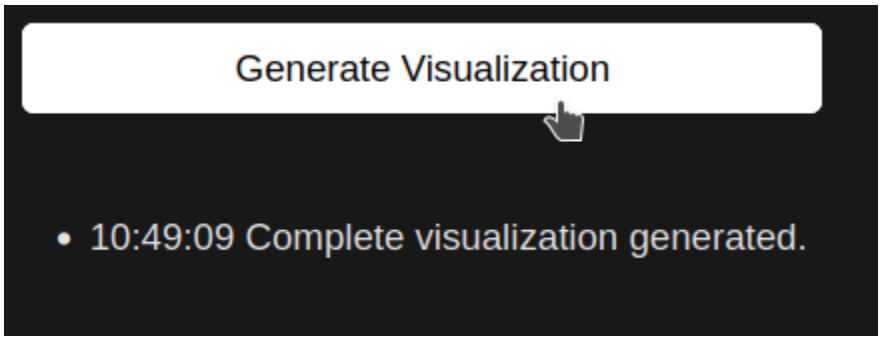
1. **Select Date Range**
   E.g., Jan 1, 2024 to Jan 11, 2024



2. **Select Interval**
   E.g., "1 Day" or "3 Days"



3. **Click Generate Visualization**
   This will trigger AISRenderer.render() to build new PNG frames



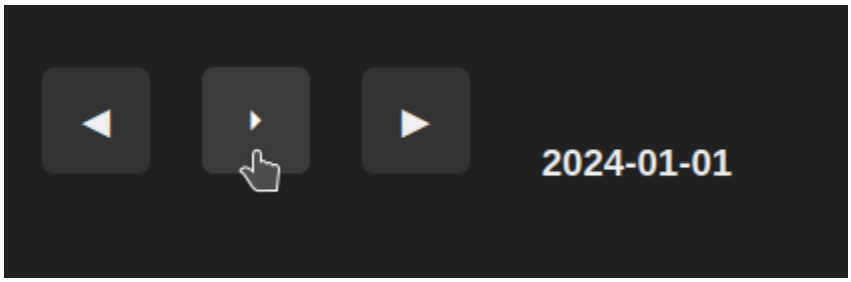
4. **Watch the Log Pane and Progress Bar**
   - Status messages update in real time

   - Of note, there is a logging mode that is disabled by default within the control.py file. If set to true, logging messages for each frame generated will be displayed as the visualizer is generating images for debugging purposes. While disabled, it will only display that the visualization was successful.

Using the Viewer

Once frames are generated:

- Click ▸ to play animation
- Use ◀ and ▶ to step through frames
- Timestamp shows currently loaded frame



---

7. Implementation Guide

Data Directory Structure

Expected AIS .parquet files should follow:

base_path/year=YYYY/month=MM/day=DD/hour=HH/AIS_YYYY_MM_DD_processed_hourHH.parquet

Each file should contain at least:

- LAT
- LON

Manual Rendering Script (if needed)

```
from renderer import AISRenderer

from config import base_path, output_root, canvas


renderer = AISRenderer(base_path, output_root, canvas)

renderer.render("2024-01-01", "2024-01-10", "1 Day")
```

8. Maintenance Instructions

- To maintain usability:
    - Ensure new parquet files are written to the correct directory structure
    - Run Generate Visualization for new date ranges as data is added.
- To reset:
    - Delete PNGs in output_root/interval/*
    - Re-run the renderer for the desired range