# Developer's Guide

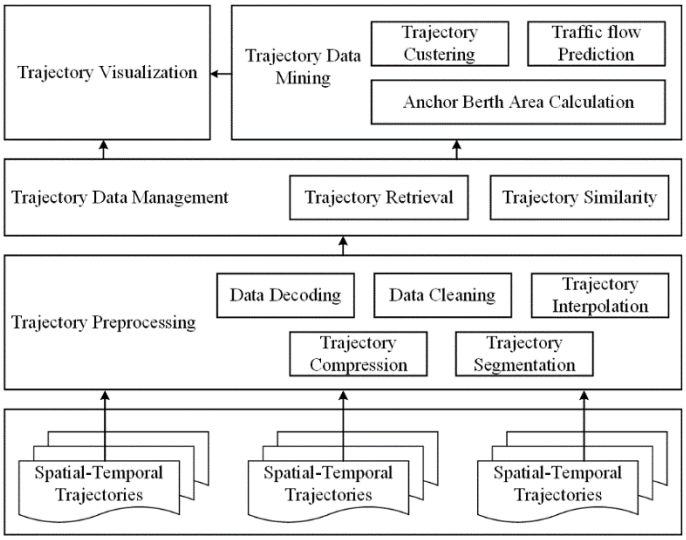



Figure 1 The technical architecture of PyVT

The toolkit PyVT mainly includes four modules: trajectory preprocessing, trajectory data management, task mining and trajectory visualization.

## 1 Trajectory preprocessing

### 1.1 Message decoding

PyVT provides AIS message decoding script, realizes decoding and outputting 27 kinds of AIS raw messages, and completes extracting ship trajectory data from the raw data.

How to decode the AIS message is shown:

```
from pyais import decode

# Decode a single part using decode
decoded = decode(b"!AIVDM,1,1,,B,15NG6V0P01G?cFhE`R2IU?wn28R>,0*05")
```

### 1.2 Trajectory cleaning

For the discovery and processing of noise points, PyVT uses two processing methods, one is heuristic filtering and the other is mean filtering.

How to use “heuristic_clean” is as follows:

```
from pyais import traj_clean

# df is the dataframe of vessel trajectories, the vthreshold is max speed threshold
cleandf = traj_clean.heuristic_clean(df, vthreshold=25)
```

How to use “sw_clean” is as follows:

```
from pyais import traj_clean

# df is the dataframe of vessel trajectories, the sw is the size of sliding window
```

```
cleandf = traj_clean. sw_clean (df, sw=5)
```

### 1.3 Trajectory interpolation

PyVT uses the PCHIP (Piecewise Cubic Hermite Interpolating Polynomial) to interpolate the breakpoints between trajectories, which improves the continuity and integrity of trajectory data.

```
from pyais import traj_interpolation

# traj_file is the raw trajectory filename, res is the time resolution, num is None
traj_data_interp = traj_interpolation.traj_interpolate(traj_file, res= 30, num= None)
```

### 1.4 Trajectory segmentation

PyVT selects the time interval as the segmentation condition to segment the trajectory for the first time, and performs the final segmentation of the trajectory after the first segmentation according to the identification of the stay point. PyVT provides two methods for vessel stay points identification proposed in this paper, the method SPT considering the geometric characteristics of the vessel trajectory and the method ST-SPT considering the spatio-temporal continuity of the vessel trajectory and based on the ST-DBSCAN clustering algorithm.

How to segment the trajectories is shown below:

```
from pyais import traj_segment

# trajdata is the raw trajectory filename, tsinterval is the time threshold
segmenteddata = traj_segment.segment(trajdata, tsinterval = 1500)
```

How to do stay point detection is shown below:

```
from pyais import traj_stay

# stay_st_detect is the ST-SPT method, stay_spt_detect is the method SPT
trajdata = traj_stay.stay_st_detect(traj_file)
trajdata = traj_stay.stay_spt_detect(traj_file)
```

### 1.5 Trajectory compression

PyVT provides two AIS trajectory compression algorithms, including the classic Douglas-Peucker algorithm and a SBC compression algorithm proposed in this paper that considers ship motion characteristics.

How to compress trajectories using the Douglas-Peucker algorithm:

```
from pyais.traj_compress import douglaspeucker

dp = douglaspeucker(trajdata[['DRLATITUDE', 'DRLONGITUDE']].values)
epsilon = dp.avg()
mask = dp.rdp(group[['DRLATITUDE', 'DRLONGITUDE']].values, epsilon, algo="iter",
              return_mask=True)
compresseddata = trajdata[mask]
```

How to compress trajectories using the SBC algorithm:

```
from pyais import traj_compress
```

```
# param s: raw trajectory, param max_dist_threhold: distance threshold, param max_spd_threhold: speed threshold

compresseddata = traj_compress.sbc(s = trajdata, max_dist_threhold = 0.025, max_spd_threhold = 0.25)
```

# 2 Trajectory data management

Trajectory data management includes two aspects: trajectory query and trajectory similarity measurement.

## 2.1 Trajectory query

By loading external files or setting the range of interest region (longitude and latitude coordinates of rectangular region) and time span (start and end time), all ships passing through the region of interest in the set time period can be queried. The single ship trajectory can be queried quickly by setting the call sign or MMSI and time span of the ship of interest.

Example of trajectory acquisition:

```
from pyais import chunk_read

# the param input: filename of the input file, the param output: filename of the output file, the param mmsi: MMSI of ship
chunk_read.read_ais_data_mmsi(input="1.csv",output='test.csv',mmsi='244726000')
```

## 2.2 Trajectory similarity measurement

In the PyVT, Fast Dynamic Time Warping (FastDTW) distance is used to express the similarity between two trajectories.

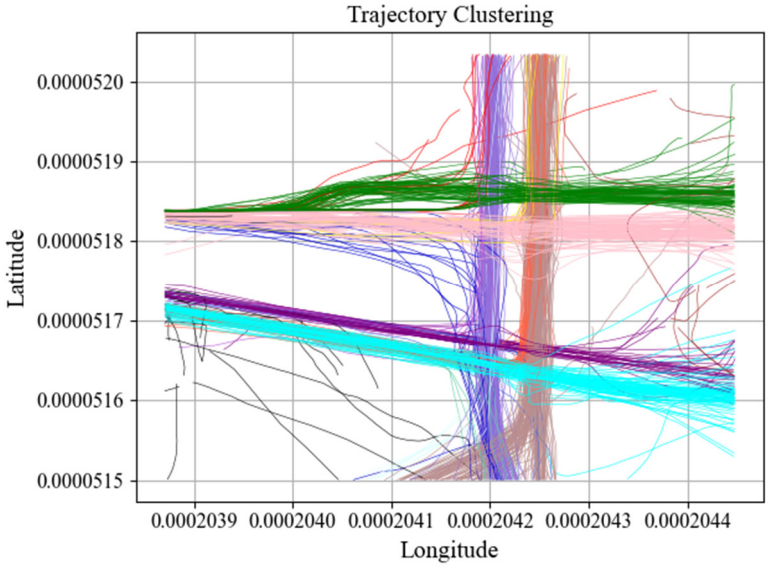
Please refer to the sample program for the usage of the code.

# 3 Trajectory mining

## 3.1 Trajectory clustering

PyVT uses HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) algorithm to cluster AIS trajectories.

The file “traj_mining\traj_cluster” gives a detailed example of trajectory clustering.




## 3.2 Calculation of ship anchor berth area

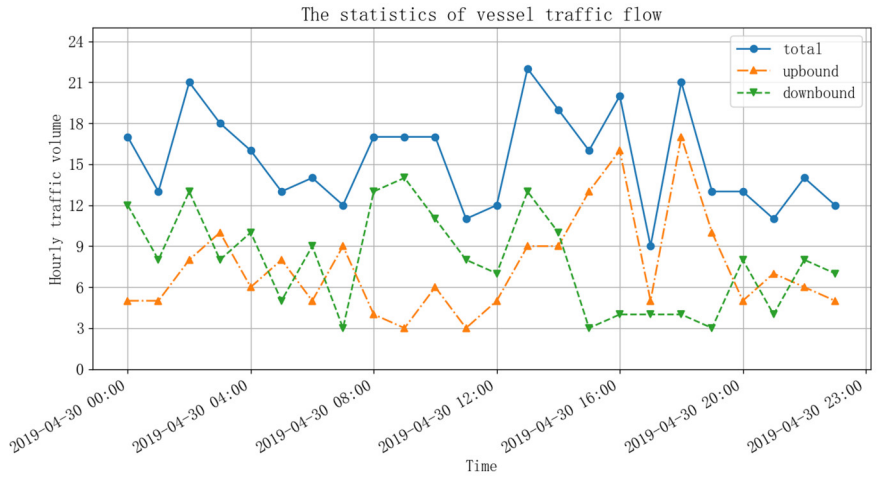
PyVT combines the ship stay point identification algorithm with computational geometry, and proposes an anchor berth area calculation method using AIS trajectory data. This method can effectively identify the ship's anchor points and accurately calculate the actual anchor berth area of the ship.

```
Step1:
# Ship stay point identification
from pyais import traj_stay
trajdata = traj_stay.stay_st_detect('./data/1.csv')
Step2:
# Use the monotone chain convex hull algorithm to calculate the anchor berth boundary
L = GiftWrapping(points)
Step3:
# Use the shoelace theorem and the polygon centroid formula to calculate the ship anchor berth
area and the approximate position of ship's anchor respectively.
mj = shoelace(L.tolist())
```

## 3.3 Vessel traffic flow forecasting

Using ship AIS data, PyVT proposes a statistical method of ship traffic flow, and performs multi-step forecasting of ship traffic flow based on Long Short-Term Memory Encoder-Decoder (LSTM-ED) prediction model.

Please refer to “traj_mining\traffic_flow_predict” for the detailed process of the prediction of ship traffic flow.




# 4 Trajectory visualization

## 4.1 Multi-ship trajectory display

PyVT has developed a multi-ship trajectory display function, importing trajectory data by loading an external CSV file, and setting the region of interest and time span to filter the trajectory data to be displayed.

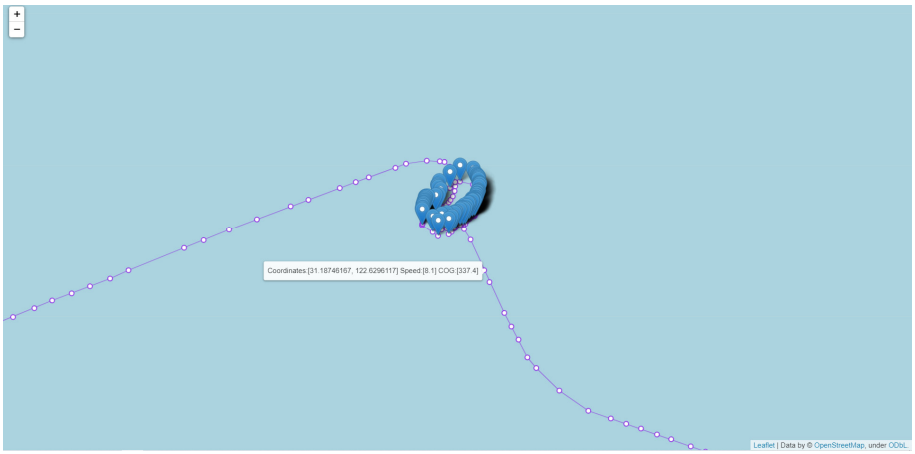
The file “traj_show” gives a detailed example of trajectory visualization.

The sample code of multi-ship trajectory display is as follows:


```
csv_path = r'./data'
draw_trajs(csv_path) # csv_path is the file path of trajectory data
```

## 4.2 Single-ship trajectory display

PyVT has developed a single-ship trajectory display function, which can display micro-information such as the position, speed, and heading of the ship, and perform semantic labelling of ship trajectories such as berthing and anchoring.




The sample code of single-ship trajectory display is as follows:

```
csv_path = r'./data'
draw_single_traj(csv_path) # csv_path is the file path of trajectory data
```