SoftwareX 19 (2022) 101176

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

PTRAIL — A python package for parallel trajectory data preprocessing

Salman Haidri [a], Yaksh J. Haranwala [a], Vania Bogorny [c], Chiara Renso [b], Vinicius Prado da Fonseca [a], Amilcar Soares [a,*]

[a] Department of Computer Science, Memorial University, St. John's, S.J. Crew Building, EN-2021, NL A1B 3X5, Canada
[b] Institute of Science and Technology A. Faedo, National Research Council of Italy, Italy
[c] Universidade Federal de Santa Catarina (UFSC), Brazil

ARTICLE INFO

Article history:
Received 1 September 2021
Received in revised form 16 March 2022
Accepted 22 July 2022

Keywords:
Trajectory preprocessing
Feature engineering
Parallel processing
Vectorization

ABSTRACT

Trajectory data represents a trace of an object that changes its position in space over time. This kind of data is complex to handle and analyze, since it is generally produced in huge quantities, often prone to errors generated by the geolocation device, human mishandling, or area coverage limitations. Therefore, there is a need for software specifically tailored to preprocess trajectory data. In this work we propose PTRAIL, a python package offering several trajectory preprocessing steps, including filtering, feature extraction, and interpolation. PTRAIL uses parallel computation and vectorization, being suitable for large datasets and fast compared to other python libraries.

© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Current code version	v0.3Beta
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00157
Code Ocean compute capsule	None
Legal Code License	BSD 3-Clause "New" or "Revised" License
Code versioning system used	git
Software code languages, tools, and services used	python, jupyter
Compilation requirements, operating environments & dependencies	numpy, pandas, geopandas, scikit-learn, hampel, scipy, psutil, folium, matplotlib, osmnx, shapely
If available Link to developer documentation/manual	http://ptrail.readthedocs.io/en/latest/index.html
Support email for questions	amilcarsj@mun.ca

1. Motivation and significance

Processing and extracting meaningful insights regarding the movement of people, vehicles, vessels, and animals are nowadays the focus of attention in several academic and industry sectors. The traces of moving objects are generally called trajectories and can be informally defined as a temporal sequence of geo-locations of a moving object. Trajectory data often contain errors that occur due to device failure, human mistakes, or connectivity problems. Data in this form cannot be directly used to extract useful information during data analysis. Besides, trajectory data in its raw format (i.e., object id, geographical position, and time) may not be sufficient for extracting relevant patterns. Processing trajectories in parallel is also desired since the volume of this data is often significant. Indeed, there is a need for automating these preprocessing steps when dealing with trajectory data. We call as preprocessing all steps that prepare the raw trajectory data for analysis, which in general, include filtering, interpolation, and feature extraction.

Currently, many python packages (e.g., scikit-mobility [1], as PyMove [2], MovingPandas [3]) are available for trajectory data representation and manipulation. However, none of them focus on preprocessing trajectory data, and neither are they designed to use the available processing resources of a machine to its best. This work introduces PTRAIL, a parallel and vectorized computational library tailored for trajectory data preprocessing. More specifically, PTRAIL addresses the tasks of filtering, interpolation, and feature extraction performed over trajectory data.

Trajectory data preprocessing can be a tedious and time-consuming task. Our objective is to allow researchers to use

* Corresponding author.
E-mail address: amilcarsj@mun.ca (Amilcar Soares).

https://doi.org/10.1016/j.softx.2022.101176
2352-7110/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

our library to easily handle these steps using the maximum of their resources and easily answer higher-level research questions without the need of coding and validating preprocessing steps. The PTRAIL functionalities and its core have been used in several projects within our group, including feature engineering in the context of transportation mode detection [4], trajectory classification [5], trajectory annotation [6] and anomaly detection [7]. However, the idea of parallelizing and vectorizing the processes is the main novelty of the current version of our library.

PTRAIL has been developed in a way that it can be directly used in a Python 3 environment, and can be simply installed using the *pip install* command. After installing, the user can import several functionalities available in the library and use the documentation as a reference for the requirements to use them. First, the user should read trajectory data from a file, creating a PTRAILDataframe. Once the data is properly loaded and validated, all the library functionalities are enabled on the data. Furthermore, the library is open source, and therefore it can be distributed and modified by forking the repository. The code has been well documented for the user's convenience, and several code examples are available in the developer documentation.

The development of PTRAIL is built on top of state-of-the-art libraries that fulfilled several needs in its implementation. PTRAIL uses numpy [8] for storing a collection of data because numpy provides a wide array of mathematical functions which are more efficient than traditional Python lists. The PTRAILDataFrame is an extension of the pandas [9] DataFrame and inherits all its functionalities. The parallelization is done using python's built in multiprocessing module [10], which allows us to use multiple processors on a given machine by enabling concurrency and parallelism for each trajectory loaded as a PTRAILDataFrame. We also used geopandas [11] and shapely [12] functionalities for extracting features from the data. We also used the implementation of Hampel filters available in [13], and some functionalities of scipy [14]. The main point is that, in PTRAIL, the processing of the functionalities used from geopandas, shapely, Hampel filters, and scipy occur in parallel.

2. Software description

PTRAIL offers several trajectory data preprocessing steps, including temporal, kinematic, and contextual feature extraction, filtering, and trajectory interpolation. In PTRAIL we define temporal, kinematic, and semantic features as follows. *Temporal features* refers to the features that can be extracted based on the temporal dimension. Examples are the day of the week, hour, etc. *Kinematic Features* refers to those features which are concerned with the motion of the object, i.e., speed, acceleration, etc. *Contextual features* describe contextual information that is related to the movement and are generally available through geographical layers in the area where the movements occurred. Examples are the point of interest visited or the weather conditions. The feature extraction and filtering methods allow the user flexibility to manipulate the raw data to gather information according to their application need. Furthermore, parallel and vectorized processing has made feature extraction and filtering computationally much faster than current state-of-the-art trajectory processing libraries. It is crucial to point out that the parallelization of the processing only takes place when at least 100 trajectories are available. The main reason for this is that early empirical experiments with parallelization processes with our library showed that the features calculations and DataFrame merges take longer with fewer trajectories than just processing it linearly with no parallelization. Therefore, we embedded this minimal threshold of 100 trajectories in our code to reflect these findings.

2.1. Software architecture

The entire process of execution of a method is explained in Fig. 1. First, the trajectory points are fed as input for the PTRAILDataFrame. After loading a valid file, all functionalities are available. Each trajectory loaded as a PTRAILDataFrame is first vectorized and the applied functionality is executed in parallel. Early experiments with such a strategy showed that not overloading a machine's memory with huge matrices (expected when loading large trajectory datasets), but assembling one matrix per trajectory and parallelizing the process execution per trajectory, showed huge improvements in processing time. The processing sequence showed in Fig. 1 is a standard way to process trajectory data. For each functionality, each trajectory is loaded as a matrix and the requested functionality is ran in parallel. It is up to the user's discretion to use filters, interpolation, and extracting useful features from the data set since the processes are independent.

2.2. Software functionalities

PTRAIL has a multitude of functionalities to be used with trajectory data, and they are detailed in the following subsections.

2.2.1. Filtering

The first step when handling any raw data is to clean them by removing noise and errors. In the trajectory data preprocessing pipeline the same step is applied. PTRAIL offers hampel, kinematic (i.e., based on speed, etc.) and temporal filters for removing errors or noise from the trajectory data sets. Our library also provides some routines to remove duplicated trajectory points since this is common in some raw data collected by geolocation devices.

2.2.2. Interpolation

After cleaning the data, a common next step is to smoothen it in regions where data is not available. PTRAIL offers standard ways to interpolate the data in such regions. We provide a parallelized version of the Linear and Cubic interpolation methods from the scipy package [14]. We also provide implementations of the Kinematic [15] and Random Walk [16] strategies coded as vectors and running in parallel.

2.2.3. Feature extraction

Finally, after the former two steps, the user is able to extract trajectory features from the data. We provide some temporal feature extraction methods such as the extraction of the date, time, day of week from the timestamp. Such features are very useful when analyzing trajectories in contexts where analyzing repetitive actions by the moving object in particular temporal are essential, such as trajectories of people moving in cities. The kinematic features are essential to understand and characterize how the moving object moves over time. PTRAIL is able to calculate kinematic features such as the traveled distance, speed, acceleration, bearing, bearing rate etc. Finally, if geographical layers about the study area are available, the user is able to create contextual features relating the movement with such layers. We provide functions that will create the moving object's visited location (inside geometry), nearest Point of Interest to a given location, functionalities to check whether trajectories lie inside a geometry or if they intersect inside one are also provided as contextual features of PTRAIL.

3. Illustrative examples

The PTRAIL gitHub repository currently contains numerous examples explaining its usage.[1] In the next two subsections we show two examples using PTRAIL.

[1] https://github.com/YakshHaranwala/PTRAIL/tree/main/examples

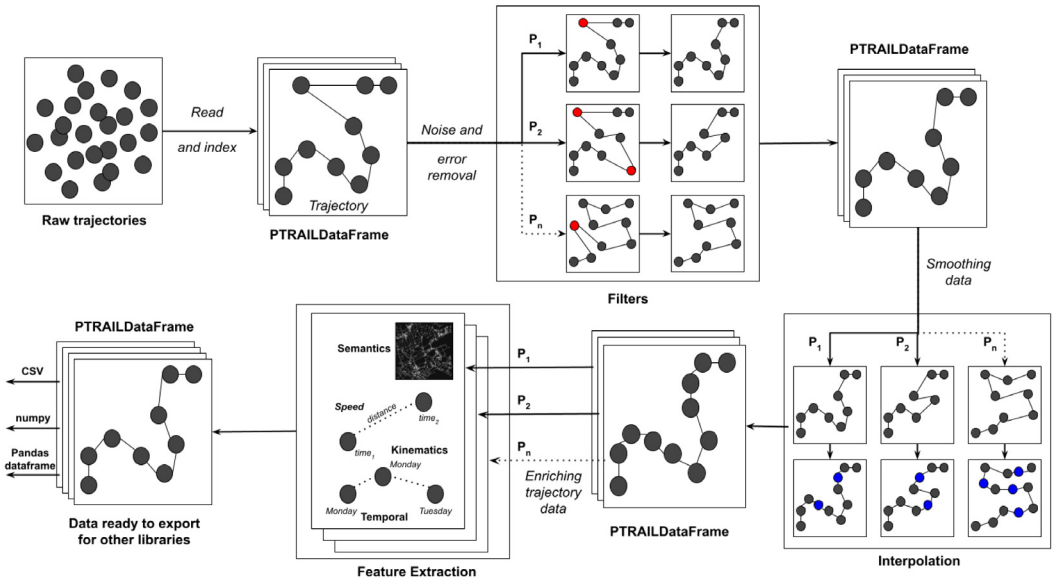

Fig. 1. The PTRAIL pipeline starts by loading a PTRAIL dataframe. After, the user can apply filters for removing noise and/or smoothing the data with interpolation. Finally, features can be extracted from a higher quality data set. In the end, the data can be exported to common formats and be used with another libraries.

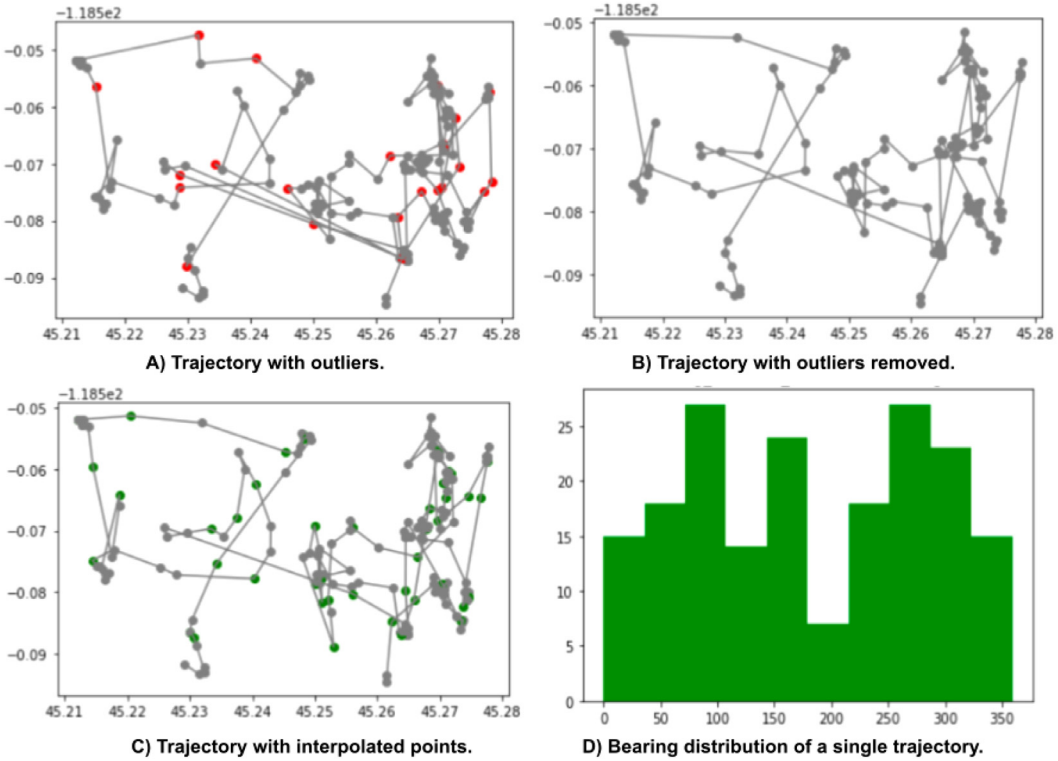

Fig. 2. An illustrative example of the use of PTRAIL.

3.1. Animals dataset

One of such examples with code snippets and outputs is illustrated in Fig. 2 and is available here.[2] We show a code snippet in Fig. 3 where we load a trajectory dataset with data regarding movements of cattle, elk and deer from the Starkey project [17]. We load the data in lines 8–9 informing the latitude, longitude, time and trajectory id columns. Then, we get a single trajectory with data of an elk to pass over our the full PTRAIL pipeline (line 12). We decided to use a single trajectory for avoiding data cluttering of the outcomes from the PTRAIL pipeline in the example discussed in this paper. First, we create a kinematic feature with the distance between two consecutive points (line 16) since this is needed to apply a Hampel filter (lines 17). After, we interpolate the data using the cubic interpolation (lines 19–20). Finally, we enrich the data with all temporal and kinematic features available in the package (lines 22–23). Fig. 2 shows step-by-step what happened with the data after using some of the functionalities. Fig. 2(A) shows the outliers detected by the Hampel filters and Fig. 2(B) shows the trajectory with these data points removed.

[2] https://shorturl.at/bl259

```python
import pandas as pd
from ptrail.core.TrajectoryDF import PTRAILDataFrame
from ptrail.features.kinematic_features import KinematicFeatures
from ptrail.features.temporal_features import TemporalFeatures
from ptrail.preprocessing.interpolation import Interpolation
from ptrail.preprocessing.filters import Filters

df = pd.read_csv('https://raw.githubusercontent.com/YakshHaranwala/PTRAIL/main/examples/data/starkey.csv%27)
starkey = PTRAILDataFrame(data_set=df, latitude='lat', longitude='lon',
                          datetime='DateTime',traj_id='Id')

elk_traj = starkey.loc[starkey.index.get_level_values(const.TRAJECTORY_ID).isin(['930429E08'])]
elk_traj = PTRAILDataFrame(data_set=elk_traj.reset_index(), latitude='lat', longitude='lon',
                           datetime='DateTime', traj_id='Id')

elk_traj = KinematicFeatures.create_distance_column(elk_traj)
filt_elk = Filters.hampel_outlier_detection(dataframe=elk_traj, column_name='Distance')

inp_elk = Interpolation.interpolate_position(dataframe=filt_elk,
                                             sampling_rate=3600*2, ip_type='cubic')

enriched_elk = TemporalFeatures.generate_temporal_features(inp_elk)
enriched_elk = KinematicFeatures.generate_kinematic_features(enriched_elk)
```

Fig. 3. A code snippet for using PTRAIL with a single trajectory.

```python
from ptrail.features.kinematic_features import KinematicFeatures
from ptrail.preprocessing.filters import Filters
from ptrail.preprocessing.interpolation import Interpolation
from ptrail.preprocessing.statistics import Statistics

# Generate Kinematic Features first
feature_df = KinematicFeatures.generate_kinematic_features(dataframe=ships)

# Using Hampel Filter based on Distance and Speed.
dist_filter = Filters.hampel_outlier_detection(dataframe=feature_df, column_name='Distance')
speed_filter = Filters.hampel_outlier_detection(dataframe=dist_filter, column_name='Speed')

# Performing Cubic Interpolation with a sampling rate of 15 minutes.
cubic_ip = Interpolation.interpolate_position(dataframe=speed_filter, sampling_rate=15,
                                              ip_type='cubic' ,class_label_col='VesselTypeGroup')

# Finally, generating Kinematic stats, switching to segment based view and dropping NaN values.
stats_df = Statistics.generate_kinematic_stats(dataframe=cubic_ip, target_col_name='VesselTypeGroup')
pivoted_df = Statistics.pivot_stats_df(dataframe=stats_df, target_col_name='VesselTypeGroup')
pivoted_df.dropna(inplace=True)
```

Fig. 4. Data Preprocessing pipeline of ships' data using PTRAIL.

Fig. 2(C) shows a smoothened trajectory as the result of the cubic interpolation procedure. Finally, Fig. 2(D) shows the histogram of the bearing distribution of the elk's movement.

3.2. Ships dataset

Another example[3] of the use of PTRAIL is illustrated in Fig. 4 where we create a pipeline of pre-processing the data with PTRAIL and perform trajectory classification using the scikit-learn [18] library. This example starts by loading the ships dataset present in PTRAIL's GitHub repository into a PTRAILDataFrame. After, we perform a pipeline of operations to the dataset as seen in Fig. 4 to get the data ready for predicting the type of ship using scikit-learn [18]. First, we generate kinematic features on the ships dataset (line 7). Next, we remove outliers from the dataset using a Hampel filter based on the distance and speed of the ships between consecutive points (lines 10–11). Following outlier removal, we perform cubic interpolations to smoothen the trajectories and reduce jumps in the data points (lines 14–15). Next, we generate the following kinematic features' statistics: mean, median, standard deviation, minimum, maximum, 10th percentile, 25th percentile, 75th percentile, and 90th percentile. After, we pivot the DataFrame to contain one trajectory example per row and drop the NaN values to get it ready for the classification task (lines 18–20). Finally, we perform a stratified repeated k-fold cross-validation (5 folds, 10 repetitions) using the models seen in Fig. 5. We used F-score as a scoring metric since our dataset is imbalanced. The results visualized in Fig. 5 show that the ensemble methods (Random Forest, XGBoost, and ExtraTrees) have higher median f-scores, also indicating overall stability and the absence of high variability of the models.

4. Impact

PTRAIL provides a set of functionalities that can help researchers who need to handle trajectory data for extracting information about moving objects. Its ability to handle noise, absence of data and data enrichment with new features in parallel makes it a convenient library for handling large trajectory datasets. Cleaning raw trajectory data obtained from sources like Global Positioning Systems (GPS) and Automatic Identification Systems (AIS) devices is a tedious and time consuming task. Indeed, analysis tasks like anomaly detection, pattern mining and classification do not bring to the desired results when errors are present in the data and/or the data does not contain enough features to get interesting analysis results. Furthermore, the parallelization of PTRAIL's preprocessing task is particularly useful for large trajectories dataset and the ability for fast computation makes it a convenient and efficient choice for researchers. More specifically, by using PTRAIL, researchers could focus on answering higher level research questions and focus on developing analysis methods limiting the effort into the preprocessing steps. Additionally, we point out that PTRAIL is organized in a modular way so that it can encourage researchers to develop more sophisticated preprocessing and semantically enriching methods that can later integrated into the library.

[3] https://shorturl.at/fhOQW

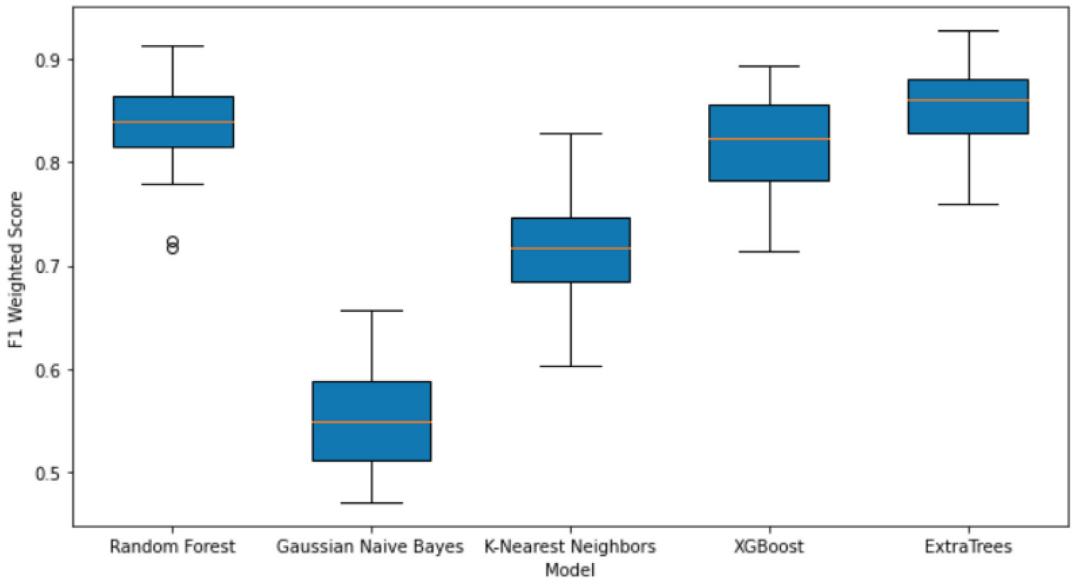

Fig. 5. A repeated cross-validation results performed on the data processed using PTRAIL.

Since the release of its beta version in August 2021, PTRAIL surpassed 9,000 downloads according to PePy [19] and thanks to the generality of the approach (we target raw trajectories that can be collected by GPS, AIS or other location devices) we expect the usage to be spread much more. Plus Python is widely used by the mobility data analysis community so this library can easily integrate the stack of analysis tools available in this domain.

5. Conclusions

PTRAIL is a state-of-the-art python package providing raw trajectories preprocessing tasks such as filtering, interpolation and feature engineering. This tool provides functions to transform raw data, as collected by location devices (therefore error prone and with noisy data) into clean and ready to use data set. This in turn enables the application of data mining and machine learning methods like clustering and classification. Furthermore, PTRAIL offers high efficiency in terms of computational speed combined with reliable and accurate results thanks to sophisticated, parallelized and vectorized calculations. It is also worth mentioning that PTRAIL can be suitably exploited by non expert users thanks to a large number of usage examples provided along with well documented code. PTRAIL follows high standards of coding and documentation which allows researchers to extend and enhance the library with newer functionalities. Finally, PTRAIL will help and reduce the researchers' burden of trajectory data preprocessing with a user friendly environment in Python for a forseeable future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank Memorial University of Newfounland for the Faculty of Science Undergraduate Research Award (SURA) given to Yaksh and Salman that was essential to the development of this library. This work was also partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant RGPIN-2022-03909.

References

[1] Pappalardo L, Simini F, Barlacchi G, Pellungrini R. Scikit-mobility: a python library for the analysis, generation and risk assessment of mobility data. 2019, http://arxiv.org/abs/1907.07062 [arXiv:1907.07062].
[2] Graser A. Movingpandas: Efficient structures for movement data in python. GI_Forum 2019;7(1):54–68. http://dx.doi.org/10.1553/giscience2019_01_s54, URL https://www.austriaca.at/rootcollection?arp=0x003aba2b.
[3] Sanches ADJAM. Uma arquitetura e implementação do módulo de pré-processamento para biblioteca pymove. 2019, Universidade Federal Do Ceará.
[4] Etemad M, Júnior AS, Matwin S. Predicting transportation modes of gps trajectories using feature engineering and noise removal. In: Canadian conference on artificial intelligence. Springer; 2018, p. 259–64.
[5] Júnior AS, Renso C, Matwin S. Analytic: An active learning system for trajectory classification. IEEE Comput Graph Appl 2017;37(5):28–39.
[6] Soares A, Rose J, Etemad M, Renso C, Matwin S. VISTA: A visual analytics platform for semantic annotation of trajectories. In: EDBT. 2019, p. 570–3.
[7] Abreu FH, Soares A, Paulovich FV, Matwin S. A trajectory scoring tool for local anomaly detection in maritime traffic using visual analytics. ISPRS Int J Geo-Inf 2021;10(6):412.
[8] Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature 2020;585(7825):357–62. http://dx.doi.org/10.1038/s41586-020-2649-2.
[9] pandas development team T. Pandas-dev/pandas: Pandas. 2020, http://dx.doi.org/10.5281/zenodo.3509134, Zenodo.
[10] Multiprocessing. 2021, https://docs.python.org/3/library/multiprocessing.html. [Accessed 25 August 2021].
[11] Jordahl K, den Bossche JV, Fleischmann M, Wasserman J, McBride J, Gerard J, et al. Geopandas/geopandas: v0.8.1. 2020, http://dx.doi.org/10.5281/zenodo.3946761, Zenodo.
[12] Gillies S, et al. Shapely: manipulation and analysis of geometric objects. 2007, toblerity.org, URL https://github.com/Toblerity/Shapely.
[13] Trofficus M. Hampel filter in python. 2021, PyPI, URL https://pypi.org/project/hampel/.
[14] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, et al. SciPy 1.0: Fundamental algorithms for scientific computing in python. Nature Methods 2020;17:261–72. http://dx.doi.org/10.1038/s41592-019-0686-2.
[15] Long JA. Kinematic interpolation of movement data. Int J Geogr Inf Sci 2016;30(5):854–68.
[16] Technitis G, Othman W, Safi K, Weibel R. From A to B, randomly: a point-to-point random trajectory generator for animal movement. Int J Geogr Inf Sci 2015;29(6):912–34.
[17] Wisdom MJ. The starkey project: a synthesis of long-term studies of elk and mule deer. 2005, Alliance Communications Group, URL https://www.fs.fed.us/pnw/starkey/.
[18] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.
[19] PTRAIL pepy tech link. 2022, https://pepy.tech/project/Ptrail. [Accessed 15 March 2022].