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, NL A1B 3X5, Canada S.J. Crew Building, EN-2021*
[b]*Institute of science and technology A. Faedo, National Research Council of Italy*
[c]*Universidade Federal de Santa Catarina (UFSC), Brazil*

Abstract

Trajectory data represent 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 limitation. 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.

Keywords: Trajectory Preprocessing, Feature Engineering, Parallel processing, Vectorization

Current code version

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v0.3Beta
C2	Permanent link to code/repository used for this code version	*github.com/YakshHaranwala/PTRAIL*
C3	Code Ocean compute capsule	None
C4	Legal Code License	BSD 3-Clause "New" or "Revised" License
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	python, jupyter
C7	Compilation requirements, operating environments & dependencies	numpy, pandas, geopandas, scikit-learn, hampel, scipy, psutil, folium, matplotlib, osmnx, shapely
C8	If available Link to developer documentation/manual	*ptrail.readthedocs.io/en/latest/index.html*
C9	Support email for questions	amilcarsj@mun.ca

Table 1: Code metadata (mandatory)

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 using a single machine 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 tasks performed over trajectory data.

Trajectory data preprocessing can be a tedious and time-consuming task. Our objective is to allow researchers to use 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], and anomaly detection [6]. However, the idea of parallezing 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 [7] 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 [8] DataFrame and inherits all its functionalities. The paralellization is done using python's built in multiprocessing module [9], 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 [10] and shapely [11] functionalities for extracting features from the data. We also used the implementation of Hampel filters available in [12], and some functionalities of scipy [13]. 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 semantic 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. *Semantic 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.

2.1. Software Architecture

The entire process of execution of a method is explained in Figure 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 paralell. 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 paralellizing the process execution per trajectory, showed huge improvements in processing time. The processing sequence showed in

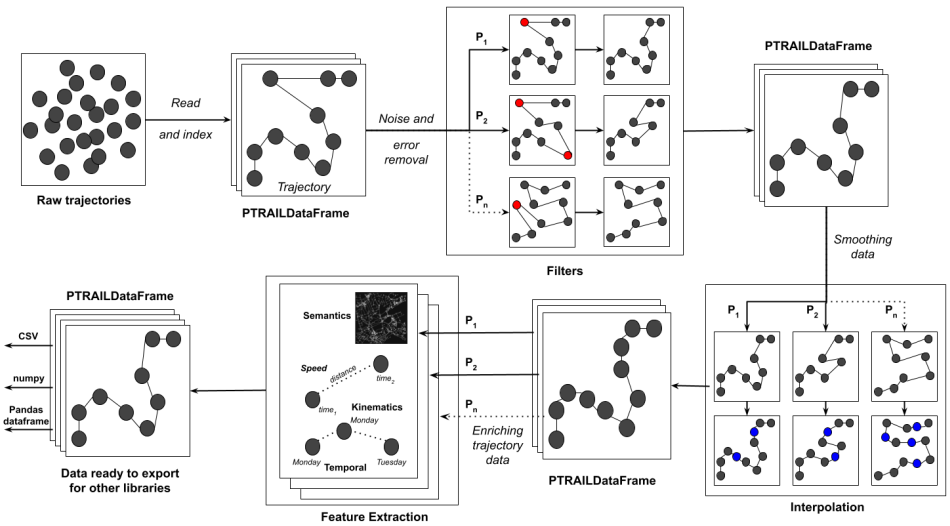

Figure 1: The PTRAIL pipeline starts by loading a PTRAIL dataframe. After, the user can apply filters for removing noise and/or smoothing the data with interpolation. Finally, features can be extracted from a higher quality data set. In the end, the data can be exported to common formats and be used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

Figure 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. The number of threads used when executing our library is a parameter that can be defined by the user. Indeed, 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

```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
# Read all trajectories from the dataset
df = pd.read_csv('./data/starkey.csv')
starkey = PTRAILDataFrame(data_set=df, latitude='lat', longitude='lon',
                          datetime='DateTime', traj_id='Id')
# Subselecting a trajectory from an elk
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')
# Applying the full pipeline of PTRAIL
elk_traj = KinematicFeatures.create_distance_between_consecutive_column(t1)
filt_elk = Filters.hampel_outlier_detection(dataframe=elk_traj,
                                            column_name='Distance_prev_to_curr')
inp_elk = Interpolation.interpolate_position(dataframe=filt_elk,
                                             time_jump=3600*2, ip_type='cubic')
enriched_elk = TemporalFeatures.generate_temporal_features(inp_elk)
enriched_elk = KinematicFeatures.generate_kinematic_features(enriched_elk)
```

Figure 2: A code snippet for using PTRAIL with a single trajectory.

data in such regions. We provide a paralellized version of the Linear and Cubic interpolation methods from the scipy package [13]. We also provide implementations of the Kinematic [14] and Random Walk [15] 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 travelled distance, speed, acceleration, bearing, bearing rate etc. Finally, if geographical layers about the study area are available, the user is able to create semantic 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 semantic features of PTRAIL.

3. Illustrative Examples

The PTRAIL gitHub repository currently contains numerous examples explaining the usage of PTRAIL. One of such examples with code snippets and outputs is illustrated in the Figure 3. Here we load a trajectory dataset

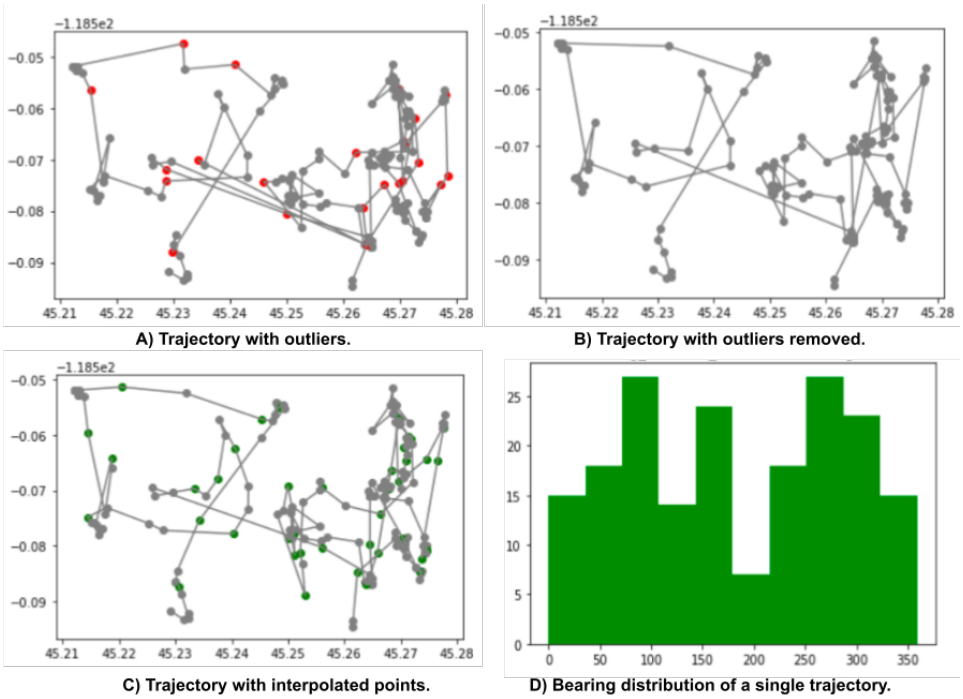

Figure 3: An illustrative example of the use of PTRAIL.

with data regarding movements of cattle, elk and deer from the Starkey project [16]. We load the data in lines 8-9 informing the latitute, 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. 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 17-18). 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 21-22). Figure 3 shows step-by-step what happened with the data after using some of the functionalities. Figure 3(A) shows the outliers detected by the Hampel filters and Figure 3(B) shows the trajectory with these data points removed. Figure 3(C) shows a smoothened trajectory as the result of the cubic interpolation procedure. Finally, Figure 3(D) shows the histogram of the bearing distribution of the elk's movement.

4. Impact

PTRAIL provides a set of functionalities that can help researchers who need to handle trajectory data for extracting information about moving ob-

jects. 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.

Since the release of its beta version in August 2021, PTRAIL surpassed 700 downloads according to PePy [17] 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.

6. Conflict of Interest

There is no conflict of interest with this publication and there has been no significant financial support for this work that has influenced its outcome.

Acknowledgements

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.

References

[1] L. Pappalardo, F. Simini, G. Barlacchi, R. Pellungrini, scikit-mobility: a python library for the analysis, generation and risk assessment of mobility data (2019). arXiv:1907.07062.

[2] A. Graser, Movingpandas: Efficient structures for movement data in python, GI_Forum 7 (1) (2019) 54–68. doi:$10.1553/giscience2019_01_s54$.
URL $https://www.austriaca.at/rootcollection?arp=0x003aba2b$

[3] A. D. J. A. M. Sanches, Uma arquitetura e implementação do módulo de pré-processamento para biblioteca pymove, Bachelor's thesis, Universidade Federal Do Ceará (2019).

[4] M. Etemad, A. S. Júnior, S. Matwin, Predicting transportation modes of gps trajectories using feature engineering and noise removal, in: Canadian conference on artificial intelligence, Springer, 2018, pp. 259–264.

[5] A. S. Júnior, C. Renso, S. Matwin, Analytic: An active learning system for trajectory classification, IEEE computer graphics and applications 37 (5) (2017) 28–39.

[6] F. H. Abreu, A. Soares, F. V. Paulovich, S. Matwin, A trajectory scoring tool for local anomaly detection in maritime traffic using visual analytics, ISPRS International Journal of Geo-Information 10 (6) (2021) 412.

[7] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant,

Array programming with NumPy, Nature 585 (7825) (2020) 357–362. doi:10.1038/s41586-020-2649-2.
URL https://doi.org/10.1038/s41586-020-2649-2

[8] T. pandas development team, pandas-dev/pandas: Pandas (Feb. 2020). doi:10.5281/zenodo.3509134.
URL https://doi.org/10.5281/zenodo.3509134

[9] Multiprocessing, https://docs.python.org/3/library/multiprocessing.html, accessed on August 25, 2021.

[10] K. Jordahl, J. V. den Bossche, M. Fleischmann, J. Wasserman, J. McBride, J. Gerard, J. Tratner, M. Perry, A. G. Badaracco, C. Farmer, G. A. Hjelle, A. D. Snow, M. Cochran, S. Gillies, L. Culbertson, M. Bartos, N. Eubank, maxalbert, A. Bilogur, S. Rey, C. Ren, D. Arribas-Bel, L. Wasser, L. J. Wolf, M. Journois, J. Wilson, A. Greenhall, C. Holdgraf, Filipe, F. Leblanc, geopandas/geopandas: v0.8.1 (Jul. 2020). doi:10.5281/zenodo.3946761.
URL https://doi.org/10.5281/zenodo.3946761

[11] S. Gillies, et al., Shapely: manipulation and analysis of geometric objects (2007–).
URL https://github.com/Toblerity/Shapely

[12] M. Trofficus, hampel filter in python (2021).
URL $https://pypi.org/project/hampel/$

[13] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.

[14] J. A. Long, Kinematic interpolation of movement data, International Journal of Geographical Information Science 30 (5) (2016) 854–868.

[15] G. Technitis, W. Othman, K. Safi, R. Weibel, From a to b, randomly: a point-to-point random trajectory generator for animal movement, International Journal of Geographical Information Science 29 (6) (2015) 912–934.

[16] M. J. Wisdom, The starkey project: a synthesis of long-term studies of elk and mule deer (2005).
URL https://www.fs.fed.us/pnw/starkey/

[17] Ptrail pepy tech link, https://pepy.tech/project/Ptrail, accessed on August 26, 2021.