Extracting Maritime Routes for Vessel Route Prediction

Daniela Duarte
Instituto Superior Técnico
Lisbon, Portugal
daniela.s.duarte@tecnico.ulisboa.pt

Manuel Lopes
Instituto Superior Técnico
Lisbon, Portugal
manuel.lopes@tecnico.ulisboa.pt

ABSTRACT

Vessels are often intermittently observed when at sea, either because the monitoring systems are shut down or the coverage link can't reach the sailed region. These gaps represent a severe threat in terms of safety at sea. The present work proposes to address this problem by predicting ships movement when there is no available information. The movement data of past behaviors is clustered into groups of similar movement patterns, from which is extracted a representative trajectory for each cluster. Ultimately, they will represent each main route and be used to identify the route membership of upcoming trajectories and predict future movements by querying the function at a given point. The evaluation is made with a real-world AIS dataset to demonstrate the viability of this approach.

Author Keywords

Automatic Identification System; Longest Common Subsequence; Trajectory Clustering; Movement Prediction.

INTRODUCTION

The maritime environment is a great source of economic growth, providing natural resources and access to trade and transport. This places the world's oceans and seas sustainability and security a high concern for many nations and international organizations.

The incorporation of the Automatic Identification System (AIS) technology on vessels, an automated on-board tracking device, has supported maritime surveillance authorities by broadcasting among AIS systems (e.g.: vessels, on-ground base stations) information stating who they are, where they are and their movement details. Although it was initially conceived for vessel collision avoidance, its use has been extended as means to achieve a deeper and broader knowledge of maritime situations since its mandatory deployment by the IMO on a significant range of vessels in 2004 [5]. However, it's not always possible to have a continuous observation of vessels at sea: the device may stop working or transmit wrong data due to some malfunction, the crew can turn off the device, and the 20 nautical miles depth of the transmission link may not cover part of the sailed area [12]. These coverage failures clearly represent high risks, such as collision situations, maritime pollution, piracy or unauthorized maritime arrivals [22].

The large amounts of available near-real time AIS data emerges as a valuable source of information for the creation of methods to automatically transform raw data into meaningful information and easily support operational decision makers. A broad area of trajectory data mining is devoted to addressing this issue, known as trajectory uncertainty [26]. A common approach is to find in the midst of a set of trajectories, one or more samples that follow the same (apparent) path as the uncertain one and are more complete, where the gaps can be filled with the information retrieved from the sequences of high extent. This concept is nothing but route prediction, as the movement of trajectories on the same route and observed as a whole serve as basis for the next positions of the (yet) partial ones.

In this context, the present work proposes to tackle the problem of long-term vessel position estimation, which comprises the process of predicting ship movements well beyond any available positioning data, based on the movement of past vessels on the same route [9]. More specifically, this covers trajectory pattern mining tasks and methods of Machine Learning (ML), namely clustering, to discover motion patterns – routes — from historical AIS data collections. From the clusters a compact representation is extracted to model each of the existing paths, ultimately used to forecast future positions.

RELATED WORK

This section presents the state-of-the-art research made to understand the two main topics of the scope of this thesis: trajectory learning and route prediction.

Trajectory Learning

As a group pattern mining task, the learning of trajectories seeks to capture the existing but unknown motion-patterns from trajectory data. The approaches are divided into the following categories [16]: distance-based; feature-based; and, model-based. When examining the literature, we join the feature-based into distance-based approaches, because they both directly compare trajectories when clustering.

Distance-Based Clustering

They comprehend the set of methods that groups trajectories by directly comparing how similar they are. Clustering algorithms are employed, as they are able to automatically infer the hidden structure of data and allocate the data into classes of similar objects [4]. In essence, the approach is reduced to a) choosing a clustering algorithm that determines how trajectories are gathered and b) choosing a similarity measure that establishes the candidates to be in the same group.

Similarity Measure The notion of (dis)similarity is given according to a measure quantifying how close the data points

are. Regarding trajectory data, the popular Euclidean Distance fails to work with unequal-length sequences, since it compares points of common indexes one by one with a straight line; even when there is a same number of points, the measure performs poorly if they are unaligned in time, due to different sampling rates/speeds. Measures allowing to pair up elements that are not in the same index in the data sequence have been introduced in the literature, and are classified into two broad groups [3]: shape-based distances, which only take into account spatial information (e.g., Hausdorff); and warping based distances, that integrate both the spatial and temporal dimensions (e.g., Dynamic Time Warping, Longest Common Subsequence).

Clustering Methods A myriad of clustering algorithms are available, but not all are suitable for trajectories. For example, K-means and Birch use Euclidean distance as metric, thus require fixed-dimensional vectors. Other algorithms offer more flexibility by allowing to represent the data with a matrix (as known as affinity matrix), where each of its elements are the similarities between samples. This is not only useful for using tailor made metrics, but to overcame the problem of not having a suitable matrix representation with unequal-length data. In this context, some of methods that are able to deal with trajectory data Affinity Propagation, DBSCAN and Hierarchical methods.

Aiming to reduce computation time and improve model efficiency, Sheng P. and Yin J. [24] use characteristic points from vessel trajectories having a wider change on SOG and COG and with the minimum description length (MDL). DBSCAN clustering is used for latitude, longitude data, speed and course, for which a tailor made measure is used for similarity measurement. Finally, the representative trajectory is calculated with the sweep line approach. Also using a compact representation of trajectories, Pallotta G. *et al.* [9] proposed the "TREAD", focused on clustering waypoints (e.g., entry, exit and stoppage). Thrdr are incrementally activated upon vessel traffic and clustered with DBSCAN based on the event type and scene features, originating routes when the vessels with matching start and end waypoints reach a certain threshold. Route objects are statistically described by the kinematic features of the vessels that originated them. This approach can easily tackle the problems of unequal-length trajectories, incomplete paths or with gaps. Another popular framework is called "TRACLUS" [14], that aims to find groups of similar sub-paths. The trajectories are partitioned into a series of meaningful parts through the Minimum Description Length principle, which are then grouped with a tailored density-based clustering algorithm and a metric based on the Hausdorff distance.

Model-Based Clustering

They enclose the clustering techniques where the data is characterized through a given model or as a mixture of probability distributions [16]. This category is further divided into statistical models and neural networks [16]. A common technique used in neural network approaches is Self-Organizing Maps (SOM), whose clustering process is based on a projection to a lower-dimensional space. The authors in [23] apply it to cluster simulated vessel data and use speed and heading for that purpose. SOM is not able to work with variable-length time-series [1]. On the other hand, Gaffney S. and Smyth P. [8] presented a probabilistic model based clustering by representing trajectory data with a mixture of regression models, where the Expected-Maximization (EM) is used to estimate the cluster's parameters. Aiming to simplify the choice of an appropriate model to fit the data, Zhong S. and Ghosh J. [27] proposed a unified framework for model-based clustering.

Route Prediction

The approaches regarding this topic can be categorized into three types [25]: physical models, based on physical laws that consider the various aspects affecting the motion (e.g.: force, mass); learning model based methods, which consider that the aspects that physical models include can be estimated automatically by creating movement models of past behaviours; and hybrid methods, that either combine the hybrid and learning models or use multiple learning methods.

Physical Models are useful for simulation systems, but require strict environmental and state conditions that are hard to obtain in practice. Kalman Filters are a widely used technique in this category [10]. In [15], Long-Short Term Memory Networks (LSTMs) were used as a learning-based approach for path modelling, after conducting a distance-based clustering. The LSTMs are applied to each unravelled route, where an iterative forecasting process is made to get the future values of both position, speed and course, given the current ones. further applies BI-LSTMS and shows better results. Also through a learning-based approach, Giuliana *et al.* [9] use the Bayes Rule is used to get the posterior probability of following one of the existing routes, given the current position and velocity, from which the future positions of the route with highest probability are extracted with kinematic equations. Perera L. *et al.* [20] exploit a hybrid approach with Extended Kalman Filters (EKF), a variation able to handle nonlinearities, to estimate position, velocity and acceleration. The method is incorporated in a Curvilinear Motion Model, that is used to describe the movement.

DATA ANALYSIS AND CLUSTERING

Data Sample

The dataset used in this project is composed by a real-world AIS dataset[1] from the region of Brittany, in France. It covers a period of 6 months, from 01-10-2015 to 31-03-2016. The following subset of attributes was used: MMSI, longitude, latitude, speed over ground, course over ground, time and ship type.

Regarding the vessel's trajectories, it was assumed that a new trip was made when the delta time between 2 broadcasted messages by a given vessel was greater than 24 hours or when the vessel's position was less than 5 km from a port. All records with duplicated values for both longitude, latitude, and date from a given vessel were removed, as well as all vessels with the ship type field not filled. Unfeasible speed or infeasible position messages were removed from the set.

[1] http://datacron-project.eu/

Tracks whose speed is lower than 0.1 knots for more than 80% of the time (at anchor or moored vessels) were also removed. All trips were required to have at least 3 data points. After the data cleaning, the data sample contains a total of 1 461 000 records, 2 045 vessels, and 5 157 trips.

In this thesis, we're interested in studying long and repetitive movements that characterize the main global routes, thus we will only work with the cargo and tanker AIS messages. This subset contains a total of 3187 trajectories. It makes sense to always include trajectories of a greater extent since we're interested in finding part of main global routes. On the other hand, most of the trajectories have less than 200 km sailed, which is a small distance for a $\approx 500 \times 800$ km area regarding long-term paths, so we excluded those with less 100 km because they were not a meaningful portion of the data. To sum up, the data sample was further divided into three sets according to the sailed distance: the 1^{st} contains trajectories with more than 100 km (total of 1381 trajectories); the 2^{nd} contains trajectories with more than 200 km (total of 754 trajectories); and the 3^{rd} contains trajectories with more than 300 km (total of 183 trajectories).

Data Treatment

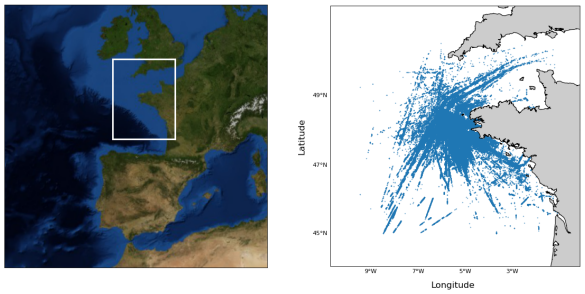
For the purposes of forecasting movement in regular time-intervals, it is required for trajectories to have observations of fixed and uniform frequency. Moreover, the gaps between two consecutive sampling points can be wide enough to fail in capturing the trajectory movement, which may introduce errors when measuring distances. The dataset contains irregularly sampled messages, so a pre-processing step is thus needed, namely resampling. A natural way of resampling data is to fit the data with a mathematical function that intersects the set of discrete points, and where new new points are easily retrieved [7]. Splines were used in this work, more specifically linear splines. Trajectory data can have multiple dimensions, so each individual dimension's sequence was fitted separately, and the resulting vectors joined in the end. It is required that both dimensions have the same independent variable, which in this work was time. Under the interpolation module from the Python-based SciPy library[2], the method *UnivariateSpline* was used to create linear spline representations, and evaluate new points with a step size of 300 (5-minute interval) regarding the latitude, longitude and speed dimensions. Finally, the heading and speed was then manually re-calculated, given the new set of points.

Clustering Method

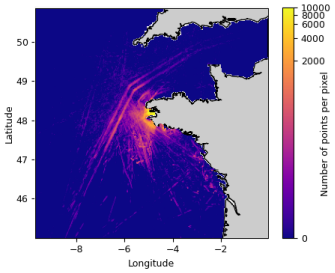
Choosing a clustering method is highly dependent on the data's characteristics, so the technique must satisfy the following requirements: no size uniformization needs to be made; the clusters can have different densities; the clusters can have an irregular shape. On this note, from the methods that are able to work with trajectories, three were chosen: Agglomerative Clustering, DBSCAN, and OPTICS. Under the Python's machine learning framework scikit-learn[3], which provides implementations of all methods in the clustering module, the algorithms were further compared.

[2]https://www.scipy.org/
[3]https://scikit-learn.org/

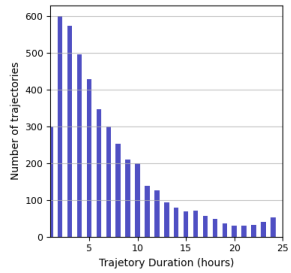
Figure 1. Dataset visualization and statistics.



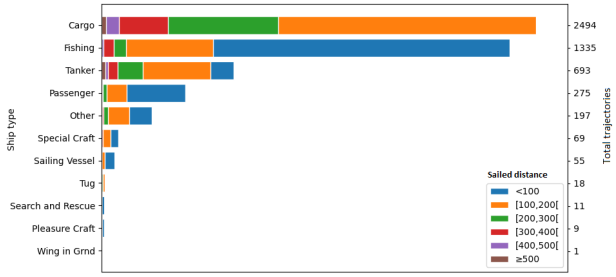
(a) $\approx 500 \times 800$ km area. **(b)** AIS positions in the covered area.



(c) Positions heatmap.



(d) Trajectory duration distribution.



(e) Trajectories length distribution per ship type.

From each of the obtained clusters, a representative trajectory was retrieved that provides the underlying baseline movement of the cluster and materializes one of the main routes. In technical terms, this trajectory is the cluster centroid and in the present work is the one with the lowest sum of distances to all its respective cluster members. At a later point, a re-classification was made to evaluate the routes assignment.

Similarity Measure

The first step towards applying any clustering method is the definition of the similarity measure. To accurately measure the distance between the trajectories at hand, the method must be able to assess the affinity between the pair of trajectories even if: the number of points differs; the sampling are different; the starting and ending positions are not the same. Given that there is no agreement on what method is the best, we tested and compared the two most used measures in the literature, namely Dynamic Time Warping and the Longest Common Subsequence, which are both able to work with the above points.

Longest Common Subsequence Let $dist(x_n, y_n)$ represent the function measuring the distance between two data points, the Longest Common Subsequence $LCSS(X, Y)$ between two trajectories X and Y is recursively defined as follows [13]:

$$\begin{cases} 0, & \text{if } X \text{or } Y \text{ is empty} \\ 1 + LCSS_{\delta,\epsilon}(Rest(X), Rest(Y)), & \text{if } dist(x_i, y_j) \leq \epsilon \\ & \text{and } |j - i| \leq \delta \\ \max \begin{cases} LCSS_{\delta,\epsilon}(Rest(X), Y) \\ LCSS_{\delta,\epsilon}(X, Rest(Y)) \end{cases} & \text{otherwise} \end{cases} \quad (1)$$

To retrieve a meaningful dissimilarity value from the length of the longest common subsequence, the percentage of that value regarding the length of the shortest trajectory is returned. Let min_length represent the number of points of the shortest trajectory, the distance $lcss\_dist(X, Y)$ between two trajectories X and Y is therefore obtained by:

$$lcss\_dist(X, Y) = \frac{LCSS(X, Y)}{min\_length(X, Y)} \times 100 \quad (2)$$

Position Data Initially, LCSS was tested with positional information (longitude, latitude). Since we are dealing with points on the surface of a sphere, the haversine formula was adopted, which determines the distance between geographic coordinates on the globe. Let R represent the average radius of the earth, the distance between points p_1 and p_2 along its two dimensions (longitude and latitude) is as follows:

$$\mathrm{d}(\mathrm{p}_1, p_2) = 2Rarcsin(\sqrt{sin^2(\tfrac{p_2^1 - p_1^1}{2}) + cos(p_1^1)cos(p_2^1)sin^2(\tfrac{p_2^2 - p_2^2}{2})}) \quad (3)$$

The implementation of LCSS from the traj-dist library[4] developed in Python was used with the spherical option, which employs the haversine distance between 2D-coordinates. Several tests were made to choose the best distance threshold that can only be evaluated by visually checking the values for each pair of trajectories. Based on data visualizations of the routes, the value was based on the average width of the routes, and it was chosen a value of 19 km. The results showed that working only with this type of data does not allow us to identify trajectories in different directions.

Angle Data For the directional component, expressed by the COG attribute, the distance was measured by the cosine similarity, which is a widely used measure for assessing the affinity between angles. The cosine distance between two angles a_1 and a_2 is thus represented as:

$$cosine\_distance(a_1, a_2) = 1 - cosine(a_1 - a_2) \quad (4)$$

The code from Python's traj-dist library regarding LCSS was adapted to incorporate the cosine distance instead. Regarding the distance threshold, a value of 10^o was set for both datasets, estimated once more by visually checking different values until they were acceptable according to the trajectories at hand. In this case, only angle data is not able to determine that trajectories are far apart.

[4]https://github.com/bguillouet/traj-dist

LCSS Joint Distance Measure A natural conception of the final measure is the average values of both position and angle. Taking the example where we have two trajectories close together (e.g., position distance to 0%) but with opposite directions (e.g., angle distance to 100%), the resulting distance is 50%, which is too optimistic for such difference. So, for the cases where one of these values is too low, this should be a straight indication that the trajectories differ greatly. For this reason, it was assumed that when either the distance with position or direction is greater than 60% then that was the value returned by the joint measure, while the remaining cases took the average. The pseudo-code is as follows:

Algorithm .1: LCSS Joint distance

Input: Trajectory1, Trajectory2
Output: Percentage representing how much the trajectories match
begin
  $position\_sim \longleftarrow lcss(trajectory1.positions, trajectory2.positions)$
  $direction\_sim \longleftarrow lcss\_cosine(trajectory1.angles, trajectory2.angles)$
  $max\_value \longleftarrow max(position\_sim, direction\_sim)$
  if $max\_value > 60$ **then**
    return max_value
  else
    return$(position\_sim + direction\_sim)/2$

Dynamic Time Warping

Let $dist(x_n, y_n)$ the function measuring the distance between two data points, the Dynamic Time Warping DTW(X,Y) between two trajectories X and Y is defined as follows [17]:

$$\begin{cases} 0, & \text{if } X \text{ and } Y \\ & \text{is empty;} \\ \infty, & \text{if } X \text{ or } Y \\ & \text{is empty;} \\ dist(x_1, y_1) + min \begin{cases} DTW(Rest(X), Rest(Y)) \\ DTW(Rest(X), Y) \\ DTW(X, Rest(Y)) \end{cases}, & \text{otherwise.} \end{cases} \quad (5)$$

DTW requires all points from two trajectories to be matched. Hence when measuring two trajectories with different lengths, an overflow is introduced from the remaining points of the longest sequence, and the measure would perform poorly. That being so, a subsequence from the longest trajectory corresponding to the nearest points of the beginning and end of the shorter trajectory was used.

DTW Joint Distance Measure Due to the same issues as with LCSS, both position and direction data were used in an averaged joint distance. The implementation of DTW from tslearn[5] (function *dtw_from_metric*) was used once more using the haversine distance, for positions, and the cosine distance, for directions. The pseudo-code is as follows:

[5]https://tslearn.readthedocs.io/

Algorithm .2: DTW Joint distance

Input: Trajectory1, Trajectory2
Output: Distance in km between trajectories
begin

$shortest\_traj, longest\_traj \longleftarrow find\_lengths(trajectory1, trajectory2)$
$longest\_subsequence \longleftarrow get\_longest\_subsequence(shortest\_traj, longest\_traj)$

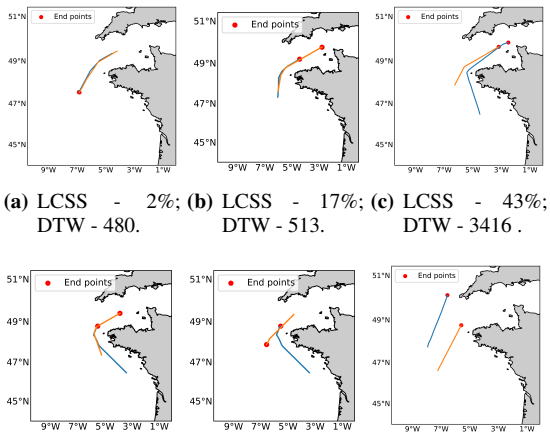
$position\_sim \longleftarrow dtw\_from\_metric(shortest\_traj.positions, longest\_subsequence.positions, haversine)$
$direction\_sim \longleftarrow dtw\_from\_metric(shortest\_traj.angles, longest\_subsequence.angles, cosine\_distance)$

return $(position\_sim + direction\_sim)/2$

Results

Comparing the results of LCSS and DTW for the same subset of samples, it is possible to conclude that both measures were able to identify when the routes matched entirely (or almost) (e.g., Figures 3(a) to 3(b)), and when they were far apart (e.g., Figure 3(f)). On the one hand, Dynamic Time Warping did not give fair values when the routes were close but with different shape or direction, as seen in Figure 3(e), since it attributed higher to partially identical trajectories, in Figure 3(c). This can be explained by the fact that DTW simply calculates the distances between aligned points. So, when trajectories are too close, this makes it enough to dictate that the routes might be similar. In spite of the fact that using a subset from the longest trajectory helps to tackle the sum excesses from unequal-length sequences, this distance may become even shorter because there are even less points to add. On the other hand, Longest Common Subsequence provided reasonable values for all examples, and they are easier to understand than those from DTW. Furthermore, this allows to attribute automatically higher values for opposite courses, which DTW did not. In conclusion, the Longest Common Subsequence had the best results.

Figure 2. Representative sample of joint distance values.



(a) LCSS - 2%; DTW - 480. **(b)** LCSS - 17%; DTW - 513. **(c)** LCSS - 43%; DTW - 3416 .

(d) LCSS - 67%; DTW - 5073 . **(e)** LCSS - 100%; DTW - 3061 . **(f)** LCSS - 100%; DTW - 10810.

Agglomerative Clustering

Hierarchical Agglomerative Clustering (HAC) falls into the hierarchical methods category, which builds a tree, also known as a dendrogram, where each node contains a set of samples that are merged or split repeatedly [4]. The root node contains all the samples, and the leaf nodes group each sample individually. In a bottom-up approach, as in HAC, it starts with all samples in individual clusters, merging nodes that minimize one of the following linkage criteria: the mean of the distances between each pair of observations from both nodes (average); the maximum distance between the observations of both sets (complete); the minimum of the distances between observations of both sets (single).

It was used the function *AgglomerativeClustering* from scikit-learn. The linkage criteria needs to be set, and, when the number of clusters is unknown, a distance threshold is required to establish up until when clusters are merged.

Model Selection

A standard evaluation measure for finding the best linkage criteria is the CPCC [18]. The cophenetic distance is equivalent to the minimum distances between any of the samples that are to be included in the same node. A matrix is created with this distance and is further correlated with the data's distance matrix to quantify how accurately the pairwise distances were preserved. The common approach to find an ideal distance threshold is to analyze the resulting dendrogram for a given link and look for groups that combine at a higher dendrogram level.

DBSCAN

This technique incorporates the notion of density, where clusters are high-density areas followed by low-density areas [6]. These higher density areas are obtained by retrieving the data points that have at least k neighbors in an ε radius — core points —, each one of them formalizing a cluster. These samples are further grouped with the neighbors that are core as well — density reachable core points —, along with their neighborhood. This verification is made for every sample in the ε-neighborhood of every core sample, allowing to create a network of density connected core points. A cluster emerges as a set of core points and the points that are close to one of the cores but are not themselves core — non-core points. The low-density areas — noise/outliers — are then the non-core samples that are not close to any core [19]. The function *DBSCAN* from scikit-learn was used, and it requires the minimum number of neighbors and the maximum distance between samples to be set.

Model Selection

For a given k, the respective ε can be automatically extracted by taking the distance from the k^{th}-nearest neighbors of every point, sort them increasingly, and plot the points in the so-called k-dist plot [18]. The ideal ε is evaluated at the point where the distance has a sharper rise. There is no technique for finding the k, but values between 3 and 24 have been used with most datasets in the literature, for density-based methods [21]. In this work, values in [3,6,9] were tested.

OPTICS

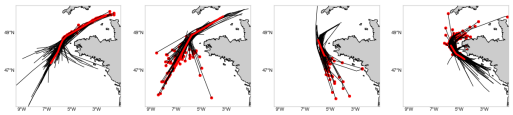
OPTICS [2] closely follows DBSCAN but relaxes the ε and takes it as a value range ($0 \leq \varepsilon_i \leq \varepsilon$), exploring clustering with multiple ε_i to find different densities. In practice, OPTICS requires only the minimum number of neighbors; the parameter ε can be fixed to its maximum value to find clusters across all scales [19]. The function *OPTICS* was used from scikit-learn's clustering module, and the same values for ε in DBSCAN were tested. The default setting of ε to ∞ was adopted, to explore all possible ranges.

Results

Using 70% of each one of the data samples, the data was clustered with the Longest Common Subsequence, because it provided better results and allows to easily interpret the distances. The representative trajectories are shown in red, together with the trajectories' end point. For each set, we illustrate the best results from each clustering method, and enumerate the following remarks:

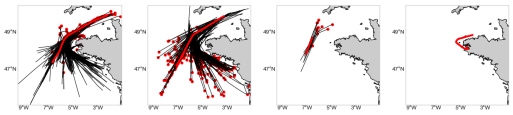
- HAC was the only clustering method to obtain reasonable clusters for the 1^{st} set, even though many of the clusters had misplaced trajectories. It could unravel groups of movements from the two big clusters that DBSCAN and OPTICS extracted. This was due to the cutoff to nodes that exceed the distance defined by the linkage criteria.
- For set 2, while the number of trajectories was reduced, working with longer trajectories proved to achieve better results with OPTICS; HAC had few clusters, but their quality was improved in terms of finding local behaviors from the clusters for set 1, and providing paths with more information; the parameters yielded by the model selection of DBSCAN did not provide reasonable clusters, grouping most of the data into two big clusters (upwards and downwards movements).
- A decrease in the number of clusters was seen in all clustering algorithms for the 3^{rd}, with the exception of DBSCAN that had the same and rendered useless clusters again. Indeed this set is much smaller than the previous ones; nonetheless the quality of the clusters for HAC and OPTICS was better than of set 1, and a good compromise between the data size and the clusters representation was achieved.
- Regarding the influence of route lengths, it was concluded that the 1^{st} set had many small trajectories that provided an easy gateway to unwanted merges between clusters. The 2^{nd} set delivered more clusters with reasonable representation, although the best compactness was achieved in the 3^{rd} set.

Figure 3. 4 out of 65 HAC Clusters with average link and distance threshold=70, for set 1.



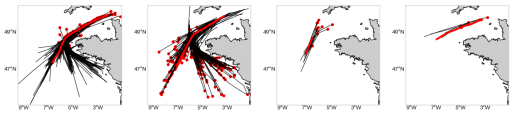
(a) Cluster 1 **(b)** Cluster 2 **(c)** Cluster 3 **(d)** Cluster 4

Figure 4. 4 out of 8 DBSCAN clusters with $k = 3$ and ε=13, for set 1.



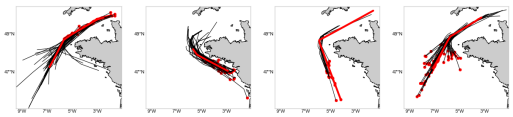
(a) Cluster 1 **(b)** Cluster 2 **(c)** Cluster 3 **(d)** Cluster 4

Figure 5. 4 out of 15 OPTICS cluster with $k = 3$, for set 1.



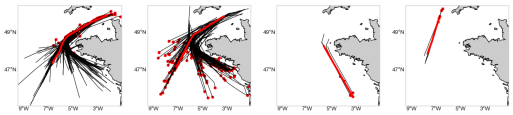
(a) Cluster 1 **(b)** Cluster 2 **(c)** Cluster 3 **(d)** Cluster 4

Figure 6. 4 out of 39 HAC clusters with average link and distance threshold = 70, for set 2.



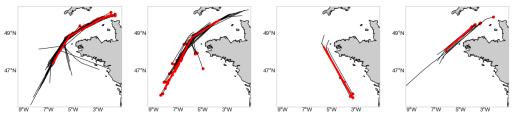
(a) Cluster 1 **(b)** Cluster 2 **(c)** Cluster 3 **(d)** Cluster 4

Figure 7. 4 out of 5 DBSCAN clusters with $k = 3$ and ε=22, for set 2.



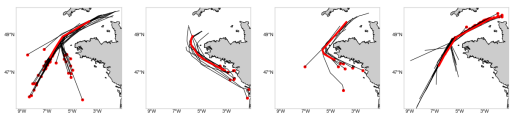
(a) Cluster 1 **(b)** Cluster 2 **(c)** Cluster 3 **(d)** Cluster 4

Figure 8. 4 out of 22 OPTICS clusters with $k = 3$, for set 2.



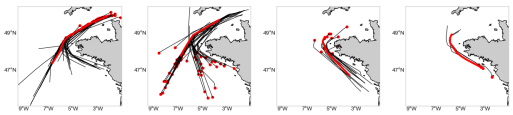
(a) Cluster 1 **(b)** Cluster 2 **(c)** Cluster 3 **(d)** Cluster 4

Figure 9. 4 out of 22 HAC clusters with average link and distance threshold=70, for set 3.



(a) Cluster 1 **(b)** Cluster 2 **(c)** Cluster 3 **(d)** Cluster 4

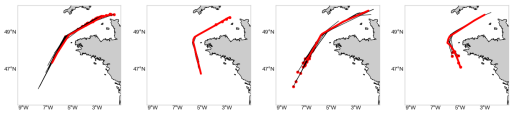
Figure 10. 4 out of 5 DBSCAN clusters with $k = 3$ and ε=30, for set 3.



(a) Cluster 1 **(b)** Cluster 2 **(c)** Cluster 3 **(d)** Cluster 4

PREDICTION

Figure 11. 4 out of 15 OPTICS clusters with $k = 3$, for set 3.



(a) Cluster 1 **(b)** Cluster 2 **(c)** Cluster 3 **(d)** Cluster 4

The route prediction was approached through a probabilistic measure over the nearest routes' positions. In short, the estimation for a given Δt is the weighted sum of the positions with respect to t of the three most probable paths. The softmax function was used to acquire the weights distribution for each route.

Algorithm Description

For an incoming trajectory, a radius is centered in the observation's end point to trace the area where the nearest routes should be. If no routes are inside that radius, the route is considered as an "outlier". Likewise, if none of the routes found have a similar direction. The same distance threshold used in LCSS was set to represent the radius (19 km). Given the intersecting routes, it is calculated the hierarchy of the three closest ones to feed the softmax function. The LCSS joint distance fails in computing this hierarchy, so the DTW was used. Finally, the nearest points between the current observations' end point and the closest routes are calculated, aiming to define the location from where the prediction will start in each route. In order for the extracted movement to be made with a same speed as the incoming trajectory, the same approach used for resampling was employed to interpolate the data points. To accomplish this, it is calculated the average speed of the incoming trajectory for the last 6 points (equivalent to 30 minutes), assuming that it provides a good estimate of future speed. With both the speed and the five-minute interval between each data point, it is extracted the average distance sailed per point, which is used as the new resampling interval. The independent variable was the accumulative sailed distance. An overview of the algorithm is shown in fig. 12 .

Softmax

Softmax is a commonly used function to express the probability distribution of a discrete variable with n different values (classes) [11]. In the present work, it was chosen due to its simplicity and usefulness to convert the vector of distance scores to a vector of probabilities, that serves as the mixture of weights for the path prediction. The procedure is shown in Figure 13.

Results

The clusters that did not render reasonable clusters were removed from the evaluation. Thus, the prediction was performed with the following settings: for the 1^{st}, only HAC was used; for the 2^{nd} and 3^{rd} set, both HAC and OPTICS (with $k = 3$) were used. The remaining 30% of each data sample was used as the testing set. Each trip was divided into two sets in order to simulate a moving vessel: one representing the current observation and the other serving as the ground truth of the future movement, to which the system's

Figure 12. Iterative route prediction.

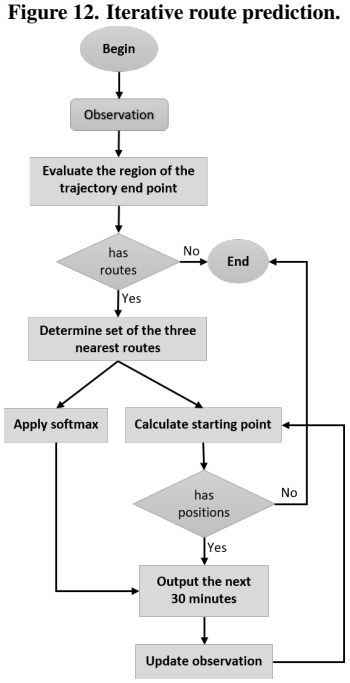
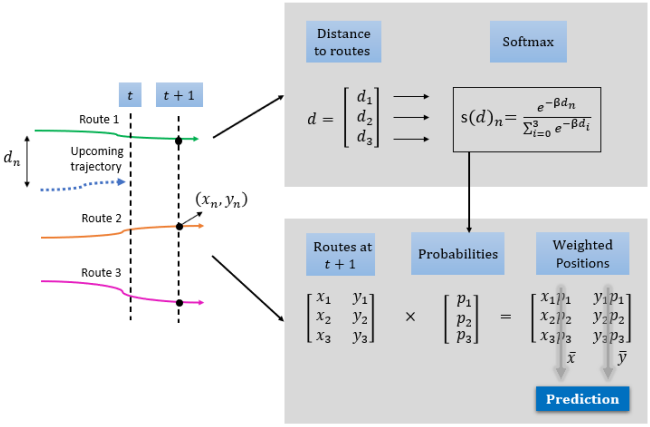


Figure 13. Route prediction with softmax for $t + 1$.



estimate will be compared. This split was done according to 25%, 50% and 75% of the beginning of the trajectory, and it corresponds to the current observation. The average lateral deviation per km is used to evaluate the prediction accuracy. Let L be the prediction distance in km and N the number of predicted points, for the set of points p_i and its estimates p_i' the error,in km, is defined as follows:

$$error = \frac{1}{L}\sum_{i=1}^{N} haversine(p_i^{'}, p_i)| \tag{6}$$

A parametric test was performed to evaluate the effect of the β parameter, in softmax's transformation, on the resulting prediction. This allows to select the best setting for β, for which the considerations regarding each observation size were made.

Set 1

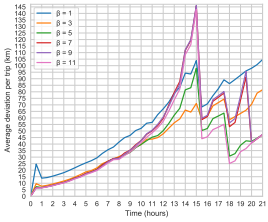
The evaluation for HAC clusters is exhibited in Figure 14 and Table 1. The results showed the following:

- The number of outliers increases directly with the amount of observation that is being used, meaning that while we get more positional information, the main routes do not have for more information to keep up with predictions. In fact, the representative paths was small (see Figure 3), so the area is not well represented. This also explains why the performance did not necessarily improve with increasing observations.
- There was no overall agreement on what is the best setting for β, in order to achieve lower average deviation per trip.
- Bearing in mind that 19 km were established for the width of a maritime corridor, then if a vessel has approximately 10 km of lateral deviations, it would still belong to that corridor. This was achieved for short times, in any of the observation sizes.
- The predictions' average deviations were not stable, and, looking back at the trajectories duration, in Figure 2(d), we conclude that more than a half has less than 15 hours, so, the error might be made according to fewer trajectories along the way and the results can suffer variances. Yet, this demonstrates that the deviation could keep up with an increasing distance for those trajectories.

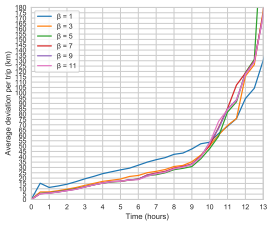
Table 1. Outliers for HAC clusters, for set 1.

Observation	Outliers
25%	36%
50%	41%
75%	52%

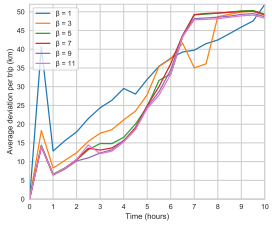
Figure 14. Comparative analysis of different beta values for each observation size – HAC clusters, on set 1.



(a) 25% of observation.



(b) 50% of observation.



(c) 75% of observation.

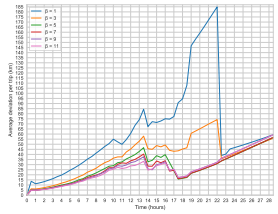
Set 2

The prediction evaluation made with the representative routes from HAC clusters is demonstrated in Figure 15 and Table 2. It was possible to verify that: the number of outliers revealed again a lack of movement information; the higher the β the lower the average deviations, showing that following the only the closest route leads to lower deviations; the mean deviations were kept below 10 km for a longer amount of time (e.g., for a 75% observation, it was maintained for most of the prediction).

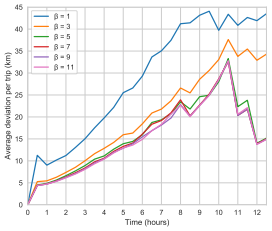
Table 2. Outliers for HAC clusters, for set 2.

Observation	Outliers
25%	22%
50%	34%
75%	50%

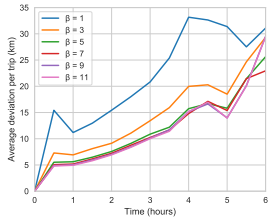
Figure 15. Comparative analysis of different beta values for each observation size – HAC clusters, on set 2.



(a) 25% of observation.



(b) 50% of observation.



(c) 75% of observation

In relation to OPTICS main routes, the results showed, once more, that the number of outliers increased for bigger sizes of observation, but less than HAC. Effectively, the prediction window was extended up to 19 hours, while HAC provided 12 hours, for observations of 50%. Still, for that extra time window, there was high average deviations per trip. The outperforming β did not reach a general agreement, but with β= 3 in a 75% observation, the prediction achieved low average deviation for all the time window.

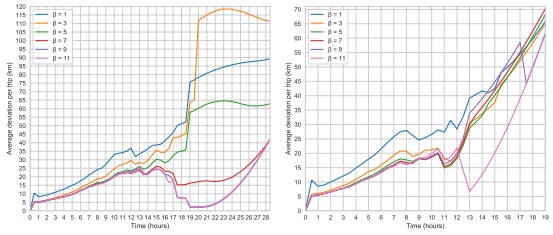
Table 3. Outliers for OPTICS clusters, for set 2.

Observation	Outliers
25%	25%
50%	32%
75%	46%

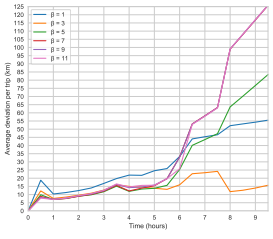
Set 3

Similarly as in set 2, HAC had the lowest errors with β=11, providing low deviations throughout all the prediction, for a 75% observation.

Figure 16. Comparative analysis of different beta values for each observation size – OPTICS clusters, on set 2.



(a) 25% of observation. **(b)** 50% of observation.

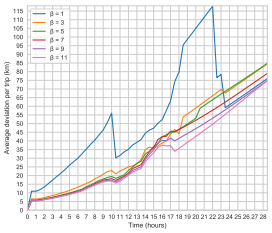


(c) 75% of observation.

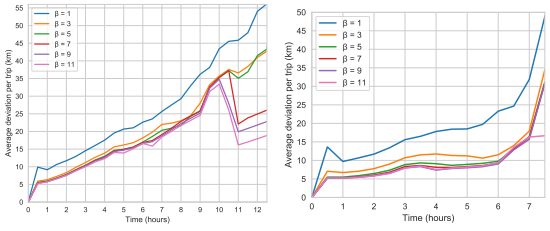
Table 4. Outliers for HAC clusters, for set 3.

Observation	Outliers
25%	23%
50%	37%
75%	46%

Figure 17. Comparative analysis of different beta values for each observation size – HAC clusters, on set 3.



(a) 25% of observation – average deviation of km.



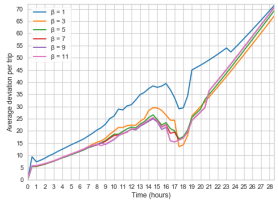
(b) 50% of observation. **(c)** 75% of observation.

OPTICS also found β=11 to be the best setting, and had lower deviations errors than HAC with a 25% observation, but equivalent performance, for a 75% observation. For a 50%, it was able to provide a bigger prediction window.

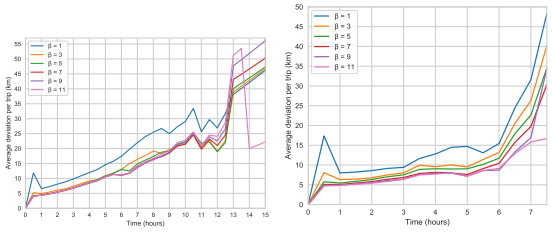
Table 5. Outliers for OPTICS clusters, for set 3.

Observation	Outliers
25%	21%
50%	28%
75%	42%

Figure 18. Comparative analysis of different beta values for each observation size – OPTICS clusters, on set 3.

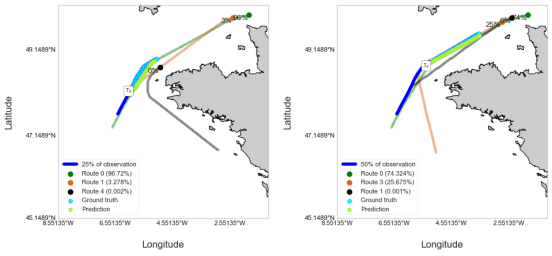


(a) 25% of observation.

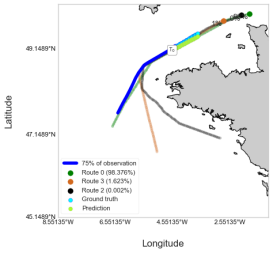


(b) 50% of observation. **(c)** 75% of observation.

Figure 19. Example of movement prediction with OPTICS clusters, for set 3.



(a) 25% of observation – average deviation of 5 km. **(b)** 50% of observation – average deviation of 2 km.



(c) 75% of observation – average deviation of 3 km.

CONCLUSION AND FUTURE WORK

This thesis focused in providing a study for the problem of route prediction, exploring three different clustering algorithms (Hierarchical Clustering, DBSCAN and OPTICS), along with two similarity measures (Dynamic Time Warp-

ing and the Longest Common Subsquence) to extract maritime routes. The tests showed that LCSS was the outperforming measure, providing a more comprehensive view on the distance values. DBSCAN did no render useful clusters. The prediction based on the weighted movement of the three nearest routes showed that the lowest average deviations are achieved using mainly the information from the nearest route, and as more information is received the more accurate the estimates can be.

Possible future work includes enriching the clustering with others types of information (e.g.; origin and destination), and compare the prediction approach to other methods.

REFERENCES

1. Aghabozorgi, S., et al. Time-series clustering - A decade review. *Information Systems 53* (2015), 16–38.
2. Ankerst, M., Breunig, M. M., Kriegel, H.-p., and Sander, J. Optics: Ordering points to identify the clustering structure. *ACM SIGMOD Record 28*, 2 (1999), 49–60.
3. Besse, P., Guillouet, B., Loubes, J.-M., and François, R. Review and Perspective for Distance Based Trajectory Clustering. 1–10.
4. Bian, J., Tian, D., Tang, Y., and Tao, D. A survey on trajectory clustering analysis.
5. Cazzanti, L., Millefiori, L. M., and Arcieri, G. A document-based data model for large scale computational maritime situational awareness. *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015* (2015), 1350–1356.
6. Daszykowski, M., and Walczak, B. Density-Based Clustering Methods. *Comprehensive Chemometrics 2* (2009), 635–654.
7. Dierckx, P. *Curve and Surface Fitting with Splines*. Oxford University Press, 1993.
8. Gaffney, S., and Smyth, P. Trajectory clustering with mixtures of regression models. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM Press (1999), 63–72.
9. Giuliana, P., Michele, V., and Karna, B. Vessel pattern knowledge discovery from AIS data: A framework for anomaly detection and route prediction. *Entropy 15*, 6 (2013), 2218–2245.
10. Huang, Y., Chen, L., Chen, P., Negenborn, R. R., and van Gelder, P. H. Ship collision avoidance methods: State-of-the-art. *Safety Science 121*, March (2020), 451–473.
11. I. Goodfellow, Y. B., and Courville, A. Deep-learning. *MIT Press* (2016).
12. Kerbiriou, R., Rajabi, A., and Serry, A. The Automatic Identification System (AIS) as a Data Source for Studying Maritime Traffic.
13. Kollios, G., Vlachos, M., and Gunopulos, D. Discovering Similar Multidimensional Trajectories. *Encyclopedia of GIS* (2016), 1–8.
14. Lee, J.-g., and Han, J. Trajectory Clustering: A Partition-and-Group Framework.
15. Li, W., Zhang, C., Ma, J., and Jia, C. Long-term vessel motion predication by modeling trajectory patterns with AIS data. *ICTIS 2019 - 5th International Conference on Transportation Information and Safety* (2019), 1389–1394.
16. Liao, T. W. Clustering of time series data — a survey. 1857–1874.
17. Magdy, N., Sakr, M. A., Mostafa, T., and El-Bahnasy, K. Review on trajectory similarity measures. *2015 IEEE 7th International Conference on Intelligent Computing and Information Systems, ICICIS 2015*, December (2016), 613–619.
18. P. Tan, M. Steinbach, V. K. *Data Mining Cluster Analysis: Basic Concepts and Algorithms*. 2013.
19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.
20. Perera, L. P., Oliveira, P., and Guedes Soares, C. Maritime Traffic Monitoring Based on Vessel Detection, Tracking, State Estimation, and Trajectory Prediction. *IEEE Transactions on Intelligent Transportation Systems 13*, 3 (2012), 1188–1200.
21. Pourrajabi, M. Model selection for semi-supervised clustering. *Proceedings of the 17th International Conference on Extending Database Technology (EDBT) 28*, 2 (2014), 49–60.
22. Ristic, B., La Scala, B., Morelande, M., and Gordon, N. Statistical analysis of motion patterns in AIS data: Anomaly detection and motion prediction. *Proceedings of the 11th International Conference on Information Fusion, FUSION 2008* (2008), 40–46.
23. Riveiro, M., Johansson, F., Falkman, G., and Ziemke, T. Supporting maritime situation awareness using Self Organizing Maps and Gaussian Mixture Models. *Frontiers in Artificial Intelligence and Applications 173*, January (2008), 84–91.
24. Sheng, P., and Yin, J. Extracting Shipping Route Patterns by Trajectory Clustering Model Based on Automatic Identification System Data.
25. Tu, E., Zhang, G., Rachmawati, L., Rajabally, E., and Huang, G. B. Exploiting AIS Data for Intelligent Maritime Navigation: A Comprehensive Survey from Data to Methodology. *IEEE Transactions on Intelligent Transportation Systems 19*, 5 (2018), 1559–1582.
26. Zheng, Y. U. Trajectory Data Mining : An Overview. 1–41.
27. Zhong, S., and Ghosh, J. A Unified Framework for Model-Based Clustering. *Journal of Machine Learning Research 4* (2003), 1001–1037.