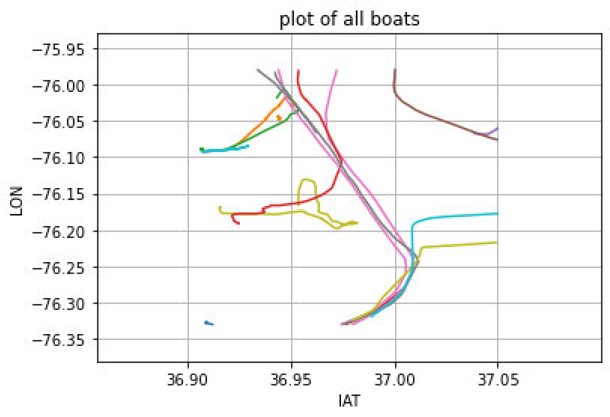
Case Study 2: Marine Vessel Trajectory Clustering

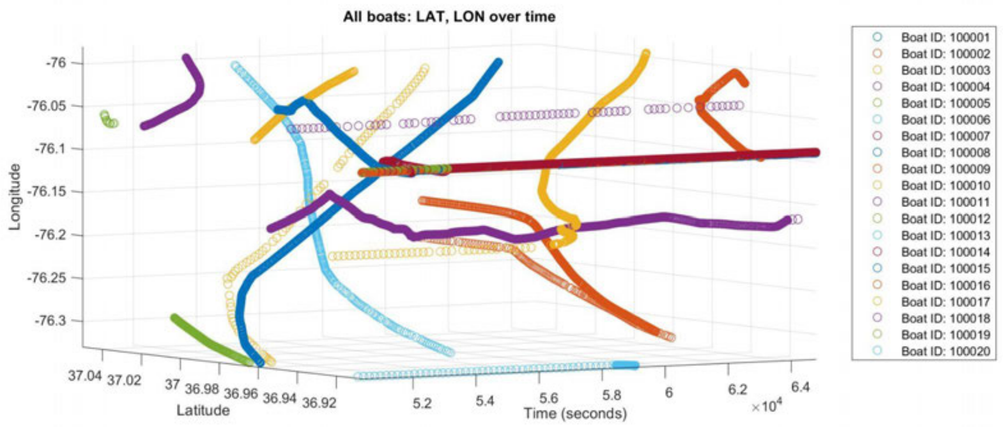
Introduction

Tracking multiple moving targets over time is important for a variety of applications, one of which is tracking ships. Marine vessels of a certain size have transponders aboard that periodically broadcast position, movement, and a form of identification. This automatic identification system (AIS) produces a large dataset that is well suited for a machine learning task. Specifically, we investigated vessel identification when AIS data is missing vessel ID (VID) information. To achieve this aim we visualized the given datasets, reviewed the object tracking literature, and implemented a two stage algorithm which we based on the paper from Ahmed et al [1].

Approach

In order to understand the nature of the data we would be working with, visualization was the first step. Figures 1 and 2 display dataset 1, where each color represents a different vessel. From 1 (latitude vs longitude), it was clear that the vessels tend to follow similar paths, indicating that a clustering task would be non-trivial. When considering time, plotting in 3D was the most natural. In 3D, there was much more space between clusters than in 2D, but overlapping clusters were still apparent (such as the green and red cluster at t = 5.2).





As expected, clusters weren't spheroids, ruling out algorithms like k-means. Instead, due to the nature of the data, the clusters appeared as long strings over the time dimension. Our first idea of how to identify clusters of this shape was using a custom distance metric involving the current and predicted distance between two points. After calculating distances in this manner, they would then be passed into a variety of clustering algorithms, the plan being to keep the one with the best ARI. The calculation of the distance matrix for this approach took over 8 hours and when used as an input for various clustering algorithms, none produced an ARI above baseline.

Our next idea was to use a neural network where any two points would give the likelihood value of whether they were consecutive in the same path or not. We used various techniques to pass in equal data with 50% being consecutive pairs and 50% not. The neural network computed the distance matrix accurately but again, using it as an input to clustering algorithms produced low ARIs. Additionally, the time to compute the distance matrix was around 4-5 hours, after many optimizations, meaning it would be impractical for iterating our design. Below is a summary of the various clustering algorithms and the ARI they achieved on dataset 1:

Time Series KMeans with Dynamic Time Warping:	0.163
DBSCAN with Predictive Distance Likelihood:	0.110
Hidden Markov Model:	0.123
DBSCAN with Mahalanobis distance	0.169
Neural Network-Enhanced Hierarchical Clustering	-0.001
Custom Iterative based clustering (algorithm used)	0.631

We also considered an agglomerative hierarchical design using single linkage clustering, but we were concerned this would fail in the case where a boat was transmitting its AIS data infrequently. This would introduce the possibility of a point from some vessel being further in time than a point from a different vessel in space. In this case, the algorithm would incorrectly add the other vessel to the cluster. Additionally, this algorithm ignored the course and speed over ground information, which we were beginning to think would be necessary for a successful algorithm.

With our initial ideas leading nowhere, we decided to review the object tracking literature for insight. Ansari et al describes the different types of spatiotemporal clustering including moving object, object trajectory, and geo-referenced time series problems [2]. We discerned that our task is an object trajectory problem, which helped refine further searching. Dorabiala et al's description of a spatiotemporal k-means algorithm seemed promising but lacked implementation details [3]. Eventually, we came across the paper from Ahmed et al, entitled "A Spatio-temporal Track Association Algorithm Based on Marine Vessel Automatic Identification System Data." This paper described an algorithm for our specific task. We decided that understanding and implementing this would be the best way to achieve high accuracy on dataset 3. Along with this, once we did implement the algorithm its ARI easily beat baseline and all the other algorithms we have tried, therefore we decided to stick with it.

Algorithm

Data Preprocessing

We converted all time values to be time in seconds elapsed from the first scan, all speed over ground values to be m/s, and all course over ground values to be in terms of ones, allowing us to use these values with various positional prediction equations. We decided to do no feature selection or feature extraction as all the given (SEQUENCE_DTTM, LAT, LON, SPEED_OVER_GROUND, COURSE_OVER_GROUND) all held important information that we believe would hurt our algorithm if removed or combined.

Algorithm Implementation

This algorithm is composed of two stages. In the first stage, the algorithm decides whether to add a given point to a pre-existing track (cluster) or create a new track. If the point is similar to the predicted next point for a track, it is added to the track. If it is too dissimilar, it becomes the first point in a new track. The algorithm is tuned to be quite sensitive to dissimilarity and hence generates many more than the true number of tracks. This part of the algorithm is able to generate its own minimized number of clusters and works without specifying and k values. The second stage is designed to combine these tracks using a variety of heuristics that account for time gaps in a vessel's AIS data and sudden changes in its course over ground.

This algorithm is an **unsupervised algorithm** as it does not rely on inputted labels during training. It uses only the test features to establish which points belong to the same group.

Stage 1: Online Track Association

As mentioned above, stage 1 relies on a prediction and a dissimilarity metric. For an unclustered point i in the set of unclustered points I, the algorithm checks each element in the set of existing tracks S. Let A be one of these tracks with most recently added datapoint a. Following Vincenty's geodesic problem, the prediction uses a's position, speed over ground, and course over ground to predict the track as a function of time, f(t). Using the time difference between i and a, t_ai, as the argument to the function, the specific latitude and longitude of the predicted point is returned: a' = f(t_ai).

Now the spatial dissimilarity between i and a' can be computed. The angular dissimilarity between i and a (not a') can also be computed and the two metrics can be added to create a total dissimilarity score. This process can be repeated for each track in S, and i can be added to the track with the minimum dissimilarity, as long as that dissimilarity is less than some threshold βsmall. If the dissimilarity is greater than some βlarge for all tracks in S, then i becomes the first point in some new track. If the dissimilarity is between βsmall and βlarge, the decision is based on variable parameters, like distance traveled. For the specific values of these thresholds, we began by using the optimal values reported by the paper. Using randomized gridsearch to test a range of other combinations of values, however, we were unable to get ARI values about the suggested hyperparameters.

Stage 2: Post-hoc Merging

As mentioned before, stage 2 combines tracks from the many created by stage 1. This stage uses the minimum and maximum latitude and longitude values to set the boundaries of the area under consideration. We used gridsearch to determine the boundary padding, the distance from the border after which points could be included in merges. We landed on a boundary distance of 0.01°. We included this boundary check because if a track starts within the boundary then we assume it didn't come from open sea or the dock, meaning it belongs to a pre-existing track.

For some track B, a set of possible tracks P that could be merged with it is assembled by seeing if a given track meets the above boundary condition and starts after B ends. Iterating through P can reveal time gaps, i.e. where stage 1 created a new track that was really the continuation of an old one that stopped transmitting for some time. It can also reveal sharp turns, i.e. where stage 1 created a new track for a vessel that changed direction rapidly. Both of these misclassifications can be counteracted. A track starting more than τ seconds after another and staying within γ meters of it is determined to be a continuation after a time gap and the two tracks are merged. Tracks with their start and end nodes within some η is likely an artifact of a quickly turning vessel. These tracks can be merged. These values for τ, γ, and η were given by the paper and we ran a gridsearch based on ARI to confirm that these were indeed the optimal values.

For the case where we are given the number of tracks k, we adapted the above-described merging to force merge the closest tracks until there were only k tracks left, ensuring that the number of tracks did not exceed the number provided to us. Before arriving at force merging, we tried two other methods: in one, after k tracks are generated, simply do not allow the creation of anymore. In another, the β parameters from stage 1 would adapt based on the number of tracks produced relative to the given k. While all of these manipulations decreased the ARI slightly, force merging tracks decreased it the least, making this our choice when given k. These are the ARIs for each approach:

Before merging clusters of set 1:	0.636
Stop creating clusters after k clusters:	0.357
Adaptive beta:	0.462
Force merging:	0.573

Recommendations

There are many possible metrics we can use that are tailored to our problem and give us a better evaluation and understanding of our effectiveness. One possibility is using Average Path Deviation, or APD. APD calculates the deviation between the prediction and actual paths of each vessel. This would give some insight into how well our algorithm is able to pick out and follow the vessel movements in this multiple-target tracking problem. Another algorithm that may be used is Multiple Object Tracking Accuracy, or MOTA. MOTA can help with misses, false positives, and identity switches, providing deep insight into how well our algorithm is able to stick to one track.

One way to deal with gaps in data (e.g. no data available from all vessels for 10 minutes) is similar to the method described in the stage 1 section: extrapolating each vessel's speed and direction over the gap and once the data comes back, seeing if the data is inline with the extrapolation. This is not able to account for boats rotating during the gap. Another possibility is using a supervised learning algorithm that would take the current track and calculate which vessel it is most likely to be. It could then predict the point it is most likely to end up at after the gap ends. An issue we might encounter is data inaccuracies creating objects that we do not want to track. To avoid this issue, we can run an algorithm such as DBSCAN, which can detect and remove outliers from clusters. This helps to ensure that we are focusing on the true path of each vessel.

References:

1.
Ahmed I, Jun M, Ding Y. A Spatio-Temporal Track Association Algorithm Based on Marine Vessel Automatic Identification System Data. *IEEE Transactions on Intelligent Transportation Systems*. 2022;23(11):20783-20797. doi:https://doi.org/10.1109/tits.2022.3187714
2.
Ansari MY, Ahmad A, Khan SS, Bhushan G, Mainuddin. Spatiotemporal clustering: a review. *Artificial Intelligence Review*. Published online July 15, 2019. doi:https://doi.org/10.1007/s10462-019-09736-1
3.
Dorabiala O, Webster J, Kutz N, Aravkin A. Spatiotemporal k-means. arXiv.org. doi:https://doi.org/10.48550/arXiv.2211.05337