IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 15, 2021, accepted June 22, 2021, date of publication June 28, 2021, date of current version July 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3092948

A Comparison of Trajectory Compression Algorithms Over AIS Data

ANTONIOS MAKRIS[1], IOANNIS KONTOPOULOS[1], PANAGIOTIS ALIMISIS[1], AND KONSTANTINOS TSERPES[1,2]

[1]Department of Informatics and Telematics, Harokopio University of Athens, 177 78 Athens, Greece
[2]Department of Electrical and Computer Engineering, National Technical University of Athens, 157 73 Athens, Greece

Corresponding author: Antonios Makris (amakris@hua.gr)

This work was supported by the MASTER and SmartShip Projects through the European Union's Horizon 2020 Research and Innovation Programme under the Marie-Slodowska Curie Grant 777695 and Grant 823916.

This work did not involve human subjects or animals in its research.

ABSTRACT Today's industry is flooded with tracking data originating from vessels across the globe that transmit their position at frequent intervals. These voluminous and high-speed streams of data has led researchers to develop novel ways to compress them in order to speed-up processing without losing valuable information. To this end, several algorithms have been developed that try to compress streams of vessel tracking data without compromising their spatio-temporal and kinematic features. In this paper, we present a wide range of several well-known trajectory compression algorithms and evaluate their performance on data originating from vessel trajectories. Trajectory compression algorithms included in this research are suitable for either historical data (offline compression) or real-time data streams (online compression). The performance evaluation is three-fold and each algorithm is evaluated in terms of compression ratio, execution speed and information loss. Experiments demonstrated that each algorithm has its own benefits and limitations and that the choice of a suitable compression algorithm is application-dependent. Finally, considering all assessed aspects, the Dead-Reckoning algorithm not only presented the best performance, but it also works over streaming data, which constitutes an important criterion in maritime surveillance.

INDEX TERMS Error metrics, lossy compression techniques, similarity measures, simplifying trajectory algorithms, trajectory compression algorithm, trajectory similarity.

I. INTRODUCTION

In recent years, the number of vessel tracking data has drastically increased, following an impressive exponential trend. This is due to the fact that not only all larger vessels are obliged to be equipped with an Automatic Identification System (AIS) transponder, but smaller vessels are also adopting this technology voluntarily. AIS is a global tracking system that allows vessels to transmit data about their whereabouts. Through the AIS, vessels are able to be aware of vessel traffic in their vicinity and avoid potential collisions. Despite the fact that the initial purpose of the AIS was safety, it did not take long for the maritime authorities to exploit it for the identification of illegal vessel activities and the monitoring of vessels' behavior.

The exploitation of AIS data from the maritime authorities has shifted the focus of researchers' attention towards the development of trajectory mining techniques. Such techniques allow the authorities to further take advantage of the trajectories formed from the AIS messages either in real-time [1]–[3] or on historical data [4]–[6]. The main challenge to be tackled in the development of trajectory mining techniques is the data's huge volume generated globally and the associated high frequency. The increasing amount of data poses new challenges related to storing, transmitting, processing, and analyzing these data [7]–[11]. For instance, the transmission rate of the AIS protocol can reach a frequency of one message every two seconds per vessel. Such a frequency in combination with the approximately 200, 000 vessels using the AIS worldwide, yields a total amount of 16, 000 messages per second and 46GB of data per day. Thus, the enormous volumes of trajectory data can quickly overwhelm available data storage systems and the redundant information often contained in these data can overwhelm human analysis. In general, the more points collected, the more accurate a trajectory representation becomes. Nevertheless, most vessel

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Zakarya.

movements on the water are relatively steady in space, thus the shape of a vessel trajectory can be represented by only a small portion of carefully selected points.

A typical approach towards these challenges is to reduce the size of trajectory data by employing compression techniques. A compression algorithm is required to not only compress redundant position information but also retain the vessel's behaviour information included in the trajectory. In other words, trajectory compression aims at substantial reductions in the amount of data while minimizing the loss of information and at the same time preserving the quality of trajectory. Quality is strongly related to the information loss and measured through metrics that calculate the applied error over a trajectory if a certain point is discarded. Therefore, it is become clear that there is a trade-off between the compression rate and the quality of trajectory achieved after compression [12].

The objective of this work is to evaluate some of the most representative lossy compression algorithms over AIS data. Specifically, the algorithms evaluated are: Douglas-Peucker (*DP*), Time-Ratio (*TR*), Speed-based (*SP*), Heading-based (*HD*), Time-Speed-Heading-based (*TSH*), STTrace (*STT*), SQUISH (*SQ*), Dear-Reckoning (*DR*) and Open-Window Time-Ratio (*OWT*). The comparison is based on the compression ratio and the execution time achieved across a real-world dataset. Their performance characteristics were compared against different dynamically defined thresholds. Each trajectory is different from others, hence it is beneficial to select the appropriate threshold for each trajectory according to its characteristics. Threshold refers to the discarding criterion employed by each algorithm. Thus, we suggest an automatic technique, in which a different threshold is applied in the corresponding algorithm, which depends on the actual features and peculiarities of each trajectory. In this way, we have eliminated the need of arbitrary user-defined thresholds.

The examined algorithms are lossy which means they present information loss and therefore it is important to measure the quality of the compressed AIS data. Different error metrics are used in the literature to evaluate compression. For instance, the error in [13] is provided by a formula which measures the mean error of the compressed representation in terms of distance from the original trajectory. In [14] the average and median synchronized Euclidean distance (SED) as well as the speed error are employed in reference to compression ratio while in [15] the same metrics are used with the extra addition of heading error. Furthermore, in [16] the average SED, spatial and speed error are utilized. In this research work, the quality of the compression is evaluated by employing similarity measures over both the original and the compressed trajectory data, in order to evaluate their accuracy. The purpose of a similarity measure is to obtain a quantitative measure between any two trajectories, thus to identify to what extent two objects are similar. Most of the well-established similarity measures selected in this work, consider the shape of the trajectory which consist a characteristic of utmost importance in the maritime domain and can provide insights about vessels' behavior [3], [17].

The main contributions of this study are as follows:

- nine trajectory compression algorithms are compared and evaluated. The examined algorithms are suitable either for historical data or real-time data streams compression;
- the compression evaluation is based on six different trajectory similarity measures which consider the spatiotemporal aspect of the trajectory;
- different dynamically defined thresholds are applied by each compression algorithm based on the actual features and peculiarities of each trajectory, thus eliminating the need for arbitrary user-defined thresholds;
- Dead-Reckoning is considered the best algorithm when dealing with AIS data, followed by Douglas-Peucker and Open-Window Time-Ratio. These algorithms presented the most suitable performance in terms of compression ratio, execution time and information loss.

The rest of the paper is structured as follows. Section II serves as a literature review in the field of trajectory compression. Section III describes the compression algorithms while Section IV presents the similarity measures. Section V details the compression evaluation and presents the results, while Section VI summarizes the merits of our work.

II. RELATED WORK

Trajectory compression algorithms are widely used in various areas, such as cartography and computer graphics, trajectory clustering, road traffic, pedestrian movement information, weather analysis and map generalisation. The aim of trajectory compression algorithms is to balance the trade-off between the achieved compression rate and the acceptable degree of error.

Meratnia and de By [18] was among the first research studies considering three dimensional mobility data, where the temporal factor was taken into consideration in the compression techniques. Leichsenring and Baldo [19] present an evaluation of seven lossy compression algorithms with a view to identify the most important aspects in selecting the appropriate compression algorithm. Muckell *et al.* [15] also present a performance comparison of seven compression algorithms when utilizing two different errors, SED and the median difference in speed. The comparison is based on the execution times of compression algorithms and on the error committed. Chen *et al.* [20] present a lossy compression batch algorithm for GPS trajectories that considers both line simplification and quantization in the compression process. The quantization technique can improve the encoding procedure that takes into account speed and direction changes for selecting the approximated trajectory for compression. For the evaluation of the compression, the maximum SED is employed. Birnbaum *et al.* [21] present a method which exploits the similarities between sub-trajectories and creates a time mapping by finding for each time value of a trajectory the corresponding time value of a similar trajectory, using

linear interpolation. A compression algorithm is applied on the time mapping, which removes the points but keeps the compression error under a user-defined threshold. The strong correlation between time values of similar trajectories, allows high compression of time mapping.

A dynamic storage system called TrajStore which is able to co-locate and compress spatially and temporally ''close'' adjacent trajectories is presented in [22]. The compression algorithm combines two schemes: a lossless delta compression scheme which encodes spatiotemporal coordinates within a trajectory and a lossy compression scheme for clustering trajectories traveling on nearly identical paths. Another system called REST is presented in [23]. The proposed framework extracts a small collection of subtrajectories, called reference trajectories, from raw trajectory data that form the compressed trajectory within a given spatio-temporal deviation threshold. In addition, greedy and dynamic algorithms achieve an optimal compression ratio and high levels of efficiency.

Regarding vessel trajectory compression from AIS data, several studies have already conducted. Liu *et al.* [24] present an adaptive Douglas-Peucker algorithm with automatic threshold selection for AIS-based trajectory compression. In general, Douglas-Peucker requires a static user-defined distance threshold to decide which points to retain or remove from the resulting curve. However, the choice of an optimal threshold value is difficult and differs across various applications. The proposed algorithm is able to maintain the main geometrical structures of each trajectory and automatically calculates a different threshold for each trajectory based on the actual trajectory characteristics. Then, the DTW algorithm with a warping window is employed in order to calculate the distances between the various time series. The evaluation of the proposed algorithm is based on classification and clustering experiments. Wei *et al.* propose a method in [25], which is able to compress AIS data by considering both spatial and motion features. Douglas-Peucker algorithm is employed to simplify trajectories, according to spatial features and a sliding window is adopted to simplify trajectories based on motion features such as the course alterations and speed variations. A fixed threshold of 0.8 times the ship length is employed in DP, as proposed in [26], while for sliding window the Gaussian distribution is utilized for threshold determination regarding speed and course variations. Then, a merging operation is performed, which gathers the simplification results of the two methods. The final compression results are evaluated based on the compression rate, length loss rate and shape similarity. DTW method is used to calculate the similarity between the original and the compressed trajectory. On the other hand, Zhao and Shi [27] present an improved version of Douglas–Peucker which considers the shape of the vessel trajectory derived only from course. The method is able to recognize the straight and curve parts of a trajectory by detecting the transition point. Transition points are directly retained as the start and end points of the track segments. In the first phase of the algorithm, a window detects the clear turning while in the second, each segment is compressed by the traditional DP algorithm. For user-defined threshold elimination, the threshold of each individual trajectory is based on each ship's length. In [28], the authors propose a density-based spatial clustering of trajectories based on DBSCAN. In the clustering process, the similarities between trajectories are computed. As the number of AIS points are extremely large, the similarity process can be prohibitively costly. Thus, the DP algorithm is used again for AIS trajectory compression. The threshold of the algorithm is determined according to the changes in shape of the trajectories. Etienne *et al.* [29] reduced the number of positions of a trajectory by adopting the DP algorithm while retaining only the virtual positions. Their purpose was to optimize the calculation time of traffic flow pattern recognition. Vries and Someren [30] employed DP algorithm to retain the stop and move information on the ship's trajectory. To the compressed trajectory data, alignment based similarity measures are applied. It is evident that Douglas-Peucker is among the most popular and successful algorithms on trajectory simplification due to its speed and accuracy.

In [31] the compression effect is evaluated using the KDE-based AIS vessel density visualization on both the compressed and uncompressed trajectory data. Thus, the threshold value selected is the one for which the visualization results are the same for both compressed and uncompressed data. Similarly in [32], the DP algorithm and kernel density estimation (KDE) algorithms are utilized for trajectory compression and visualization respectively. The massively large-scale parallel computing function computation capabilities of the Graphics Processing Unit (GPU) have been employed to reduce the execution time of the Douglas-Peucker algorithm. The compression is evaluated through the density map visualizations produced by KDE.

As trajectory compression is very sensitive to parametrization, Fikioris *et al.* [33] present an approach for fine-tuning the selection of parameter values for compression. A genetic algorithm that iterates over several combinations of the parameter values is employed until converging to a suitable configuration per vessel type. This genetic algorithm is trained on a public dataset and as the system supports incremental optimization, by training in data batches, the performance continuously improves. The system aims to minimize the compression ratio, while at the same time to minimize the approximation error which is quantified with the Root Mean Square Error (RMSE). Sun *et al.* [34] propose an online compression of vessel trajectories, by utilizing the classic SPM (scan-pick-move) compression algorithm with the addition of a sliding window. The maximum offset distance reference trajectory point in the sliding window is utilized as the criterion of whether to retain or to discard a point. For the compression evaluation several performance metrics are employed such as compression time, compression ratio and compression error.

In our previous study [12], several trajectory compression algorithms are evaluated by employing classification techniques and similarity measures. Specifically, four

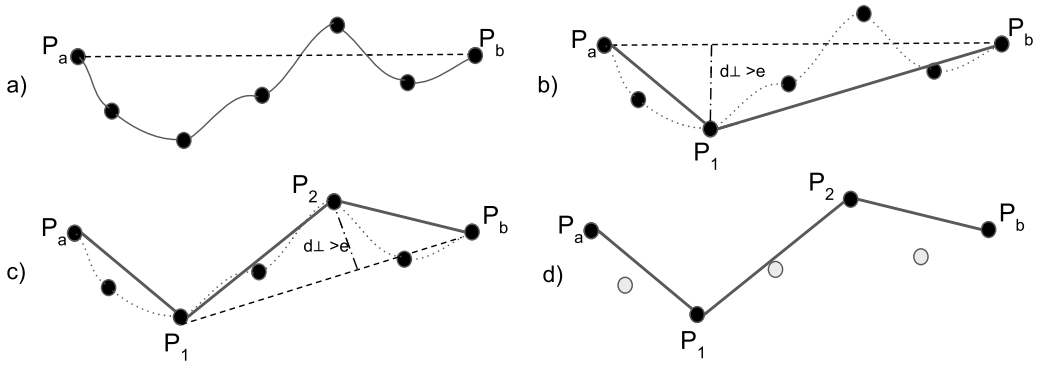


FIGURE 1. Douglas–Peucker algorithm. (a) Line segment formed by anchor and float points. (b) The point (P_1) with the maximum perpendicular distance from line segment is selected. (c) The process is repeated using recursion on each line segment. (d) New simplified trajectory.

classification methods which extract trajectory features are selected based on the classification accuracy. After the extraction of the features, the Random Forest algorithm is used for comparing the different techniques. Two distance measures are employed, EDR and UMS for trajectory similarity. In order to evaluate the similarity results, the F-Score metric was utilized. Four real-world datasets were selected in order to capture the full spectrum of trajectories ranging from the mobility of people, vehicles, animals even natural phenomena. On the other hand, in the present work our intention is to focus specifically on the effect of compression over AIS data. Thus, a wide range of offline and online compression algorithms are compared in contrast to our previous work which compares only five offline compression algorithms. Furthermore, the evaluation process in this work is based solely on the average distance produced by the different similarity measures. Finally, the intention of this work is to provide some insights regarding the selection of the most suitable compression algorithm for AIS data while in our previous work we focused on the effect of compression in trajectory similarity and classification problems.

III. TRAJECTORY COMPRESSION ALGORITHMS

Compression algorithms can be classified into two broad categories depending on the moment the compression procedure is performed [35]: offline and online.

More specific, offline compression is performed after a trajectory has been fully generated by discarding redundant points from the original trajectory. Online algorithms compress the trajectory during the point collection process which has the advantage of supporting online applications [36].

Offline algorithms, also known as batch algorithms, are able to obtain more accurate results and present smaller errors than online algorithms as the algorithm has a global view of the entire trajectory. On the other hand, online algorithms remove redundant data from trajectories as they occur, avoiding the unnecessary transfer of data over the network, improving data storage and reducing the memory space. With the exponential growth of AIS data, online algorithms seem to be a more appropriate approach for data compression. AIS data will eventually become prohibitively "big" for storage and transmission, thus offline algorithms are not suitable, as only collected historical data can be compressed.

In this research work, the examined algorithms are lossy in the sense that attempt to preserve the major characteristics of the original trajectory by removing the less significant data when compared to the original data. The main advantage of lossy compression techniques is that they can reduce storage size while maintaining an acceptable degree of error [14].

A. OFFLINE COMPRESSION ALGORITHMS

1) DOUGLAS-PEUCKER (DP)

Douglas-Peucker algorithm was proposed in 1973 [37] as an approximate simplification method with error bound guarantee. The process of the algorithm is illustrated in Figure 1. An AIS trajectory is represented as a point set $S = \{P_a, \ldots, P_b\}$, as shown in Figure 1a. Initially, the first (anchor) and last (float) point from the trajectory are selected and these two points form a line segment $LS_{P_a \to P_b}$. The starting curve is a set of points and a threshold (distance dimension) $\varepsilon > 0$. After, the algorithm calculates the perpendicular distance PD, of each point (P_i) between the trajectory and its projection on the line segment $LS_{P_a \to P_b}$. Subsequently, the point with the maximum perpendicular distance d_{P_1} from $LS_{P_a \to P_b}$ is selected (P_1). If $d_{P_1} > \varepsilon$, the point is retained in the resulting set and becomes the new float point for the first segment, and the anchor point for the second segment as shown in Figure 1b. Thus, the original trajectory is splitted into two sub-trajectories $STr_{P_a - P_1}$ and $STr_{P_1 - P_b}$. Otherwise, if $d_{P_1} < \varepsilon$, all the points between P_a and P_b are discarded. This process is repeated using recursion on each line segment (Figure 1c). The algorithm halts when the maximum distance between the original trajectory and the line segments is below ε. When the recursion is completed, a new simplified trajectory is generated of only those points that have been marked as kept (Figure 1d). In the worst case scenario its running time is $\mathcal{O}(n^2)$ where n is the amount of points in the trajectory. An improved version which includes convex hulls (DPhull) is presented in [38] with running time $\mathcal{O}(n \log n)$.

2) TOP-DOWN TIME-RATIO (TR)

The main drawback of line generalization algorithms like Douglas-Peucker, is that when dealing with trajectory data of moving objects, they do not consider the temporal aspect. They treat each trajectory as a line in two-dimensional space. But a trajectory has an important extra dimension, time. Time-Ratio algorithm computes the distances between pairs of estimated temporally synchronized positions, one on the original and one on the corresponding approximated trajectory as illustrated in Figure 2.

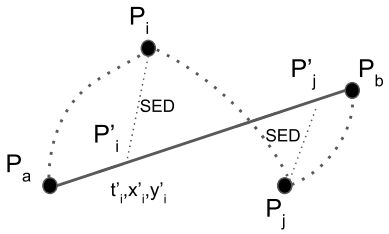


FIGURE 2. Time Ratio algorithm.

For each point on the original trajectory such as P_i, the temporally synchronized point $P_{i'}$ is located on the approximated trajectory Tr_{Pa-Pb} and the coordinates (x_i', y_i') of Pi' are calculated using linear interpolation as:

$$x_i' = x_a + \frac{t_i - t_a}{t_b - t_a}(x_b - x_a)$$
$$y_i' = y_a + \frac{t_i - t_a}{t_b - t_a}(y_b - y_a) \quad (1)$$

After the temporally synchronized points are determined, the next step is to calculate the distance between Pi' and Pi. If this distance is greater than a user-defined threshold, the particular point is included in the resulting set otherwise it is discarded. By including the temporal information, as well as spatial data in its compression heuristic, the algorithm provides more accurate results. The worst-case running time of TR is $\mathcal{O}(n^2)$ since it extends the original Douglas-Peucker algorithm.

3) SPEED-BASED (SP)

Speed-based algorithm exploits the speeds from subsequent segments of a trajectory. If the absolute value of speed difference of two subsequent trajectory segments, for example $|V_{STr_{Pb-Pc}} - V_{STr_{Pa-Pb}}|$, is greater than a threshold, the point in the middle (P_b) is retained otherwise it is discarded. The process continues until all points in the trajectory are examined. The algorithm is illustrated in Figure 3.

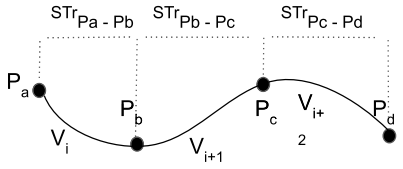


FIGURE 3. Speed-based algorithm.

4) HEADING-BASED (HD)

Heading-based algorithm exploits the angle formed by subsequent segments of a trajectory. Initially, the distances of continuous segments of the trajectory are calculated; i.e. d_STr_{Pa-Pb} and d_STr_{Pb-Pc}. Then for each triangle formed by three continuous points like $P_aP_bP_c$, the distance (length) of the opposite side of the examined angle is calculated e.g. d_STr_{Pa-Pc}. Knowing the size of three sides and utilizing the law of cosines, the angle is calculated. For example we can calculate A_1 ($\angle$ $PaPbPc$) as $c^2 = a^2 + b^2 - 2ab\cos\gamma$, where γ denotes the examined angle (A_1) contained between the sides of length a (d_STr_{Pa-Pb}) and b (d_STr_{Pb-Pc}) and located opposite the side of length c (d_STr_{Pa-Pc}). If the angle of two subsequent segments is greater than a threshold, the point in the middle is retained otherwise it is discarded. The process continues until all points in the trajectory are examined. The algorithm is illustrated in Figure 4.

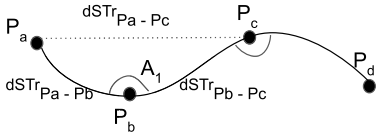


FIGURE 4. Heading-based algorithm.

5) TIME-SPEED-HEADING-BASED (TSH)

By integrating the concepts of time ratio distance, speed and heading, a new algorithmic approach can be obtained. The proposed algorithm, namely Time-Speed-Heading-based, considers ship behaviour characteristics in terms of motion features related to speed and course variations but also examines time, in the sense of synchronized euclidean distance of temporally synchronized points. The algorithms sets an anchor point and then gradually ''opens the window''. In each recursion, three halting conditions are examined, the synchronous euclidean distance measure, the speed difference and the heading difference between trajectory subsegments. If SED, speed or heading are greater than a SED, speed and heading threshold respectively, the point is retained otherwise it is discarded. Of note, the integration of different compression algorithms is not a novel idea but was originally proposed by Meratnia and Rolf in [13].

B. ONLINE COMPRESSION ALGORITHMS

1) SPATIOTEMPORAL TRACE – STTrace (STT)

STTrace was proposed by Potamias *et al.* [11] and introduces the use of SED as an error metric. The algorithm is designed to preserve spatiotemporal, heading and speed information in a trace. Initially, every incoming point along with its SED is inserted in the allocated memory. The SED is calculated after its successor is stored in the sample. As soon as the allocated memory gets exhausted and a new point is examined for possible insertion, the sample is searched for the item with the lowest SED, which represents the least possible loss of information. If SED of the inserted point is larger than

the minimum one found already in the sample, the current processed point is inserted into the sample at the expense of the point with the lowest SED. When a point is excluded from the memory, SED attributes for the neighboring points of the removed one are recalculated along with a reassessment of the new minimum SED. The time complexity of the algorithm is $\mathcal{O}(\frac{1}{n}\log\frac{n}{m}\log m)$ where n refers to trajectory size (number of points) and m to the memory size.

2) SPATIO QUALITY SIMPLIFICATION HEURISTIC – SQUISH (SQ)

SQUISH is presented by Muckell *et al.* [14] and although it is a similar approach to STTrace, it differs on the criteria of adding a point in memory. The algorithm requires an input parameter, i.e. the size of a buffer that contains the number of points that will exist in the final data after compression. Initially, all incoming points are inserted in the buffer until is full and any new incoming point requires the removal of another point that is already stored inside the buffer. The selection of the point to be removed is based on the SED estimation if that point is removed. In other words, the removed point is that with the minimum estimated SED error, which will introduce the lowest amount of error in the compression. Subsequently, the algorithm estimates the upper bound SED error of the adjacent neighbors by adding the SED value of the deleted point. Since removal of any point would require reassigning priorities to every point in the buffer, the prioritization algorithm uses local optimization instead of a more accurate global approach. The time complexity of the algorithm is $\mathcal{O}(n\log(n'))$ where n refers to trajectory size (number of points) and n' to the compressed trajectory size.

3) DEAD-RECKONING (DR)

Dead-Reckoning algorithm, proposed by Trajcevski *et al.* [39], employs a prediction location strategy in order to estimate the localization of the next trajectory point in time, using the point's current position and velocity. The algorithms requires a single tolerance value as an input parameter, which specifies the maximum distance a future position can deviate from the estimated position. Then, the distance between the predicted location and the actual location of an incoming point is calculated. If this distance is greater than the defined tolerance, the point is included in the compressed trajectory, otherwise, the point is discarded. The algorithm has a Sliding Window approach with $\mathcal{O}(n)$ time complexity. This complexity is due to the fact that it takes only O(1) time to compare the actual and predicted locations of each point.

4) OPENING-WINDOW TIME-RATIO (OWT)

Opening Window algorithm proposed by Meratnia and Rolf [13] starts by defining a segment between the anchor (first data point) and the float (third data point) in a trajectory. Subsequently, the perpendicular distances of each intermediate point are calculated and if they are under a predefined error threshold, the float is moved to the next point. The process continues until the perpendicular distance of a point inside the window exceeds the error threshold. Then, two strategies can be employed depending on which version of the algorithm is executed: i) the data point causing the threshold violation is the one selected to remain (Normal OW) and ii) the predecessor of the actual float point is selected (Before OW). The selected point becomes the anchor and the float is set two points ahead. If no threshold is exceeded, the float is moved one up the data points and the method continues, until the entire series has been transformed into a piecewise linear approximation. Opening-Window Time-Ratio is an improved version of the OW algorithm. It employs the time-ratio distance measuring technique presented in III-A2, thus considering the temporal aspect of the trajectory in the compression process. The worst-case running time of Opening Window algorithms is $\mathcal{O}(n^2)$.

Table 1 depicts the basic characteristics of the compression algorithms compared in this study. Specifically, the compression algorithms are categorized into offline and online mode and the computational complexity is presented in terms of time complexity and memory (space cost). Furthermore the error criterion of each algorithm is presented. Each trajectory compression algorithm employs particular heuristic(s) in order to discard or retain points. Different error metrics such as perpendicular distance, time-ratio distance, synchronized Euclidean distance etc. are utilized as criteria to decide which points to discard. Finally, the table illustrates the different trajectory features each algorithm considers in compression. For example, DP only considers the geometrical shape of a trajectory while TR takes also time into consideration. TSH considers all the examined features followed by DR which considers all except heading. As the table suggests, online algorithms tend to consider more features in compression than offline.

C. THRESHOLD DEFINITION

One of the challenges is to define the required thresholds to be employed by the compression algorithms. Setting the proper threshold can significantly affect the compression results in terms of compression ratio and achieved quality. Threshold determination is an application-dependent process and varies between different studies. For example, in [26], [40] the 0.8-times the ship length-threshold is utilized while in [41] the threshold distance is set experimentally to 500 m. Each trajectory is different, hence it is beneficial to select the appropriate threshold for each trajectory automatically.

In general, each of the examined algorithms follows the steps below:

- group the points and create a trajectory of each object based on an identifier. This practically means that the number of trajectories in a dataset is as large as the number of objects
- compress the whole trajectory of each identifier
- write the points remaining after compression to a file grouped by identifiers

As we group the points for each identifier (object) in the dataset, we extract the trajectories (one trajectory for

TABLE 1. Summary of trajectory compression algorithms.

Algorithm	Mode	Time Complexity	Space Cost	Error Criterion	considers geometric shape	considers time	considers speed	considers heading
DP	offline	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Perpendicular distance (PD)	✓	×	×	×
TR	offline	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Time-ratio distance (TRD)	✓	✓	×	×
SP	offline	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Speed difference	×	×	✓	×
HD	offline	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Heading difference	×	×	×	✓
TSH	offline	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	SED, Speed diff, Heading diff	✓	✓	✓	✓
STT	online	$\mathcal{O}(\frac{1}{n}\log\frac{n}{m}\log m)$	$\mathcal{O}(m)$	SED	✓	✓	×	×
SQ	online	$\mathcal{O}(n\log(n))$	$\mathcal{O}(m)$	SED	✓	✓	×	×
DR	online	$\mathcal{O}(n)$	$\mathcal{O}(1)$	SED distance, speed	✓	✓	✓	×
OWT	online	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Time-ratio distance (TRD)	✓	✓	×	×

each object). In order to determine the threshold that each algorithm will use, a dynamic process is proposed: for each individual trajectory a different threshold is automatically defined based on an average. Specifically, the average value refers to the discarding criterion of each algorithm. For example, in case of DP the average epsilon (ε) is calculated while for STTrace two thresholds are calculated, the average speed and the average orientation. Then we use this average value as a reference point to define a common rule for the threshold calculation. This practically means that in every trajectory of each dataset a different threshold is applied in the corresponding algorithm, which depends on the actual features and peculiarities of this trajectory. Thus, we have eliminated the need of arbitrary user-defined thresholds.

IV. TRAJECTORY SIMILARITY MEASURES

In this section, we describe the similarity measures that were used in the experiments to evaluate the trajectory compression algorithms. Several trajectory similarity algorithms have been proposed and evaluated in the literature [42]–[47], each with its own advantages and drawbacks. Similarity measures are employed for different purposes in information retrieval and data mining, such as top-K similarity queries, trajectory outlier detection and clustering techniques [48]. One of the most recent studies [49], has created a hybrid unsupervised learning method for trajectory similarity computation, by combining auto-encoders and convolutional neural networks. The original vessel trajectories are transformed into images which in turn are remapped into two-dimensional matrices. Then, low-dimensional representations of the aforementioned produced trajectory images were obtained using the proposed hybrid network. Finally, the trajectories similarities are measured by calculating the distances between the low-dimensional feature vectors learned from the corresponding trajectory images. The results show that the proposed method outperformed both Fréchet distance and DTW in terms of efficiency and effectiveness.

Most of the well-established similarity measures selected in this work consider both the spatial and the temporal aspect of the trajectory [50]. The spatial aspect or the shape of a trajectory is a key characteristic that is of utmost importance in the maritime domain and can provide insights about vessels' behavior [3]. The well-known Longest Common Sub-Sequence (LCSS) [51] and Edit Distance on Real sequence (EDR) [52] algorithms were not selected because they try to find matching pairs of points based on a user-defined parameter, ε. This parameter is a threshold that defines two points between two trajectories as similar if the distance between the points is less than ε. When comparing compressed and uncompressed trajectories all of the points of the compressed trajectory are contained in the uncompressed trajectory, therefore a parameter of $\varepsilon \geq 0$ would result in misleading results.

A. DYNAMIC TIME WARPING

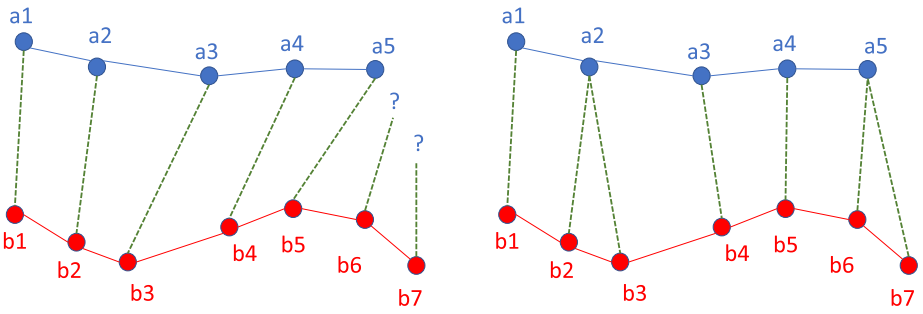
The Dynamic Time Warping (DTW) is a method that calculates the distance between two temporal sequences or time-series of different length and a one-to-one match between these sequences is not possible. Let A and B be two temporal sequences of spatial or positional data points, $A = [a_1, a_2, \ldots, a_{n-1}, a_n]$ and $B = [b_1, b_2, \ldots, b_{m-1}, b_m]$, where n and m is the number of points of A and B respectively and $n \neq m$. These sequences can form a matrix $M = A \times B$, where each point $M_{i,j}$ is the distance between a_i and b_j. The Dynamic Time Warping algorithm then finds an optimal path W between $M_{0,0}$ and $M_{n,m}$. As a result the optimal path W is a set of edges $M_{i,j}$ that connect points of a_i and b_j and the DTW distance is the total distance of these edges:

$$D_{dtw} = \sum_{x=0}^{k} W_x \tag{2}$$

where k is the number of elements in W. More details on the optimal path finding can be found in [53]. Figure 5 illustrates an example of the DTW algorithm. In Figure 5(a) we can see that a one-to-one match is not possible between sequences of different length. In Figure 5(b) we can see the edges connecting A and B that consist the optimal path W.

B. EDIT DISTANCE WITH REAL PENALTY

The Edit distance with Real Penalty (ERP) algorithm is a metric that computes the distance between two numeric sequences or time-series of different length. Similar to the DTW algorithm, ERP creates a distance matrix M that describes the mapping between two series A and B,



(a) A one-to-one match between A and B is not possible (b) DTW distance between A and B

FIGURE 5. Example of the Dynamic Time Warping distance.

$M = A \times B$ and tries to find an optimal path between $M_{0,0}$ and $M_{y,y}$, where $y = max(n, m)$ and n and m is the number of elements in A and B respectively. The key difference between ERP and DTW is that ERP allows for gaps or sequences of data points that cannot be matched with any other data point. These gaps are then penalized based on the distance of the unmatched points from a constant value g. The distance between a_i and b_i can be formulated as follows:

$$D_{erp}(a_i, b_i) = \begin{cases} |a_i - b_i|, & \text{if } a_i, b_i \text{ not gaps} \\ |a_i - g|, & \text{if } b_i \text{ is a gap} \\ |b_i - g|, & \text{if } a_i \text{ is a gap} \end{cases} \quad (3)$$

More details on the optimal path finding and the computation of the ERP distance can be found in [52].

C. FRÉCHET

The Fréchet distance is a similarity measure between curves and is named after the mathematician Maurice Fréchet. This distance metric takes into account the location and the ordering of the points along the curves, thus reflecting the course of the curves. Let $f : [a, b] \rightarrow V$ and $g : [a', b'] \rightarrow V$ be two curves or continuous mappings, where $a, b \epsilon R$, $a \leq b$ and (V, d) is a metric space. Given these curves the Fréchet distance between them is defined as:

$$D_F(f, g) = \inf_{\alpha,\beta} \max_{t\epsilon[0,1]} d(f(\alpha(t)), g(\beta(t))) \quad (4)$$

where α and β are two arbitrary continuous and non-decreasing functions from $[0, 1]$ onto $[a, b]$ and $[a', b']$ respectively. For instance, assuming that a person walks a dog on a leash, both traverse two separate finite curves with different speeds, but neither can go backwards. The Fréchet distance between these two curves is the length of the shortest leash long enough to traverse both separate paths from the beginning to the end of the curves. According to equation 4, $f(\alpha(t))$ would be the position of the dog and $g(\beta(t))$ would be the position of the person at time t or vice versa. More details on the Fréchet distance can found in [54].

D. DISCRETE FRÉCHET

The difference between the Fréchet distance and its discrete counterpart is that the latter is a restriction of the continuous case described in the example with the person and the dog. Considering the previous analogy one would imagine that in the case of the discrete Fréchet distance, snapshots of both the person's and the dog's curves would be taken at discrete time points, thus representing poly-line points or polygonal curves. Given two polygonal curves, $P = [u_1, u_2, \ldots, u_{p-1}, u_p]$ and $Q = [v_1, v_2, \ldots, v_{q-1}, v_q]$ where u_i and v_i are their discrete points respectively, a coupling L between these curves is defined as a sequence $(u_{a_1}, v_{b_1}), (u_{a_2}, v_{b_2}), \ldots, (u_{a_m}, v_{b_m})$ of distinct pairs where $a_1 = 1$, $b_1 = 1$, $a_m = p$, $b_m = q$ and for all $a = 1, \ldots, q$ we have $a_{i+1} = a_i$ or $a_{i+1} = a_i + 1$ and $b_{i+1} = b_i$ or $b_{i+1} = b_i + 1$, thus respecting the ordering of the points in P and Q. The length of the longest link in L is defined as:

$$||L|| = \max_{i=1,\ldots,m} d(u_{a_i}, v_{b_i}) \quad (5)$$

and the discrete Fréchet distance between P and Q is defined as:

$$D_{dF}(P, Q) = min||L|| \quad (6)$$

where $D_{dF}(P, Q) = D_{dF}(Q, P)$ and $D_{dF}(P, P) = 0$. For a visual illustration, see Figure 6 where there are two poly-lines or polygonal curves: $[a_1, a_2, a_3]$ and $[b_1, b_2]$. The possible couplings between the two are $[b_1\ a_1, b_2\ a_2, b_2\ a_3]$ and $[b_1\ a_1, b_1\ a_2, b_2\ a_3]$ taking into account that the ends of both polygonal curves must match given that we respect the ordering of the points and we do not go backwards. The smallest of the maximum pairwise distances is the discrete Fréchet distance. In this example, the maxima $b_2\ a_3$ of both couplings is equal to two, therefore the minimum of both is also two.

E. HAUSDORFF

The Hausdorff distance is defined as the maximum distance of a set to the nearest point in the other set and was named

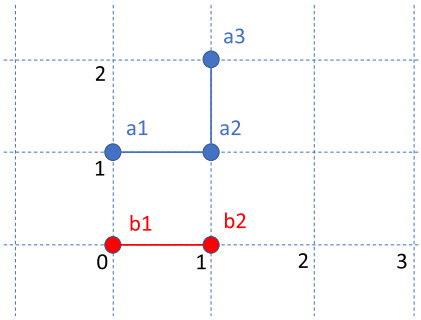


FIGURE 6. Example of the discrete Fréchet distance.

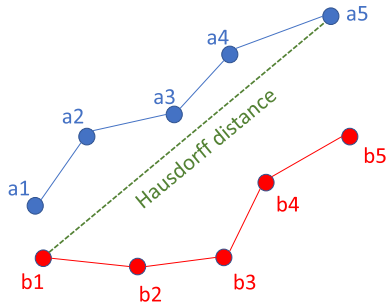


FIGURE 7. The Hausdorff distance $H(A, B)$ between A and B.

after Felix Hausdorff. Formally, let A and B be two sets of positional data points, $A = [a_1, a_2, \ldots, a_{n-1}, a_n]$ and $B = [b_1, b_2, \ldots, b_{m-1}, b_m]$, where n and m is the total number of points of A and B respectively. Then, the Hausdorff distance $h(A, B)$ is defined as:

$$h(A, B) = \max_{a \epsilon A}(\min_{b \epsilon B}(d(a, b))) \tag{7}$$

where d can be any distance metric between the points such as Euclidean or Haversine. Furthermore, the Hausdorff distance is directed which means that $h(A, B) \neq h(B, A)$. For instance, in Figure 7 we can observe that $h(A, B) = d(a_5, b_1)$ where b_1 is the nearest point of B to A and $d(a_5, b_1)$ is the maximum distance. Similarly, $h(B, A) = d(b_5, a_1)$ where a_1 is the nearest point of A to B and $d(b_5, a_1)$ is the maximum distance. Due to this asymmetry, a more general definition is given to the Hausdorff distance:

$$H(A, B) = max(h(A, B), h(B, A)) \tag{8}$$

where $H(A, B)$ is defined as the maximum distance between the two directed Hausdorff distances. Figure 7 illustrates the distance $H(A, B)$ between A and B. For the rest of the paper when referring to the Hausdorff distance, the distance of equation 8 is used.

F. SYMMETRIZED SEGMENT-PATH DISTANCE

The Symmetrized Segment-Path Distance (SSPD) is similar to the Hausdorff distance but the key difference is that instead of calculating the maximum distance of a set to the nearest point in the other set, the mean distance is used. Therefore, equation 7 can be rewritten as:

$$D_{SPD}(A, B) = mean(\min_{b \epsilon B}(d(a, b))) \tag{9}$$

where $D_{SPD}(A, B) \neq D_{SPD}(B, A)$. The SSPD is simply the mean of the two asymmetric Segment-Path Distances (SPD) and is defined as:

$$D_{SSPD}(A, B) = mean(D_{SPD}(A, B), D_{SPD}(B, A)) \tag{10}$$

More details on the SSPD can found in [55].

Table 2 summarizes the features of the trajectory similarity measures used in this work.

V. EVALUATION

This section presents the experimental evaluation of the examined compression algorithms. The performance evaluation is conducted in terms of compression ratio, execution speed and distance similarity. The underlying computing infrastructure has been a commodity machine with the following configuration: Ubuntu 18.04.4 LTS 64-bit; Intel Xeon E312xx @ 1.70GHz × 12; and 64 GiB RAM.

A. DATASET DESCRIPTION

The analysis of a global AIS dataset is challenging as it combines areas of very different message density due to the patterns that vessels follow, the system's technical characteristics and the means of collection (i.e. satellite or terrestrial network).

There is a lack of temporal and spatial uniformity in global AIS datasets affected by several factors; for example, in coastal areas the spatial and temporal distance between the collected positions is much smaller as opposed to open sea journeys where the lack of coverage can create much sparsely defined trajectories (e.g. a single position received in several hours). In this perspective it makes sense to focus on a representative dataset as the following which is a subset in the Mediterranean sea, rather than examining a very sparse dataset, e.g. in the Pacific or Atlantic ocean.

The dataset used in the following experiments contains AIS messages collected from a Terrestrial AIS receiver (T-AIS). The surveillance area covers the Saronic Gulf (Greece) and the vessels have been monitored for almost one and a half month period starting at February 18th, 2020 and ending at March 31th, 2020. The dataset provides information for 1229 unique vessels and contains 11, 769, 237 AIS records in total, each comprising 8 attributes as described in Table 3. A sample of the dataset used can be found in [56].

B. COMPRESSION EVALUATION

The compression ratio achieved by the different algorithms is presented in Figure 8. Top-down algorithms *DP* and *TR* depict the highest compression ratios. In top-down algorithms, a trajectory is recursively splitted until a halting condition is met. Specifically, *DP* presents a compression ratio of almost 96%.

TABLE 2. Trajectory similarity measures.

Measure	Meaning	Time complexity	Parameter free
DTW	Raw representation	$O(nm)$	✓
ERP	Raw representation	$O(nm)$	×
Fréchet	Shape	$O(nm\log(nm))$, where n and m are number of edges in two curves	✓
Discrete Fréchet	Shape	$O(n^m)$	✓
Hausdorff	Shape	$O(k\log k)$, where k is the total number of vertices	✓
SSPD	Shape	$O(k\log k)$, where k is the total number of vertices	✓

TABLE 3. Dataset attributes.

Feature	Description
timestamp	the time at which the message was received (UTC)
ship_id	unique identifier for each ship
lon	the longitude of the current ship position
lat	the latitude of the current ship position
ship_type	AIS reported ship-type
speed	Speed over ground in knots
course	Course over ground in degrees with 0 corresponding to north
heading	Ship's heading in degrees with 0 corresponding to north

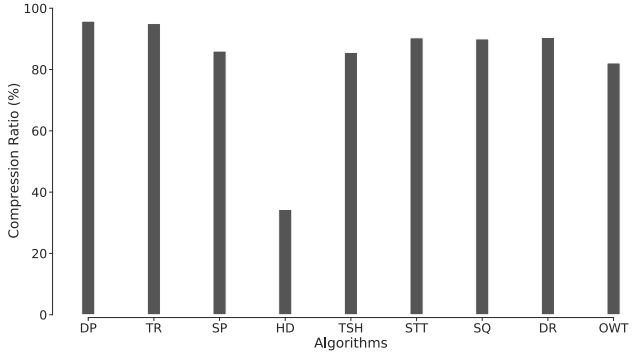


FIGURE 8. Compression ratio achieved by different algorithms.

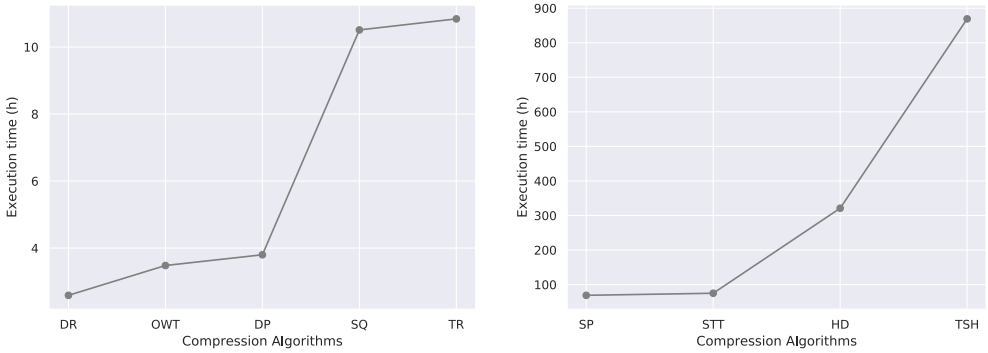
Compression is slightly lower in *TR* but nevertheless remains extremely high at about 95%. Algorithms that consider time information to decide whether or not to discard a point, tend to preserve more points in a trajectory. As *DP* treats a trajectory as a line in two-dimensional space and does not consider the temporal aspect, it possess the highest compression. On the other hand, opening-window algorithms *SP* and *TSH* preserve more points in the resulting set than the aforementioned algorithms, with a compression ratio of 10% less. Compression ratio is decreased in case of online algorithms *STT*, *SQ* and *DR* which employ SED as error criterion (*DR* also exploits speed), but maintained at a high level, about 90%. *OWT* which utilizes TRD is slightly behind by 8% from the other online algorithms with a compression of 82%. *HD* presents the lowest compression ratio of 34%, almost three times smaller than *DP* and *TR*. This practically means, the majority of trajectories present significant heading variations in some subsegments that exceed the average heading threshold variation of the trajectory as a whole. Thus, *HD* algorithm retains these "significant" points where the heading variations occur.

Besides the compression ratio, it is important to assess the execution time of each algorithm as it constitutes an important index to measure efficiency and can serve as a decision element for selecting an algorithm among others. As illustrated in Figure 9, the execution times (measured in hours) of the examined algorithms are presented in two groups, based on the exhibition time scales. Indeed, the first group's execution times are maintained low with a maximum of 11 hours, while in the second group the measurements range from 69 to over 800 hours. As shown in Figure 9(a), *DR* presents the lowest execution time among all algorithms followed by *OWT* and *DP*. The fairly low execution time of *DR* is justified by its low complexity. The execution times of *OWT* and *DP* are slightly higher, but the advantage of *DP* compared to *DR* is that the former reaches around 6% more compression than the latter. On the other hand, the compression ratio of *OWT* is 8% less in comparison with *DR*. *SQ* and *TR* spent considerable more time than the other algorithms belonging to the first group. Specifically, *TR* is 4 times slower than *DR* but reaches around 5% more compression. Additionally, when compared to *OWT* the compression ratio difference of *TR* is even higher, about 13% but accompanied with 3 times slower execution of the latter. *SQ* exhibits almost the same execution time with *TR* but presents almost the same compression ratio as *DR*. *DP* holds the highest compression ratio (96%) and is accompanied by a fairly low execution time, only 1.4 times slower than *DR*. Nevertheless, at this point it is worth noting that online algorithms have the advantage of executing in online mode.

In the second group, as shown in Figure 9(b), the execution times of *SP* and *STT* differ by 6 hours with the latter to present 4% more compression. The execution time of *HD* is 4 times larger than *STT*, with a compression ratio of only 34%. *TSH* exhibits an extremely high execution time due to its exponential complexity, 869 hours. This means that this algorithm is 2.7 times slower than *HD* and 335 times slower than *DR* which presents the lowest execution time. The compression ratio of *TSH* is 2.5 times better than *HD*, but lacks only 5% when compared to *DR*. Only *STT* from the online algorithms has a relatively high execution time. In fact, the execution times of *DR*, *OWT* and *SQ* remain low, except for the *OWT* which slightly increases. On the other hand, all the offline algorithms depict high execution times, extremely high in some cases (*HD*, *TSH*) except for the *DP* algorithm.

C. DISTANCE EVALUATION

To evaluate the amount of information loss that is posed by each compression algorithm, trajectory similarity measures



(a) First group (b) Second group

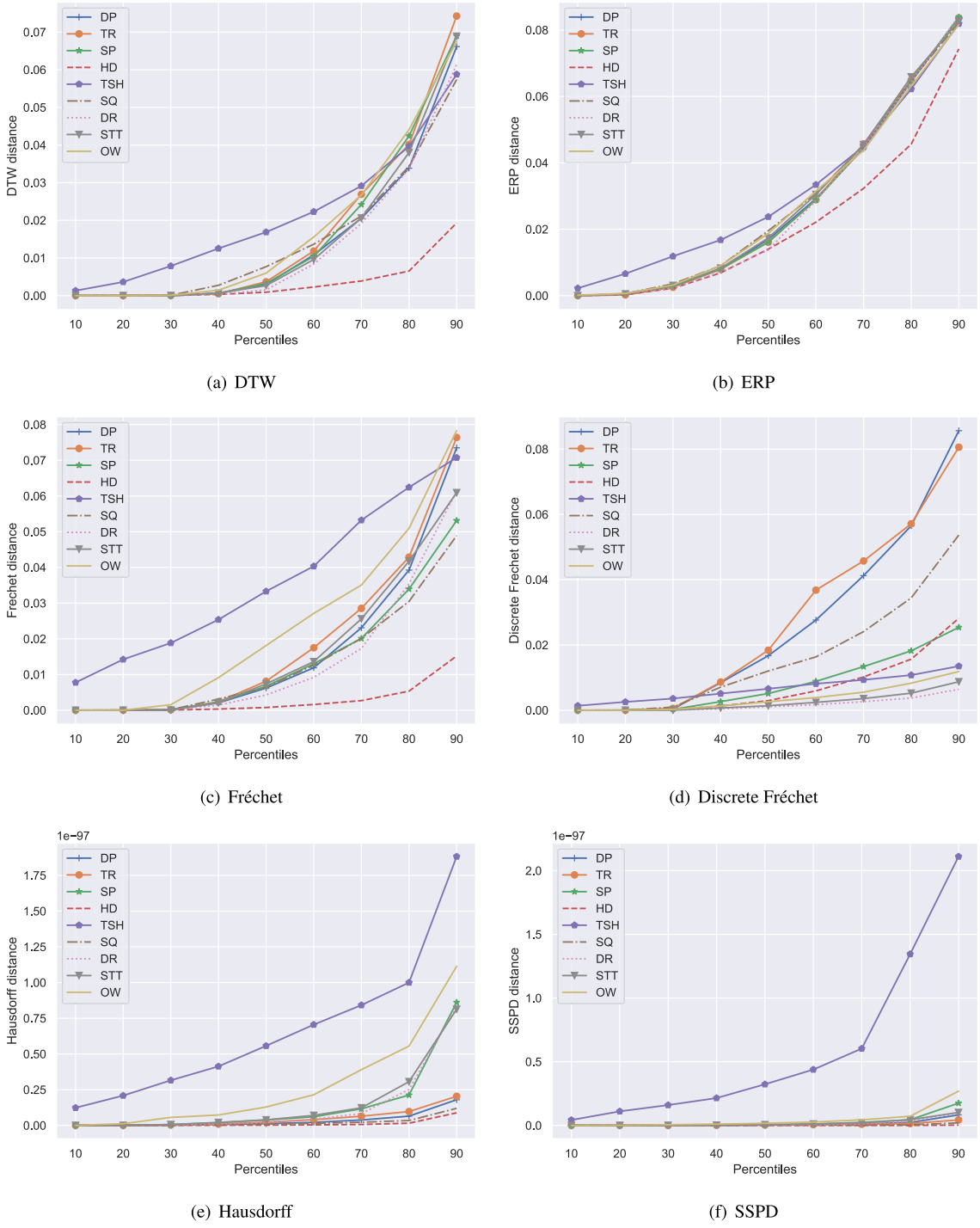
FIGURE 9. Execution times of different groups.

TABLE 4. Distance results.

Similarity Measures		Compression Algorithms								
		DP	TR	SP	HD	TSH	SQ	DR	STT	OWT
DTW	μ	0.0217	0.0231	0.0224	**0.0121**	0.0279	0.0222	0.021	0.0218	0.0237
	σ	0.0381	0.037	0.0377	0.0371	0.0376	0.0378	0.0385	0.0381	0.0369
ERP	μ	0.03	0.0299	0.03	**0.0268**	0.0343	0.0305	0.0294	0.0301	0.0302
	σ	0.032	0.0321	0.032	0.0348	0.0318	0.0315	0.0326	0.0319	0.0318
Frechet	μ	0.0232	0.0248	0.0203	**0.0074**	0.0382	0.0196	0.0205	0.0226	0.0283
	σ	0.0372	0.0362	0.0389	0.0332	0.027	0.0392	0.0388	0.0376	0.0335
discret Frechet	μ	0.0291	0.0299	0.0116	0.0117	0.0115	0.0221	**0.0054**	0.0055	0.0069
	σ	0.0328	0.0321	0.0423	0.0423	0.0453	0.0379	0.0436	0.0435	0.0433
Hausdorff	μ	0.0128	0.0128	0.0128	**0.0125**	0.0136	0.0128	0.0129	0.0128	0.0169
	σ	0.042	0.042	0.042	0.042	0.0447	0.042	0.0419	0.042	0.0405
SSPD	μ	0.0128	0.0128	0.0128	**0.0125**	0.0136	0.0128	0.0129	0.0128	0.0161
	σ	0.042	0.042	0.042	0.042	0.0447	0.042	0.0419	0.042	0.0408

were chosen (see Section IV) that are able to measure the distance between the original trajectory and the compressed one in terms of shape. The shape of the trajectory and the pattern that it forms can play an important role in the identification of illegal activities [3], [17]. Therefore, for each compression algorithm, we measured the distances D between each trajectory and its compressed counterpart and calculated the mean distance and the standard deviation of distance for the entirety of the dataset. Furthermore, the trajectory distance measures do not calculate the distance in the range of 0 to 1 and for that reason we normalized the results.

Table 4 illustrates the normalized distance results for each compression algorithm and for each trajectory distance measure. It can be seen that the *HD* compression algorithm presents the lowest distance in most of the cases and therefore the highest similarity with its uncompressed counterpart. Furthermore, the *TSH* compression algorithm which is a combination of *TR*, *SP* and *HD* yields the highest distance in most of the cases. This can be explained by the fact that it filters out data points based on both the speed and the heading, thus the amount of points kept is greatly reduced and the distance to the original trajectory is increased. The rest of the compression algorithms present similar distances in the majority of the similarity measures. Moreover, it is worth noting that the Hausdorff distance and the SSPD yield similar results due to the similarity of these two distance metrics (see Section IV). These results can be further verified by the percentiles from 10 to 90 percent of each distance measure and compression algorithm that are illustrated in Figure 10. Each percentile denotes the percentage of the dataset that has a distance value less than or equal to the illustrated value. The x axis of all subfigures of Figure 10 represents the percentile of the dataset, the y axis represents the respective distance value and each line represents a different compression algorithm. Similar to Table 4 it can be seen that the *HD* algorithm has the lowest distance in most of the percentiles and distance measures, while the *TSH* algorithm has the highest distances in most of the cases. Finally, despite the fact that all compression algorithms present an upward trend as the percentage of the dataset increases, it can be observed that all algorithms except *HD* present a steeper curve in most distance measures. This denotes that the *HD* compression algorithm is less volatile and it is an indicator that it is more suitable for datasets in the maritime domain. This is further confirmed by research conducted in [17] where the proposed trajectory classification algorithm achieves a better classification performance when the *HD* is used in trajectories compared to the rest of the algorithms.



(a) DTW (b) ERP

(c) Fréchet (d) Discrete Fréchet

(e) Hausdorff (f) SSPD

FIGURE 10. **Distance percentiles of compression algorithms.**

D. DISCUSSION

Table 5 presents a comparison of the different algorithms in terms of compression ratio, execution time and similarity score. The algorithms are represented in descending order regarding their performance in each aspect with the top algorithm demonstrating the best performance and the bottom algorithm representing the worst performance. The results demonstrate high compression ratios for *DP*, *TR* and *DR*. *DR* is considered the best algorithm, because it is one of the three best algorithms in all evaluating criteria. *DP* is also one of the best three algorithms in two of the three evaluating criteria. However, it presents the worst similarity score along with *TR* and *TSH*. The high compression ratio of *TR* is accompanied with a moderate execution time and a low similarity score.

TABLE 5. Compression algorithms comparison.

Position	Compression Ratio	Execution Time	Similarity Score
1^{st}	DP	DR	HD
2^{nd}	TR	OWT	DR
3^{rd}	DR	DP	STT
4^{th}	STT	SQ	SP
5^{th}	SQ	TR	SQ
6^{th}	SP	SP	OWT
7^{th}	TSH	STT	DP
8^{th}	OWT	HD	TR
9^{th}	HD	TSH	TSH

SQ and *SP* are neither in the best or worst results. They present a moderate performance in all evaluating criteria. *STT* presents a moderate value regarding compression ratio but achieves one of the best similarity scores. Nevertheless, the execution time is among the three worst. *STT* and *SQ* are the only two algorithms that use a memory size-based compression approach. *OWT* belongs to the best three algorithms regarding execution time but presents a moderate similarity score and one of the worst compression ratios. *STT* and *OWT* compete for the third place among the best algorithms with *STT* to present slightly better results. In general terms, *TSH* exhibit the worst results in the sense that present the smaller accuracy for all the evaluation criteria. In fact, its execution time is extremely high in comparison with the other algorithms. *HD* presents the highest similarity score but on the other hand the compression ratio is significantly lower in comparison with the other algorithms and also the execution time is among the lowest.

Finally, it is worth noting that all of the compression algorithms require user defined parameters. In the set of experiments proposed in this work, the parameters were simply an average value of either speed or heading, depending on the algorithm. Despite the fact that this technique of parameter selection is suitable for compression algorithms comparison, the parameters of its compression algorithm should dynamically change for the same trajectory in real-world settings. The reason for this is that vessels exhibit different behavior such as speed and movement in different geographic areas, thus a global static parameter would yield erroneous results in large trajectories that pass through a variety of water areas. Therefore, a recommended solution for either offline or online compression algorithms is to segment the trajectory in temporal sliding windows and use a different parameter for each window.

VI. CONCLUSION

In this work, we presented a set of well-established trajectory compression algorithms. In particular, nine compression algorithms were compared against different defined thresholds. The dynamic determination of threshold values eliminates the need of arbitrary user-defined thresholds. Instead, a threshold is applied in each trajectory, based on its actual features and its peculiarities. Furthermore, we presented the distance metrics that are suitable to best evaluate their performance and run experiments on a maritime dataset that contains AIS messages. In the experimental evaluation all compression algorithms were evaluated on three aspects: execution time, compression ratio and distance between the compressed trajectories and their uncompressed counterparts. Experiments demonstrated that the best performance of offline and online compression algorithms was achieved by *DP*, *SP* and *DR*, *STT* respectively. However, considering all assessed aspects, *DR* presented the most suitable performance in general. Undoubtedly, online algorithms have the advantage of executing in online mode.

The results suggest that there is a trade-off between the compression ratio and the quality achieved. Choosing a proper compression algorithm is not an easy task as the selection is application-dependent. Different application scenarios may require different trajectory characteristics preservation. Nevertheless, this research work can provide some insights of choosing and handling trajectory compression algorithms over AIS data.

ACKNOWLEDGMENT

The work reflects only the authors' view and that the EU Agency is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] I. Kontopoulos, G. Spiliopoulos, D. Zissis, K. Chatzikokolakis, and A. Artikis, "Countering real-time stream poisoning: An architecture for detecting vessel spoofing in streams of AIS data," in *Proc. IEEE 16th Int. Conf. Dependable, Autonomic Secure Comput., 16th Int. Conf. Pervas. Intell. Comput., 4th Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Athens, Greece, Aug. 2018, pp. 981–986.

[2] I. Kontopoulos, K. Chatzikokolakis, D. Zissis, K. Tserpes, and G. Spiliopoulos, "Real-time maritime anomaly detection: Detecting intentional AIS switch-off," *Int. J. Big Data Intell.*, vol. 7, no. 2, pp. 85–96, 2020.

[3] I. Kontopoulos, K. Chatzikokolakis, K. Tserpes, and D. Zissis, "Classification of vessel activity in streaming data," in *Proc. 14th ACM Int. Conf. Distrib. Event-based Syst.*, J. Gascon-Samson, K. Zhang, K. Daudjee, and B. Kemme, Eds., Montreal, QC, Canada, Jul. 2020, pp. 153–164.

[4] G. Pallotta, M. Vespe, and K. Bryan, "Vessel pattern knowledge discovery from AIS data: A framework for anomaly detection and route prediction," *Entropy*, vol. 15, no. 12, pp. 2218–2245, Jun. 2013.

[5] D. Zissis, K. Chatzikokolakis, G. Spiliopoulos, and M. Vodas, "A distributed spatial method for modeling maritime routes," *IEEE Access*, vol. 8, pp. 47556–47568, 2020.

[6] I. Kontopoulos, I. Varlamis, and K. Tserpes, "A distributed framework for extracting maritime traffic patterns," *Int. J. Geographical Inf. Sci.*, pp. 1–26, 2020.

[7] G. Langran, "Issues of implementing a spatiotemporal system," *Int. J. Geographical Inf. Syst.*, vol. 7, no. 4, pp. 305–314, Jul. 1993.

[8] A. Makris, K. Tserpes, D. Anagnostopoulos, M. Nikolaidou, and J. A. F. de Macedo, "Database system comparison based on spatiotemporal functionality," in *Proc. 23rd Int. Database Appl. Eng. Symp. (IDEAS)*, 2019, pp. 1–7.

[9] A. Makris, K. Tserpes, G. Spiliopoulos, and D. Anagnostopoulos, "Performance evaluation of MongoDB and postgresql for spatio-temporal data," in *Proc. EDBT/ICDT Workshops*, 2019.

[10] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis, and D. Anagnostopoulos, "MongoDB Vs PostgreSQL: A comparative study on performance aspects," *GeoInformatica*, vol. 25, pp. 243–268, 2021.

[11] M. Potamias, K. Patroumpas, and T. Sellis, "Sampling trajectory streams with spatiotemporal criteria," in *Proc. 18th Int. Conf. Sci. Stat. Database Manage. (SSDBM)*, 2006, pp. 275–284.

[12] A. Makris, C. L. D. Silva, V. Bogorny, L. O. Alvares, J. A. Macedo, and K. Tserpes, "Evaluating the effect of compressing algorithms for trajectory similarity and classification problems," *GeoInformatica*, pp. 1–33, May 2021.

[13] N. Meratnia and A. Rolf, "Spatiotemporal compression techniques for moving point objects," in *Proc. Int. Conf. Extending Database Technol.* Cham, Switzerland: Springer, 2004, pp. 765–782.

[14] J. Muckell, J.-H. Hwang, V. Patil, C. T. Lawson, F. Ping, and S. S. Ravi, "SQUISH: An online approach for GPS trajectory compression," in *Proc. 2nd Int. Conf. Comput. Geospatial Res. Appl. (COM.Geo)*, 2011, pp. 1–8.

[15] J. Muckell, J.-H. Hwang, C. T. Lawson, and S. S. Ravi, "Algorithms for compressing GPS trajectory data: An empirical evaluation," in *Proc. 18th SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst. (GIS)*, 2010, pp. 402–405.

[16] J. Muckell, P. W. Olsen, J.-H. Hwang, C. T. Lawson, and S. S. Ravi, "Compression of trajectory data: A comprehensive evaluation and new approach," *GeoInformatica*, vol. 18, no. 3, pp. 435–460, Jul. 2014.

[17] I. Kontopoulos, A. Makris, and K. Tserpes, "A deep learning streaming methodology for trajectory classification," *ISPRS Int. J. Geo-Inf.*, vol. 10, no. 4, p. 250, Apr. 2021.

[18] N. Meratnia and R. A. D. By, "A new perspective on trajectory compression techniques," in *Proc. ISPRS Commission II IV, WG II/5, II/6, IV/1 IV/2 Joint Workshop Spatial, Temporal Multi-Dimensional Data Modeling Anal.*, 2003.

[19] Y. E. Leichsenring and F. Baldo, "An evaluation of compression algorithms applied to moving object trajectories," *Int. J. Geograph. Inf. Sci.*, vol. 34, no. 3, pp. 539–558, 2020.

[20] M. Chen, M. Xu, and P. Franti, "Compression of GPS trajectories," in *Proc. Data Compress. Conf.*, Apr. 2012, pp. 62–71.

[21] J. Birnbaum, H.-C. Meng, J.-H. Hwang, and C. Lawson, "Similarity-based compression of GPS trajectory data," in *Proc. 4th Int. Conf. Comput. Geospatial Res. Appl.*, Jul. 2013, pp. 92–95.

[22] P. Cudre-Mauroux, E. Wu, and S. Madden, "TrajStore: An adaptive storage system for very large trajectory data sets," in *Proc. IEEE 26th Int. Conf. Data Eng. (ICDE)*, 2010, pp. 109–120.

[23] Y. Zhao, S. Shang, Y. Wang, B. Zheng, Q. V. H. Nguyen, and K. Zheng, "REST: A reference-based framework for spatio-temporal trajectory compression," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2797–2806.

[24] J. Liu, H. Li, Z. Yang, K. Wu, Y. Liu, and R. W. Liu, "Adaptive douglas-peucker algorithm with automatic thresholding for AIS-based vessel trajectory compression," *IEEE Access*, vol. 7, pp. 150677–150692, 2019.

[25] Z. Wei, X. Xie, and X. Zhang, "AIS trajectory simplification algorithm considering ship behaviours," *Ocean Eng.*, vol. 216, Nov. 2020, Art. no. 108086.

[26] S.-K. Zhang, Z.-J. Liu, Y. Cai, Z.-L. Wu, and G.-Y. Shi, "AIS trajectories simplification and threshold determination," *J. Navigat.*, vol. 69, no. 4, pp. 729–744, Jul. 2016.

[27] L. Zhao and G. Shi, "A method for simplifying ship trajectory based on improved Douglas–Peucker algorithm," *Ocean Eng.*, vol. 166, pp. 37–46, Oct. 2018.

[28] L. Zhao and G. Shi, "A trajectory clustering method based on douglas-peucker compression and density for marine traffic pattern recognition," *Ocean Eng.*, vol. 172, pp. 456–467, Jan. 2019.

[29] L. Etienne, T. Devogele, and A. Bouju, "Spatio-temporal trajectory analysis of mobile objects following the same itinerary," *Adv. Geo-Spatial Inf. Sci.*, vol. 10, pp. 47–57, Jul. 2012.

[30] G. K. D. de Vries and M. van Someren, "Machine learning for vessel trajectories using compression, alignments and domain knowledge," *Expert Syst. Appl.*, vol. 39, no. 18, pp. 13426–13439, Dec. 2012.

[31] Y. Li, R. W. Liu, J. Liu, Y. Huang, B. Hu, and K. Wang, "Trajectory compression-guided visualization of spatio-temporal AIS vessel density," in *Proc. 8th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2016, pp. 1–5.

[32] Y. Huang, Y. Li, Z. Zhang, and R. W. Liu, "GPU-accelerated compression and visualization of large-scale vessel trajectories in maritime IoT industries," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10794–10812, Nov. 2020.

[33] G. Fikioris, K. Patroumpas, A. Artikis, G. Paliouras, and M. Pitsikalis, "Fine-tuned compressed representations of vessel trajectories," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 2429–2436.

[34] S. Sun, Y. Chen, Z. Piao, and J. Zhang, "Vessel AIS trajectory online compression based on scan-pick-move algorithm added sliding window," *IEEE Access*, vol. 8, pp. 109350–109359, 2020.

[35] P. Sun, S. Xia, G. Yuan, and D. Li, "An overview of moving object trajectory compression algorithms," *Math. Problems Eng.*, vol. 2016, pp. 1–13, Apr. 2016.

[36] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, p. 38, 2014.

[37] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica, Int. J. Geographic Inf. Geovisualization*, vol. 10, no. 2, pp. 112–122, Dec. 1973.

[38] J. E. Hershberger and J. Snoeyink, "Speeding up Douglas–Peucker line-simplification algorithm," Dept. Comput. Sci., Univ. British Columbia, Vancouver, BC, Canada, Tech. Rep., 1992.

[39] G. Trajcevski, H. Cao, P. Scheuermanny, O. Wolfsonz, and D. Vaccaro, "On-line data reduction and the quality of history in moving objects databases," in *Proc. 5th ACM Int. workshop Data Eng. Wireless Mobile Access (MobiDE)*, 2006, pp. 19–26.

[40] S.-K. Zhang, G.-Y. Shi, Z.-J. Liu, Z.-W. Zhao, and Z.-L. Wu, "Data-driven based automatic maritime routing from massive AIS trajectories in the face of disparity," *Ocean Eng.*, vol. 155, pp. 240–250, May 2018.

[41] H. Rong, A. P. Teixeira, and C. Guedes Soares, "Data mining approach to shipping route characterization and anomaly detection based on AIS data," *Ocean Eng.*, vol. 198, Feb. 2020, Art. no. 106936.

[42] H. Wang, H. Su, K. Zheng, S. W. Sadiq, and X. Zhou, "An effectiveness study on trajectory similarity measures," in *Proc. 24th Australas. Database Conf. (ADC)*, vol. 137, H. Wang and R. Zhang, Eds. Adelaide, SA, Australia: Australian Computer Society, Feb. 2013, pp. 13–22.

[43] K. Toohey and M. Duckham, "Trajectory similarity measures," *SIGSPATIAL Special*, vol. 7, no. 1, pp. 43–50, May 2015.

[44] P. Ranacher and K. Tzavella, "How to compare movement? A review of physical movement similarity measures in geographic information science and beyond," *Cartography Geographic Inf. Sci.*, vol. 41, no. 3, pp. 286–307, May 2014.

[45] N. Magdy, M. A. Sakr, T. Mostafa, and K. El-Bahnasy, "Review on trajectory similarity measures," in *Proc. IEEE 7th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2015, pp. 613–619.

[46] R. S. D. Sousa, A. Boukerche, and A. A. F. Loureiro, "Vehicle trajectory similarity: Models, methods, and applications," *ACM Comput. Surv.*, vol. 53, no. 5, pp. 1–32, Oct. 2020.

[47] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng, "A survey of trajectory distance measures and performance evaluation," *VLDB J.*, vol. 29, no. 1, pp. 3–32, Jan. 2020.

[48] A. S. Furtado, L. O. C. Alvares, N. Pelekis, Y. Theodoridis, and V. Bogorny, "Unveiling movement uncertainty for robust trajectory similarity analysis," *Int. J. Geographical Inf. Sci.*, vol. 32, no. 1, pp. 140–168, Jan. 2018.

[49] M. Liang, R. W. Liu, S. Li, Z. Xiao, X. Liu, and F. Lu, "An unsupervised learning method with convolutional auto-encoder for vessel trajectory similarity computation," *Ocean Eng.*, vol. 225, Apr. 2021, Art. no. 108803.

[50] N. Magdy, M. A. Sakr, T. Mostafa, and K. El-Bahnasy, "Review on trajectory similarity measures," in *Proc. IEEE 7th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2015, pp. 613–619.

[51] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proc. 18th Int. Conf. Data Eng.*, San Jose, CA, USA, Feb./Mar. 2002, pp. 673–684.

[52] L. Chen and T. R. Ng, "On the marriage of Lp-norms and edit distance," in *Proc. 30th Int. Conf. Very Large Data Bases (VLDB)*, M. A. Nascimento, M. Tamer Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, Eds. Toronto, ON, Canada: Morgan Kaufmann, Aug./Sep. 2004, pp. 792–803.

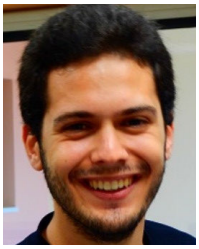
[53] J. D. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. Knowledge Discovery Databases, Papers AAAI Workshop*, U. M. Fayyad and R. Uthurusamy, Eds. Seattle, WA, USA: AAAI Press, Jul. 1994, pp. 359–370.

[54] H. Alt and M. Godau, "Computing the Fréchet distance between two polygonal curves," *Int. J. Comput. Geometry Appl.*, vol. 5, pp. 75–91, Mar. 1995.

[55] P. C. Besse, B. Guillouet, J.-M. Loubes, and F. Royer, "Review and perspective for distance-based clustering of vehicle trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3306–3317, Nov. 2016.

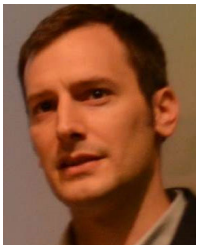
[56] I. Kontopoulos, M. Vodas, G. Spiliopoulos, K. Tserpes, and D. Zissis, "Single ground based AIS receiver vessel tracking dataset," Tech. Rep., Apr. 2020.

ANTONIOS MAKRIS received the B.Sc. degree from the Department of Informatics and Telematics, Harokopio University of Athens, and the M.Sc. degree in the area of web engineering. He is currently pursuing the Ph.D. degree with the Department of Informatics and Telematics, Harokopio University of Athens. He has participated in numerous EU funded and national projects, including Fortissimo (FP7), Asylum (Ministry of Immigration Policy), Productive 4.0 (ECSEL), MASTER and SmartShip (H2020), GLASSEAS (OP Human Resources Development, Education and Lifelong Learning), and CHARITY (H2020). His main research interests include distributed computing, big data management and analysis, NoSQL database systems, high performance computing (HPC), spatiotemporal and trajectory analysis, and machine/deep learning.



IOANNIS KONTOPOULOS received the B.Sc. degree from the Department of Informatics and Telematics, Harokopio University of Athens, in 2016. He is currently pursuing the Ph.D. degree with the Department of Informatics and Telematics, Harokopio University of Athens, and the Research Team of MarineTraffic. He has been involved in several EU and national funded projects, including Datacron (H2020), INFORE (H2020), MASTER (H2020-MSCA-RISE-2020), Smartship (H2020-MSCA-RISE-2020), and GLASSEAS (OP Human Resources Development, Education and Lifelong Learning). His major research interests include distributed systems, big data analysis and processing, real-time stream processing, spatio-temporal and trajectory analysis, and machine learning.

PANAGIOTIS ALIMISIS is currently pursuing the bachelor's degree with the Department of Informatics and Telematics, Harokopio University of Athens. He has participated in one EU funded research program, MASTER (H2020). His main research interests include full stack development and machine learning.



KONSTANTINOS TSERPES received the Ph.D. degree in the area of distributed systems from the School of Electrical and Computer Engineering, National Technical University of Athens, in 2008. He is currently an Assistant Professor with the Department of Informatics and Telematics, Harokopio University of Athens. He has been involved in several EU and National funded projects leading research for solving issues related to scalability, interoperability, fault tolerance, and extensibility in application domains, such as multimedia, e-governance, post-production, finance, and e-health. His research interests include distributed systems, software and service engineering, big data analytics, and social systems. He is a member of the Editorial Board of *Future Generation Computer Systems*.