IEEE Open Journal of Vehicular Technology

Received 21 May 2024; accepted 4 August 2024. Date of publication 14 August 2024;
date of current version 13 September 2024. The review of this article was coordinated by Editor Beatriz Soret.

Digital Object Identifier 10.1109/OJVT.2024.3443675

AIS-Based Vessel Trajectory Compression: A Systematic Review and Software Development

RYAN WEN LIU [1,2,3] (Member, IEEE), SHIQI ZHOU [1,3], SHANGKUN YIN [1,3], YAQING SHU [1,3] (Member, IEEE), AND MAOHAN LIANG [4]

[1]Hubei Key Laboratory of Inland Shipping Technology, School of Navigation, Wuhan University of Technology, Wuhan 430063, China
[2]Hainan Institute, Wuhan University of Technology, Sanya 572000, China
[3]State Key Laboratory of Maritime Technology and Safety, Wuhan University of Technology, Wuhan 430063, China
[4]Department of Civil and Environmental Engineering, National University of Singapore, Singapore 117576

CORRESPONDING AUTHOR: MAOHAN LIANG (e-mail: mhliang@nus.edu.sg).

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFC3302702 and in part by the Excellent Youth Foundation of Hubei Scientific Committee under Grant 2024AFA042.
The source code related to this work is available at https://github.com/WHUT-MIPC/AISCompress.

ABSTRACT With the advancement of satellite and 5G communication technologies, vehicles can transmit and exchange data from anywhere in the world. It has resulted in the generation of massive spatial trajectories, particularly from the Automatic Identification System (AIS) for surface vehicles. The massive AIS data lead to high storage requirements and computing costs, as well as low data transmission efficiency. These challenges highlight the critical importance of vessel trajectory compression for surface vehicles. However, the complexity and diversity of vessel trajectories and behaviors make trajectory compression imperative and challenging in maritime applications. Therefore, trajectory compression has been one of the hot spots in research on trajectory data mining. The major purpose of this work is to provide a comprehensive reference source for beginners involved in vessel trajectory compression. The current trajectory compression methods could be broadly divided into two types, batch (offline) and online modes. The principles and pseudo-codes of these methods will be provided and discussed in detail. In addition, compressive experiments on several publicly available data sets have been implemented to evaluate the batch and online compression methods in terms of computation time, compression ratio, trajectory similarity, and trajectory length loss rate. Finally, we develop a flexible and open software, called *AISCompress*, for AIS-based batch and online vessel trajectory compression. The conclusions and associated future works are also given to inspire future applications in vessel trajectory compression.

INDEX TERMS Automatic identification system (AIS), vessel trajectory, trajectory compression, error metrics, similarity measure.

I. INTRODUCTION

Spatial trajectories are of significance for improving vessel's situation awareness and enhancing maritime safety, etc [1]. In the maritime field, spatial trajectory computation is indispensable for surface vehicle trajectory clustering [2], [3], [4], trajectory prediction [5], [6], [7], vessel abnormal behavior detection [8], [9], [10], [11], [12], etc. Efficient and accurate trajectory computing can support these related studies to enhance performance, enabling intelligent maritime transportation.

In the maritime sector, maritime transportation plays a crucial role in global trade and commerce, carrying approximately 90 % of global trade [13], [14]. The vast marine traffic makes the traffic environment complex and brings a high potential for traffic accidents, which will cause serious casualties, property losses, and environmental pollution [15], [16]. For example, the traffic volume on the channel in busy ports can reach 2,000 times a day, which means there are frequent conflicts between sailing vessels. [17]. To mitigate the risks and reduce the accident rate, the initial approach was to adopt Very High Frequency (VHF) communication for surface vehicle communication [18]. However, issues with equipment operation and language barriers usually result in

miscommunication and misunderstanding between vessels, leading to potential collisions. To address these challenges and enhance the timeliness and effectiveness of vessel communications, the International Maritime Organization (IMO) mandated the installation of the Automatic Identification System (AIS) on specific vessels [19], [20]. This requirement applies to vessels with a gross tonnage of over 300 before December 31, 2004, and non-international vessels with a gross tonnage of over 500 before July 1, 2008 [21]. AIS technology enables vessels to transmit and receive important information allowing for improved situational awareness and better coordination between vessels [22], [23]. Specifically, the information transmitted by the AIS system can be categorized into static and dynamic information. Static information includes details such as MMSI Number, IMO Code, Call Sign, Vessel's Name, Dimensions, Type of Vessel, Destination, and ETA (Estimated Time of Arrival). On the other hand, dynamic information consists of real-time data including Vessel Position, Coordination Universal Time, Course over Ground (COG), Speed over Ground (SOG), Heading, Navigational Status, and other similar state information [24], [25]. The availability of AIS data has significantly expanded the possibilities for studying and addressing issues within the maritime domain, leading to improved safety, efficiency, and environmental protection in the shipping industry [26], [27], [28].

AIS data plays a crucial role in supporting research within the maritime field [29]. With the development of satellite and 5G communication technology, the data transmission rate that can be supported on the ocean is gradually reaching a high level, to enable efficient inter-vessel communications [30], [31]. However, the high frequency of vessel AIS data updates, coupled with the increasing number of vessels, contributes to the significant size of vessel AIS trajectory data [32], [33]. The frequent updates impose greater demands on system management and computing performance [34], [35], [36]. Typically, the AIS information is updated every 2 seconds to 6 minutes during voyages, resulting in the accumulation of massive AIS data [37]. The accumulation of AIS data presents challenges in terms of data storage, especially for marine bureaus and shipping companies. Importing large quantities of AIS trace data into the Electronic Chart Display and Information System (ECDIS) platform further exacerbates inefficiencies. Consequently, the response speed for electronic chart zooming and panning is significantly diminished, thereby compromising the timeliness of vessel trajectory calculations. These issues necessitate efficient system management and computing solutions to optimize data storage techniques and enhance the performance of the ECDIS platform [38]. Numerous advanced methods have been employed to effectively process and analyze AIS data.

Both batch and online compression algorithms utilize line segments to approximate and substitute the original trajectory, resulting in a significant reduction in the number of intermediate points. They all operate within the Euclidean space and measure errors based on Euclidean distance, but the key distinction lies in the scope of compression. The batch compression algorithm considers the entire trajectory, aiming for a globally optimal compressed trajectory. In terms of efficiency, online compression outperforms batch compression in compressing AIS data. This is particularly valuable for real-time monitoring of vessel trajectories. Fig. 1 shows the projection of the vessel AIS data on the map at a certain moment, expressed in the form of points. Each point corresponds to the AIS data of a vessel, reflecting its real-time position status.

Nowadays, scholars have been actively conducting research on vessel AIS data compression in terms of theory and application [39], [40]. One of the pioneering methods is the popular Douglas-Peucker (DP) algorithm, initially proposed by Douglas and Peucker [41]. This algorithm employs an iterative approach to measure the amount of information loss and selectively removes points that contribute less to compression quality. Building upon this work, Meratnia et al. [42] introduced the time-scaled Top-Down Time-Ratio (TD-TR) algorithm, a batch compression technique based on offset. TD-TR improves upon the DP algorithm by utilizing synchronous Euclidean distance as a replacement for vertical distance, resulting in a substantial enhancement in compression performance. This approach has served as an inspiration for subsequent trajectory data compression algorithms. Furthermore, Liu et al. [43] proposed the Adaptive Douglas-Peucker (ADP) algorithm, which incorporated automatic threshold based on channel and trajectory features. This algorithm employs a segmentation framework and average distance development, incorporating a novel automatic threshold selection method for each trajectory. Serving as an improvement and complement to the original DP algorithm, ADP adaptively calculates the optimal threshold for the individual trajectory. Moreover, Li et al. [44] proposed a novel trajectory compression algorithm called ADPS, which sets a different threshold for each trajectory based on geometric features, speed variation rate, and the distance between feature points. This unsupervised approach offers promising advancements in trajectory compression. The spatial and motion features of trajectories have been jointly considered to compress the vessel trajectories while maintaining the important characteristic elements [45]. To improve the compression efficiency, Zhao et al. [46] proposed an improved DP algorithm, which considered the shapes of vessel trajectories, to obviously shorten the computational time and generate satisfactory compression performance. To overcome the negative effects of experience-based threshold selection for the traditional DP algorithm, an adaptive-threshold DP algorithm was presented by automatically determining the key points of each vessel trajectory [47]. In addition, Yan et al. [48] proposed to denoise and compress raw AIS-based vessel trajectories. This method has the capacity to improve the trajectory quality while reducing the data volume. Many efforts have also been made to investigate the influences of trajectory compression on trajectory clustering [49] and modeling carbon dioxide emissions [50]. These aforementioned algorithms have played

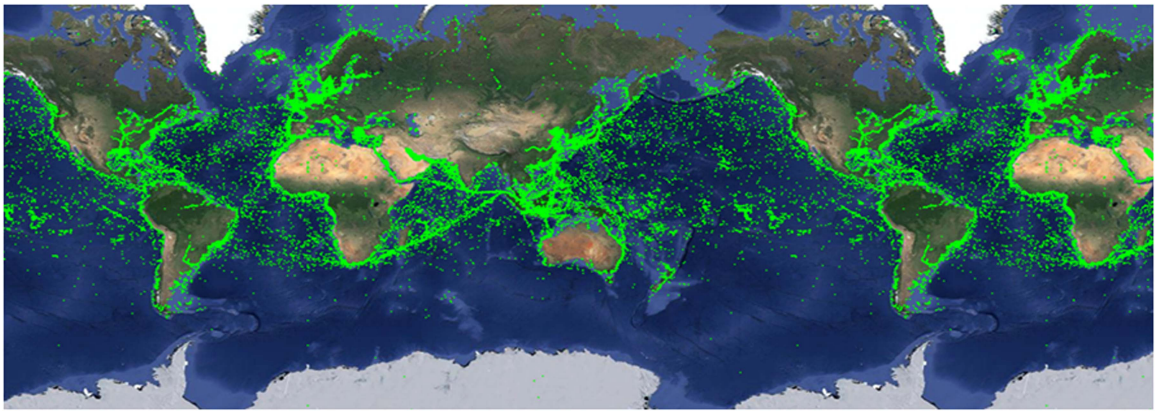

FIGURE 1. Visualization of global vessel positions collected from massive AIS messages (October 2018), available from https://www.shipxy.com/. The sequences of timestamped location points (i.e., positions) contribute to the vessel trajectories, which have the properties of large storage volume and high redundancy. Numerous data compression methods have thus been proposed to simplify the raw vessel trajectories to reduce the computational burdens for trajectory data mining tasks.

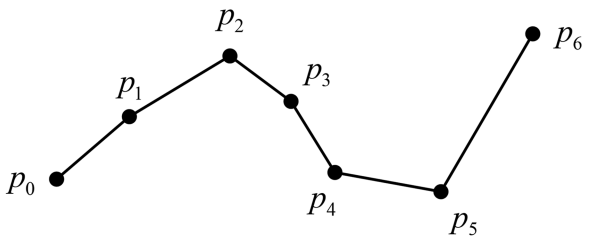

FIGURE 2. Example of one raw vessel trajectory, which contains 7 timestamped trajectory points.

a crucial role in vessel AIS data compression, providing valuable insights and contributing to the development of more effective post-processing techniques.

The Sliding Window (SW) algorithm [51] and Opening Window (OPW) algorithm [42] are similar to the DP algorithm. These methods propose to iteratively measure the amount of information loss to simplify the trajectories. However, they no longer iterate over the entire trajectory, which utilizes the concept of "window". The "window" contains some points in the track, first iterating using "window" then updating the "window" information continuously until the compression is completed. The Opening Window Time Ratio (OPW-TR) [42] addresses a critical limitation of the traditional OPW algorithm, which overlooks the time dimension. To overcome this constraint, OPW-TR incorporates the concept of Synchronous Euclidean Distance (SED). This distance metric allows for the calculation of the distance between an intermediate point and a trajectory segment. By considering both spatial and temporal aspects, OPW-TR ensures a comprehensive assessment of the trajectory, enabling more accurate and reliable simplification. The Dead Reckoning algorithm (DR) [52] introduces improvements that prioritize compression speed over compression effectiveness. This algorithm assumes that the object maintains a constant speed and direction during movement. It predicts the position of a new arrival point using the speed and direction of the last cached point. If the Euclidean distance between the predicted point and the actual value falls within an acceptable range, the new point is ignored. Otherwise, it is compressed and used as a new cache. Expanding on this concept, Potamias et al. [53] proposed the Threshold algorithm, which leveraged velocity and direction to establish a safety region for filtering trajectory points. This algorithm sets predefined threshold values for velocity and direction change. By combining the velocity and direction of the two nearest trajectory points according to these thresholds, a safety region can be constructed. Points falling within this safety region are considered redundant and can be deleted. Conversely, points falling outside the safety region indicate a change in the state of the moving object and should be retained.

Muckell et al. [54] introduced the Spatial Quality Simplification Heuristic (SQUISH) algorithm, which bears similarities to the OPW algorithm. SQUISH initializes a priority queue of size k and progressively adds trajectory points to the queue. Once the queue reaches its capacity, it removes points that contribute the least to the overall error and recalculates the priority of each remaining point. SQUISH excels in terms of low time complexity, high compression ratios, and low trajectory compression error. However, it is worth noting that the compressed trajectory generated by SQUISH does not have an explicit upper bound for the error. To overcome this limitation, Muckell et al. [55] developed the SQUISH-E algorithm as an advanced version of the traditional SQUISH algorithm. SQUISH-E optimizes the compression rate within a specified error threshold, achieving the best compression performance given the desired level of error control.

A. MOTIVATION AND CONTRIBUTIONS

This work aims to comprehensively review and consolidate the existing research on vessel trajectory compression to identify current gaps and propose future research directions. To the best of our knowledge, there is no previous study regarding the survey of AIS-based vessel trajectory compression. In this work, we will introduce 12 representative batch and online

trajectory compression algorithms in detail. A vessel trajectory compression software, called *AISCompress*, has also been developed to make it easier for interested readers to understand trajectory compression. Comprehensive experiments on realistic AIS data have been implemented to evaluate the trajectory compression results. The main contributions of this work can be summarized as follows.

- We complete a comprehensive literature review on AIS-based vessel trajectory compression algorithms and classify these algorithms into two categories, batch (offline) and online versions. For the sake of better understanding, we have introduced the computational details and discussed the main differences.
- A flexible and open software, regarding both batch and online vessel trajectory compression, is developed using the C++ programming language. This software could help users significantly reduce their workload when meeting the problem of large-scale vessel trajectory data compression.
- Extensive experiments on realistic AIS data sets have been implemented to comprehensively assess both batch and online vessel trajectory compression algorithms. The computing time, compression ratio, trajectory similarity, and trajectory length loss rate are jointly adopted to quantitatively evaluate the compression results. According to the comparative analysis, we can better understand the strengths and limitations of these algorithms. It is able to assist interested readers in selecting the most suitable algorithm to compress vessel trajectories, thus enhancing the efficiency and accuracy of large-scale trajectory data processing in practical applications.
- This review raises some long-standing challenges and problems, such as how to improve the compression efficiency and accuracy, and how to make the compression algorithms cope with the complex maritime environment. It could provide clear directions and deep insights for future research.

B. ORGANIZATION

The remainder of this paper is organized as follows. Section II introduces the basic concepts and preliminaries related to vessel trajectories. Section III detailedly introduces 5 trajectory compression algorithms in batch mode, and Section IV introduces 7 different online trajectory compression algorithms. Section V provides the vessel trajectory experiments, including the maritime datasets, performance metrics, comparative experiments, etc. Section VI introduces the vessel trajectory compression software we developed. We finally provide the summary and future directions of this work in Section VII.

II. BASIC CONCEPTS AND PRELIMINARIES

A. BASIC CONCEPTS OF VESSEL TRAJECTORIES

The vessel trajectory $\mathcal{X}$ is a sequence of AIS data with timestamps $\{p_1, p_2, \ldots, p_N\}$. The i-th trajectory point in $\mathcal{X}$ can be represented as $\mathcal{X}[i]$, or as a triplet (x_i, y_i, t_i), where (x_i, y_i) represents the spatial position of p_i, i.e., the latitude and longitude, and t_i is the sampling time of the trajectory data point p_i. In this work, $\mathcal{X}[t_s : t_e]$ is used to represent the sub-trajectory composed of trajectory points p_i in the time window $[t_s : t_e]$ of $\mathcal{X}$, where $s \le i \le e$. Conversely, the subsequence of $\mathcal{X}$ can be represented as $p_{s_1}, p_{s_2}, \ldots, p_{s_M}$, where the trajectory point $p_{s_i} \in \mathcal{T}$, and $1 \le s_1 < \ldots < s_M \le N$. In this paper, $\overrightarrow{p_s p_e}$, satisfying $1 \le s < e \le N$, is used to represent a directed trajectory segment, the starting trajectory point of the directed trajectory segment $\overrightarrow{p_s p_e}$ is p_s, and it ends at point p_e. The direction of the directed trajectory segment $\overrightarrow{p_s p_e}$ is defined as follows:

Definition: Direction of the trajectory segment $\overrightarrow{p_s p_e}$: The direction of the trajectory segment $\overrightarrow{p_s p_e}$ is represented by $\theta(\overrightarrow{p_s p_e})$, defined as the angle of counterclockwise rotation from the positive x-axis to the vector $\overrightarrow{p_s p_e}$.

The basic problem of trajectory simplification is to find a subsequence of M points as the best approximation trajectory of the original trajectory $\mathcal{T}$. This subsequence of M points is called the compressed trajectory $\mathcal{T}'$, where $M < N$, N represents the number of sampling points of the original trajectory. The compression rate $\gamma = M/N$. It is worth noting that the starting and ending points of the original trajectory $\mathcal{T}$ must be retained in the compressed trajectory $\mathcal{T}'$, because the starting and ending points are critical trajectory semantic information. These M points divide the original trajectory's index space $[1, N]$ into $M + 1$ intervals. In this paper, $\overrightarrow{p_{s_i} p_{s_{i+1}}}$ is called the simplified trajectory segment because the trajectory points in $\overrightarrow{p_{s_i} p_{s_{i+1}}}$ have indices in the interval $(p_{s_i}, p_{s_{i+1}})$. These trajectory points will be discarded, causing data loss. In addition, the compression error is commonly defined as the distance from the original trajectory point to the corresponding simplified trajectory segment.

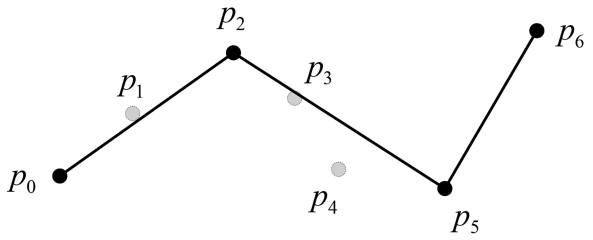

FIGURE 3. Example of one compressed vessel trajectory. The important feature points of the raw trajectory, shown in Fig. 3, are preserved during trajectory compression.

B. ERROR-BASED METRICS

The concept of Perpendicular Euclidean Distance (PED) is integral within our analysis. PED represents the shortest distance from point p_i to the line segment formed by points p_b and p_e. This distance can be mathematically expressed as $\mathrm{PED}(p_i, p_b, p_e)$, and its formal representation is provided as follows.

$$\mathrm{PED}(p_i, p_b, p_e)$$

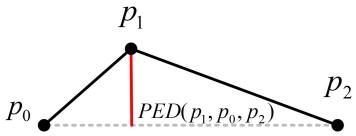

FIGURE 4. Illustration of perpendicular Euclidean distance.

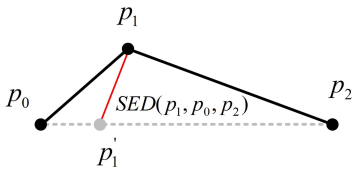

FIGURE 5. Illustration of synchronized Euclidean distance.

$$= \left| \frac{(y_e - y_b)x_i - (x_e - x_b)x_i + x_e y_b - y_e x_b}{\sqrt{(y_e - y_b)^2 + (x_e - x_b)^2}} \right|, \quad (1)$$

In Fig 4, the PED(p_1, p_0, p_2) is the shortest Euclidean distance from point p_1 to line p_0p_2.

The popular Synchronized Euclidean Distance (SED), proposed by Meratnia et al. [42], is used to address the limitations of Euclidean distance in capturing the dynamic variation of time series data. The SED represents the distance from the point p_i to the synchronization point p_i' of the line segment p_bp_e, denoted as SED(p_i, p_b, p_e). Its mathematical formula is given as follows.

$$\text{SED}\,(p_i, p_b, p_e) = \sqrt{\left(y_i - y_b - \frac{t_i - t_e}{t_e - t_b} y_e - y_b\right)^2 + \left(x_i - x_b - \frac{t_i - t_e}{t_e - t_b} x_e - x_b\right)^2} \quad (2)$$

In Fig. 5, SED(p_1, p_0, p_2) is the synchronous Euclidean distance from the point p_1 to the line segment p_0p_2, i.e., from the point p_1 to the point p_1'.

III. VESSEL TRAJECTORY COMPRESSION ALGORITHMS IN BATCH MODE

The vessel trajectory compression algorithm in batch mode is a kind of trajectory compression for raw AIS data. The algorithm can convert the raw AIS trajectory data with high precision and high resolution into low-dimensional and low-precision trajectory data by processing and analyzing AIS data. It can implement data compression under the premise of ensuring data quality.

A. UNIFORM SAMPLING ALGORITHM

The Uniform Sampling algorithm, a widely employed technique in the field of computer science, finds its applications in the filtering and retention of trajectory points through equidistant sampling. The foremost advantage of the Uniform Sampling algorithm is that it is simple, user-friendly, and ensures uniform distribution of the sampling point set.

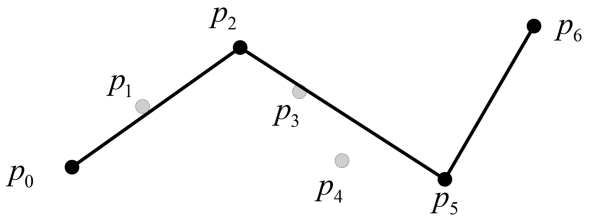

FIGURE 6. Compressed trajectory using the popular uniform sampling algorithm.

Algorithm 1: Uniform Sampling (X, ε).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$, and the error tolerance ε.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

```
Y = {};
for i = 0 to length(X) do
    if i/ε is N* natural number then
        Insert p_i into Y ;
    end
    Return Y ;
end
```

However, a notable drawback of this approach is that it only retains trajectory points at fixed interval values, consequently limiting its ability to fully preserve the intricate characteristics embedded within the trajectory data. It may not capture the meaningful dynamics and intricacies exhibited by the original trajectory to its fullest extent.

The Uniform Sampling algorithm is one of the most direct, simplest, and non-computational logic online trajectory simplification algorithms. Given the original trajectory $\mathcal{T}$ and the sampling interval λ, the Uniform Sampling algorithm downsamples the trajectory data at a fixed time interval λ to achieve the purpose of trajectory compression and remove the redundancy of adjacent points in the trajectory. The time complexity of the Uniform Sampling algorithm is $O(1)$, and the space complexity is $O(1)$. Due to the fixed sampling interval used, it is impossible to control the error, which may lead to an increase in error and become uncontrollable. As shown in Fig. 6, the compressed trajectory $\mathcal{T}' = \{p_0, p_2, p_5, p_6\}$ is obtained according to the fixed sampling interval.

B. DOUGLAS-PEUCKER ALGORITHM

The Douglas-Peucker (DP) algorithm, introduced by Douglas and Peucker in 1973, was originally proposed for line simplification [56]. It has also been widely used for trajectory compression [57], [58]. The DP algorithm takes spatial distance into account and is widely applied in cartography and computer graphics. Many cartographers consider it to be one of the most accurate algorithms for line generalization. It operates on the principle of partitioning, effectively reducing the original trajectory data into a sequence of key points [59]. This compression significantly decreases the data size and

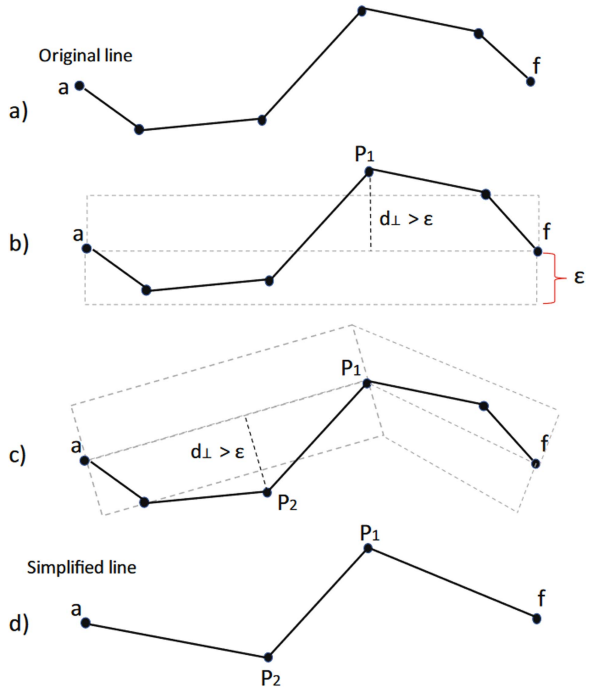

FIGURE 7. Illustration of Douglas-Peucker (DP) algorithm.

storage requirements. Compared to other trajectory compression algorithms, the DP algorithm stands out due to its ease of implementation, low computational complexity, and superior preservation of trajectory characteristics.

The process of this algorithm is depicted in Fig. 7. The first step involves selecting the first point as the anchor point and the last one as the float point (Fig. 7(a)). The starting curve is a set of points and a threshold (i.e., distance dimension) $\epsilon > 0$. For all the intermediate data points, the perpendicular distance ($\perp$) of the farthest point (P_1) and the line segment between the starting and the ending points ($LS_{a\to f}$) is determined. If the point is closer than ϵ to the $LS_{a\to f}$, all the points between a and f are discarded. Otherwise, the line is cut at this data point that causes the maximum distance, and this point (P_1) is included in the resulting set. P_1 becomes the new float point for the first segment, and the anchor point for the second segment (Fig. 7(b)). The same procedure is recursively repeated for both segments until all points have been processed (Fig. 7(c)). Once the recursion is completed, a new simplified line is generated, consisting only of those points that have been marked to be kept (Fig. 7(d)).

The fundamental concept behind the DP algorithm is to eliminate unnecessary points while maintaining the accuracy of the original trajectory data. To achieve this, the algorithm divides the trajectory data into several contiguous point segments and selects the farthest point within each segment as a key point. By recursively processing the sub-segments, the algorithm compresses the original trajectory data into a series of key point sequences. The step-by-step procedure is detailedly outlined in Algorithm 2.

Algorithm 2: DouglasPeucker (X, ε).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$, and the error tolerance ε.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

$endIndex = length(\mathrm{X}) - 1$;
$keyIndex = -1$;
$maxDistance = 0$;
$Y = \{\}$;
// Find the point of maximum vertical distance.
for $i = 2$ **to** *endIndex-1* **do**
  $distance = PED(X[i], X[1], X[endIndex])$;
  if $distance > maxDistance$ **then**
    $keyIndex = i$;
    $maxDistance = distance$;
  end
end
// If the maximum point of the vertical distance is less than the threshold, it is not necessary to keep the point.
if $maxDistance < \varepsilon$ **then**
  $Y \leftarrow [X[1], X[endIndex]]$;
else
  // Recursively process the sequence of points on the left and right sides.
  $leftX = \mathrm{DouglasPeucker}(X[1 : keyIndex], \varepsilon)$;
  $rightX =$ $\mathrm{DouglasPeucker}(X[keyIndex : endIndex], \varepsilon)$;
  // Combine the key points retained on the left and right sides.
  $Y \leftarrow leftX[:-1] + rightX$;
end
Return Y;

The main drawback of the DP algorithm is that it does not consider the temporal information of one trajectory. The spatial error still exists since the DP algorithm finds the closest point on the compressed trajectory to each point on the original trajectory. To achieve satisfactory compression results, both temporal and spatial dimensions of the vessel trajectories should be considered.

C. TOP-DOWN TIME-RATIO ALGORITHM

The Top-Down Time-Ratio (TD-TR) algorithm is a significant advancement in the field of trajectory compression [42]. It is an enhanced version of the well-known DP algorithm, incorporating temporal considerations into the compression process. The traditional DP algorithm has a limitation in that it employs the Euclidean distance as a metric for calculating distances between points and lines, thereby overlooking the temporal aspect of the trajectory data. This oversight can be problematic as some points in the trajectory data may have very short time intervals, while others might have relatively

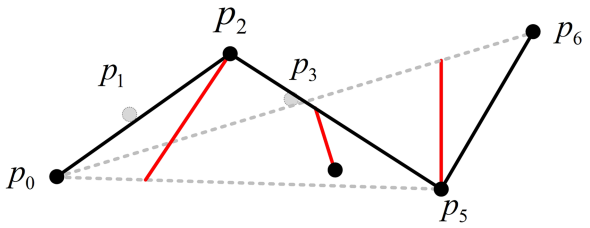

FIGURE 8. Illustration of TD-TR algorithm.

longer time intervals. Relying solely on the Euclidean distance to measure the point-line distances could yield unreasonable results in such cases. The TD-TR algorithm addresses this limitation by using the SED as its error metric, which considers both spatial and temporal aspects of the trajectory, rather than relying solely on spatial error. This approach allows for a more accurate representation of the trajectory data. Specifically, some points in the trajectory data may have very short time intervals, while others might have relatively longer time intervals. Relying solely on the Euclidean distance to measure the point-line distances could yield unreasonable results in such cases. However, the worst-case running time of TD-TR is $O(n^2)$ since it extends the original DP algorithm. The $O(n \log n)$ implementation of DP takes advantage of geometric properties that do not hold for SED. Therefore, it cannot be applied to TD-TR. This is an area that may warrant further investigation for optimization.

In the TD-TR algorithm, aside from utilizing the SED to compute distances between points and straight lines, the overall procedure closely resembles that of the DP algorithm. It involves selecting the initial and final points, identifying the point with the greatest temporal distance, recursively segmenting the point sequences, and retaining the key points. The detailed steps of the TD-TR algorithm can be found in Algorithm 3.

Fig. 8 presents the screening process of the TD-TR algorithm. In this figure, the black points represent the retained vessel trajectory points, while the grey points depict the discarded vessel trajectory points. The solid line segment represents the metric distance, whereas the red line segment signifies the SED exceeding the threshold. In Fig. 8, the points p_2, p_4, p_5 points have SED greater than the target threshold. These points are retained accordingly.

D. ADAPTIVE DOUGLAS-PEUCKER ALGORITHM

To effectively compress and simplify trajectories with varying characteristics, a single fixed threshold may not be suitable for all cases. Different trajectories, such as linear, curved, or complex paths, have varying requirements for compression and detail retention. Therefore, to more accurately address these differences, it is necessary to adaptively enhance the traditional DP algorithm. This enhancement allows the algorithm to dynamically adjust the threshold based on the specific characteristics of each trajectory. Adaptive Douglas-Peucker (ADP) algorithm is a sophisticated enhancement of the original DP algorithm, a renowned method for curve simplification [47].

Algorithm 3: TD-TR (X, ε).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$, and the error tolerance ε.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

$endIndex = length(\mathrm{X}) - 1$;
$keyIndex = -1$;
$maxDistance = 0$;
$Y = \{\}$;
// Find the point of maximum SED.
for $i = 2$ **to** $endIndex - 1$ **do**
  $distance = SED(X[i], X[1], X[endIndex])$;
  if $distance > maxDistance$ **then**
    $keyIndex = i$;
    $maxDistance = distance$;
  end
end
// If the maximum point of SED is less than the threshold, it is not necessary to keep the point.
if $maxDistance < \varepsilon$ **then**
  $Y \leftarrow [X[1], X[endIndex]]$;
else
  // Recursively process the sequence of points on the left and right sides.
  $leftX = \text{TD-TR}(X[1 : keyIndex], \varepsilon)$;
  $rightX = \text{TD-TR}(X[keyIndex : endIndex], \varepsilon)$;
  // Combine the key points retained on the left and right sides.
  $Y \leftarrow leftX[: -1] + rightX$;
end
Return Y;

The ADP algorithm is an advanced modification of the original DP algorithm, introducing an innovative automatic threshold selection method that adapts to the specific characteristics of each trajectory. This optimal threshold is computed based on channel and trajectory characteristics, segmentation framework, and mean distance, enabling more precise control over which points are retained. Unlike the DP algorithm, which requires manual specification of a fixed threshold, the ADP algorithm dynamically adjusts the threshold based on geometric characteristics and the vertical Euclidean distance of the vessel's trajectory. This adaptability makes the ADP algorithm particularly beneficial in applications like AIS-based vessel trajectory compression, where it helps eliminate redundant information, preserve key features, and simplify data for subsequent mining. Empirical studies [30] have demonstrated that the ADP algorithm not only effectively simplifies vessel trajectory data but also extracts valuable information. It has been successfully applied in trajectory

Algorithm 4: ADP(X).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

$Y = \{\}$;
// Calculate the PED from each point to the line connecting the end point and the start point.
$endIndex = length(X)$–1;
for $i = 1$ **to** $endIndex$ **do**
  $d_i = PED(X[i], X[1], X[endIndex])$;
end
// Calculate the threshold value θ based on geometric features
if *vessel Trajectory X is straight trajectory* **then**
  $\theta = |\frac{y_{pN}-y_{p0}}{x_{pN}-x_{p0}}| \times \sum_{i=1}^{N-1} d_i/(N-2)$;
else if *vessel trajectory X is curved trajectory* **then**
  $\theta = \sum_{i=1}^{N-1} d_i/(N-2)$;
else vessel trajectory $X$is complex trajectory;
Divide the complex trajectory into straight and curved trajectory;
Return to Step 6;
// Generate compression trajectory
for $i = 1$ **to** $length(X)$ – *1* **do**
  if $d_i > \theta$ **then**
    $Y \leftarrow p_i$;
  end
  Return Y
end

Algorithm 5: ADPS(X).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

$Y = \{\}$;
// Annular and semi-annular trajectory segmentation frames.
if *there is annular and semi-annular trajectory* **then**
  split them
end
// Calculate the threshold value of ADPS
$k = \frac{y_{pN}-y_{p0}}{x_{pN}-x_{p0}}$;
$endIndex = length(X) - 1$;
for $i = 1$ **to** $length(X)$ - *1* **do**
  $a_i = |v_{i+1} - v_i|/|t_{i+1} - t_i|$;
  $\overline{a} = \sum_{j=0}^{N} a_i/(N-1)$;
  $d_i = PED(X[i], X[1], X[endIndex])$;
  if $a_i > \overline{a}$ and $|k| > 1$ **then**
    $\theta_v = \frac{1}{|k|} \cdot a_i$;
    $\theta_d = \frac{1}{|k|} \cdot \sum_{i=1}^{N-1} d_i/(N-2)$;
    $\theta = \theta_v + \theta_d$;
  else
    $\theta_v = |k| \cdot a_i$;
    $\theta_d = |k| \cdot \sum_{i=1}^{N-1} d_i/(N-2)$;
    $\theta = \theta_v + \theta_d$;
  end
end
// Generate compression trajectory
for $i = 1$ **to** $length(X)$ – *1* **do**
  if $d_i > \theta$ **then**
    $Y \leftarrow p_i$;
  end
  Return Y
end

classification and clustering, reducing computational costs while maintaining the accuracy of results. This underscores the potential of the ADP algorithm as a powerful tool in data analysis and information extraction. The detailed steps of the ADP algorithm are outlined in Algorithm 4.

The determination of the distance threshold θ in the ADP algorithm is a pivotal step, as illustrated in Fig 4. This figure demonstrates how the threshold $\varepsilon = d_1 + d_2 + d_3 + d_4 + d_5 + d_6$ is derived.

E. ADAPTIVE DOUGLAS-PEUCKER WITH SPEED ALGORITHM

While the ADP algorithm significantly enhances trajectory simplification by adapting thresholds based on geometric characteristics, it does not consider dynamic aspects such as speed and rate of speed variation. These factors are crucial for analyzing moving objects accurately. The Adaptive Douglas-Peucker with Speed (ADPS) algorithm [60] addresses this limitation by incorporating additional parameters such as velocity and rate of speed variation into the threshold setting process for each trajectory. This allows for a more nuanced understanding of movement patterns by considering the distance of feature points and their dynamic changes. The detailed steps of the ADPS algorithm are outlined in Algorithm 5.

IV. VESSEL TRAJECTORY COMPRESSION ALGORITHMS IN ONLINE MODE

The batch mode vessel compression method effectively compresses entire trajectories by utilizing captured trajectory data, providing a global approximation. However, this method may not suit scenarios requiring immediate data processing, such as navigation systems and emergency responses. In these situations, online compression algorithms become essential [61], [62]. These algorithms, including the OPW [63], and Dead Reckoning Algorithm (DRA) [64], are expertly designed to process real-time data streams efficiently by discarding only the minimally essential points. This allows for the accurate

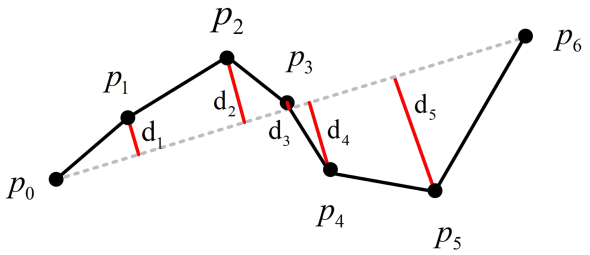

FIGURE 9. Illustration of threshold determination for ADP algorithm.

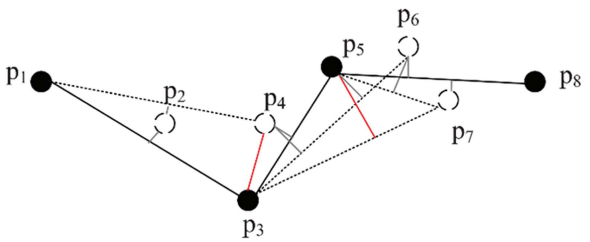

FIGURE 10. Visual illustration of opening window (OPW) compression algorithm.

representation of trajectories while ensuring system performance is not compromised, making these methods ideal for applications like real-time tracking and mobile location-based services where timely data processing and decision-making are crucial.

A. OPENING WINDOW ALGORITHM

The Opening Window Algorithm (OPW) algorithm [66] is an online trajectory simplification algorithm based on windows [65]. This algorithm calculates the vertical Euclidean distance of all points within the window, a subsequence of consecutive trajectory points. The selection of approximate endpoints for line segments is based on the relative error of the summed errors within the window compared to a specified threshold value. For instance, if an error exceeding the threshold is found within the window, the penultimate point in the trajectory within that window is chosen as the endpoint of the approximate line segment. The detailed steps of the OPW algorithm can be referenced in Algorithm 6. When the number of trajectory points in window $\mathcal{W}$ exceeds 3, i.e., $|\mathcal{W}| > 3$, all trajectory points in the window are calculated for the PED to the window's positioning point and floating point p_s, p_e. The trajectory point p_s with the maximum distance to the vector $\overrightarrow{p_s p_m}$ is found. If $PED(p_s) > \epsilon$, the starting point p_s of window $\mathcal{W}$ is added to the compressed trajectory $\mathcal{X}'$, and p_e becomes the new positioning point. Then, trajectory stream data is added to window $\mathcal{W}$. Otherwise, a new trajectory point p_{e+1} is added to the window, and p_{e+1} becomes the floating point of the window. The same process is applied to the updated window $\mathcal{W}$. The OPW algorithm requires multiple calculations of the PED of the trajectory points in the window to the window's positioning point and floating point, resulting in a relatively high time complexity. The time complexity of the OPW algorithm is $O(N^2)$, and the space complexity is $O(N)$.

Algorithm 6: OPW(X, ε).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$, and the error tolerance ε.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

```
Y = {};
// Set window start and end points.
WBI = 0;
// WBI:WinBeginIndex
WEI = 2;
// WEI:WinEndIndex
// Traversal
while WEI < length(X) do
    // Calculate PED of the points in the window.
    for i = 1to(WEI − WBI) do
        d_i = PED(X[i], X[WBI], X[WEI]);
        // Retain track points and update window when greater than threshold
        if d_i > ε then d_i > θ;
        Y ← p_i ;
        WBI = i;
        WEI = WBI + 2;
        else WEI++ ;
    end
end
Return Y
```

The process of OPW is given in Figure.10. Initially, the original trajectory is defined as $\mathcal{X}_1 = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$. Given the window $\mathcal{W}_1$ and the error threshold ϵ_t, the first step involves checking the number of trajectory points in $\mathcal{W}_1$. Since the number is initially less than 3, p_1 and p_2 are directly added to $\mathcal{W}_1$, resulting in $\mathcal{W}_1 = p_1, p_2$. When p_3 is added, the PED of p_2 to the simplified segment $\overrightarrow{p_1 p_3}$ is calculated. If $PED(p_2) < \epsilon_t$, p_3 is included in $\mathcal{W}_1$, updating it to p_1, p_2, p_3. As p_4 is introduced, $\mathcal{W}_1$ becomes p_1, p_2, p_3, p_4, and the PEDs of p_2 and p_3 relative to $\overrightarrow{p_1 p_4}$ are computed. As illustrated in the figure, it is determined that $PED(p_3) > \epsilon_t$, leading to the inclusion of p_1 in the compressed trajectory $\mathcal{X}'_1$, which becomes $\mathcal{X}'_1 = p_1$. The window is then reset to p_3. By repeating this process, the OPW algorithm compresses the trajectory to $\mathcal{X}'_1 = p_1, p_3, p_5, p_8$.

B. OPENING WINDOW TIME-RATIO ALGORITHM

The OPW algorithm effectively compresses spatial trajectory data but overlooks the temporal dimension. To address this shortcoming, the Opening Window Time Ratio (OPW-TR)

Algorithm 7: OPW-TR(X, ε).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$, and the error tolerance ε.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

$Y = \{\}$;
// Set window start and end points.
$WBI = 0$;
// WBI:WinBeginIndex
$WEI = 2$;
// WEI:WinEndIndex
// Traversal
while $WEI < length(X)$ **do**
  // Calculate SED of the points in the window.
  for $i = 1$**to**$WEI - WBI$ **do**
    $d_i = SED(X[i], X[WBI], X[WEI])$;
    // Retain track points and update window when greater than threshold
    if $d_i > \varepsilon$ **then** $d_i > \theta$;
    $Y \leftarrow p_i$;
    $WBI = i$;
    $WEI = WBI + 2$;
    else $WEI++$;
  end
end
Return Y

Algorithm 8: DR(X, ε).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$, the error tolerance ε, and $\{lat, lon, cog, sog\} \in p_i$.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

$Y = \{\}$;
$Y \leftarrow p_0$;
$SumD = 0$;
// Set window start and end points.
$DRBeginIndex = 0$;
// Traversal
for $i = 2$**to** $length(X)$ **do**
  // Accumulated parameter error
  $SumD+ = time(p_i, p_{i-1}) \times SOG(p_{DRBeginIndex}) \times sin(angles(p_i, p_{DRBeginIndex}))$;
  // Retain track points and update window when greater than threshold
  if $SumD > \varepsilon$ **then**
    $Y \leftarrow p_i$;
    $SumD = 0$;
    $DRBeginIndex = i$;
  end
end
$Y \leftarrow p_N$;
Return Y

algorithm was introduced, integrating the temporal aspect into trajectory compression [42].

In the classical OPW algorithm, the Euclidean distance is used as a metric to calculate the distance between points and lines. However, this approach overlooks the importance of temporal information. To address this limitation, the OPW-TR algorithm integrates the temporal dimension into the original OPW algorithm, resulting in a more accurate approximation of the trajectory. This integration not only enhances the accuracy of error measurement but also optimizes the time and space complexity of the compression process for trajectory data. The detailed steps of the OPW-TR algorithm can be found in Algorithm 7.

In OPW-TR algorithm, SED is utilized as the metric error. Given a trajectory T and a SED threshold ε, OPW-TR algorithm starts by defining a segment between the first point P_1 (the anchor point) and the third point P_3 (the float point). Then the algorithm calculates all the SED values of the points between the anchor point and the float point. As long as all the SED values are smaller than ε, an attempt is made to move the float point one point forward in trajectory T. When the SED threshold ε is exceeded, two strategies can be applied: either the point causing the threshold violation becomes the new anchor point, or the point just before it becomes the new anchor point. In Fig. 7, the green rectangle represents the second window, which serves as a filter to retain points that meet the criteria. Specifically, if the SED exceeds the predetermined threshold value, the point p_4 will be filtered out and excluded from the final compressed trajectory.

C. DEAD-RECKONING ALGORITHM

The Dead-Reckoning (DR) predicts the next position using estimated speed, heading, and known positional data [66], [67]. The algorithm then refines these predictions by comparing the deviation between the actual and estimated trajectories, improving accuracy and precision. This method is widely used in mobile trajectory data transmission and analysis, and it reduces data storage and transmission costs while enhancing processing efficiency. The detailed steps of the DR trajectory compression method are provided in Algorithm 8.

For the DR algorithm, the process begins by determining the vessel's projected position based on its current heading and speed. Next, the distance between this projected position and the next actual vessel point is calculated. If this distance exceeds a predefined threshold ε, the previous point is retained as the starting point for the next projection. As illustrated in Fig. 12, the blue triangles represent the retained trajectory points, while the gray triangles represent those that are discarded. The solid line segment shows the metric distance, with the red line segment highlighting the distance of the point that

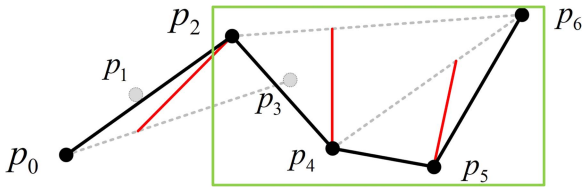

FIGURE 11. Visual illustration of OPW-TR algorithm.

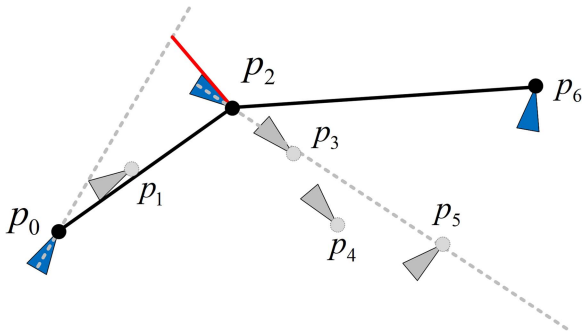

FIGURE 12. Visual illustration of dead-reckoning algorithm.

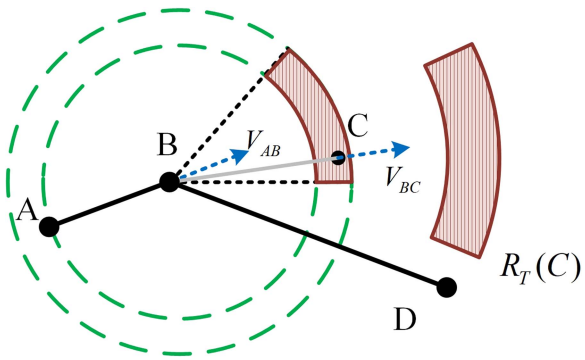

FIGURE 13. Definition of the R_T region.

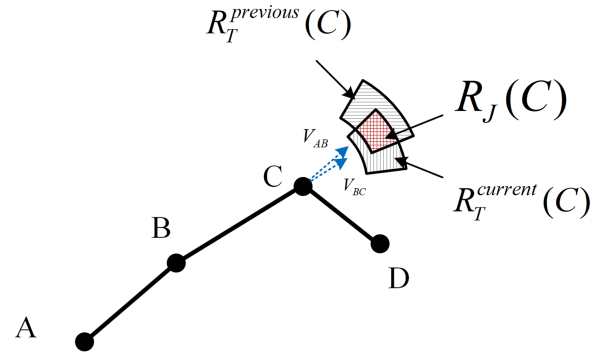

FIGURE 14. The visual illustration of threshold-guided compression algorithm.

Algorithm 9: Threshold(X, ε).

Input: Raw vessel Trajectory $X = \{p_0, p_1, \ldots, p_N\}$, the speed Threshold ε, and the direction Threshold α.

Output: Compression vessel Trajectory $Y = \{s_0, s_1, \ldots, s_M\}$

```
Y = {};
Y ← p_0;
// Traversal
for i = 2to length(X) - 1 do
    // Calculating the joint area range
    R_{p_{i-1}} = R_T^{previous}(p_{i-1}) ∩ R_T^{current}(p_{i-1});
    if p_1 isin R_{p_{i-1}} then
        break;
    end
    else Y ← p_i;
    ;
end
Return Y;
```

is retained. In this case, since the distance for p_2 surpasses the threshold, p_2 is retained.

D. THRESHOLD-GUIDED SAMPLING ALGORITHM

The threshold-guided sampling algorithm compresses vessel trajectories by analyzing the direction and velocity of timestamped points [68]. The core idea is to identify trajectory points that show significant changes beyond a specified range. Each point in the trajectory has direction and velocity parameters, which help predict a reasonable range for the next point. By combining the predicted ranges of the current and previous points, a joint range is established. Based on this joint range, the algorithm determines whether to retain or discard each trajectory point. The detailed steps of the threshold-guided sampling algorithm are outlined in Algorithm 9.

The single-step region range R_T is illustrated in Fig. 13. In this figure, it can be observed that Point C falls within the range $R_T(B)$, which is removed. On the other hand, Point D lies outside the range $R_T(C)$. It is therefore retained.

The Threshold algorithm calculates the joint region range using $R_T^{previous}$ and $R_T^{Current}$, The process is depicted in Fig. 14, and it can be found that Point D does not fall within the joint region range $R_J(C)$. As a result, Point D is retained.

E. SPATIAL QUALITY SIMPLIFICATION HEURISTIC ALGORITHM

The Spatial Quality Simplification Heuristic (SQUISH) is a popular trajectory compression algorithm, which is designed to reduce storage space and enhance processing efficiency [54]. The trajectory compression results can be obtained by sampling the temporal and spatial dimensions at varying degrees.

The core of the SQUISH algorithm lies in employing a heuristic approach to simplify spatial information quality. By analyzing the spatial distance between consecutive trajectory points, high-quality trajectory points are selectively retained

Algorithm 10: SQUISH(X, ε).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$, and the error tolerance ε.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

```
// Set buffer
Y is a fixed size(ε · length(X)) buffer ;
Y ← p_0;
Set SED of p_0 is 0 ;
// Input trajectory
for i = 1to length(X) - 1 do
    Y ← p_i;
    Update the SED of the current insertion point in Y;
    if Y is full then
        find the point p_j in Y with the smallest SED;
        delete p_j from Y;
        Update the SED of the neighbor points of p_j;
    end
end
Return Y;
```

while poor-quality trajectory points are discarded. This algorithm has demonstrated superior performance in compressing trajectory data from large-scale AIS. The detailed steps of the SQUISH algorithm are outlined in Algorithm 10.

The SQUISH algorithm follows a sequential process where vessel trajectory points are fed into a fixed-size buffer. As each point is inserted, the SED value is calculated by the algorithm. When the buffer reaches its maximum capacity and new trajectory points continue to flow in, the point with the smallest SED is removed from the buffer. This removal affects the SED values of the two adjacent points, as the SED value of the deleted point is added to their existing SED values. The flowchart is visually displayed in Fig. 15, which details the step-by-step progress of SQUISH-based trajectory compression.

F. SPATIAL QUALITY SIMPLIFICATION HEURISTIC-EXTENDED ALGORITHM

The SQUISH-E is an enhanced version of the SQUISH algorithm, and it is also specifically designed for compressing and simplifying vessel trajectory data [55]. By combining the greedy algorithm and additional input parameters, SQUISH-E selectively preserves trajectory points with high spatial information quality, resulting in more precise trajectory data compression. The detailed steps of the SQUISH-E algorithm can be found in Algorithm 11, which outlines the specific operations.

The fundamental concept behind SQUISH-E revolves around the utilization of a priority queue Y. Each point's priority within the queue is determined by an upper threshold derived from the SED metric that would result from its deletion. By employing this approach to point removal, SQUISH-E effectively curtails the proliferation of SED errors. This methodology ensures that only points with minimal impact on the overall trajectory quality are discarded, thereby enhancing the accuracy of the compression process.

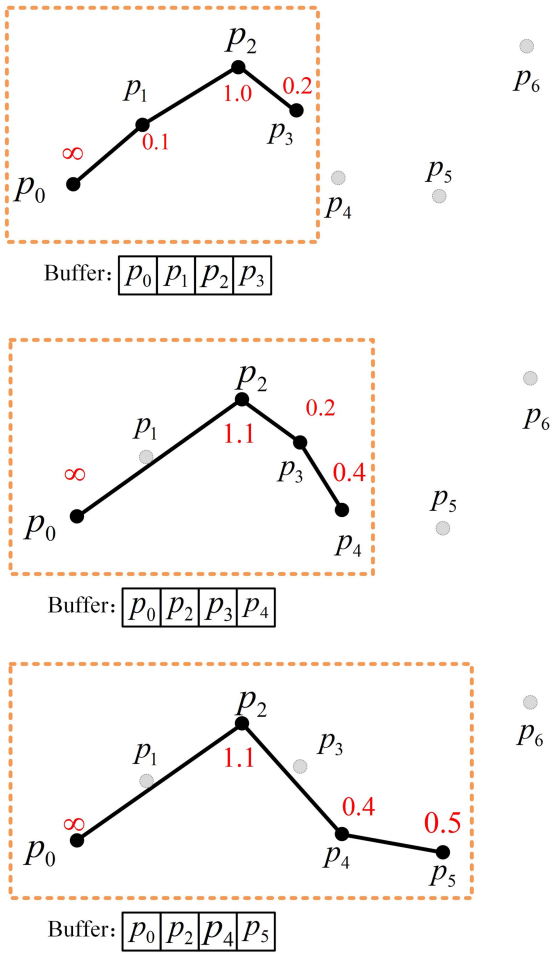

FIGURE 15. Visual illustration of SQUISH algorithm.

G. STTRACE ALGORITHM

STTrace is an efficient trajectory compression algorithm similar to SQUISH that uses a buffer and utilizes SED as a criterion for evaluating errors [53]. The algorithm operates by continuously inputting data into the buffer as a data stream, calculating the SED value for each vessel trajectory point upon input.

Once the buffer reaches its maximum capacity and a new vessel trajectory point (with a higher SED than the minimum SED value in the buffer) arrives, the point is inserted into the buffer. Simultaneously, the trajectory point with the minimum SED value is deleted, triggering the recalculation of SED values for the neighboring trajectory points.

A detailed step-by-step breakdown of the STTrace algorithm can be found in Algorithm 12, which outlines the

Algorithm 11: SQUISH-E(X, ε, α).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$, the compression ratio ε, and the SED Threshold α.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

```
// Set buffer
Y is a fixed size(ε · length(X)) buffer ;
Y ← p_0;
Set SED of p_0 is ∞ ;
// Input trajectory
for i = 1to length(X) - 1 do
    Y ← p_i;
    Update the SED of the current insertion point in Y;
    // Update buffer
    if Y is full then
        Find the point p_j in Y with the smallest SED;
        Delete p_j from Y;
        Update the SED of the neighbor points of p_j;
    end
end
p_min = the lowest value from Y;
while p_min < α do
    Delete p_min from Y;
    Update the SED of the neighbor points of p_min;
    Return to Step14;
end
Return Y;
```

Algorithm 12: STTrace(X, ε, α).

Input: Raw vessel Trajectory $X = \{p_0, p_1, ..., p_N\}$, and the error tolerance ε.
Output: Compression vessel Trajectory $Y = \{s_0, s_1, ..., s_M\}$

```
// Set buffer
Y is a fixed size(ε · length(X)) buffer ;
Y ← p_0;
Set SED of p_0 is ∞ ;
// Input trajectory
for i = 1to length(X) - 1 do
    Y ← p_i;
    Update the SED of the current insertion point in Y;
    if SED < min[Y] then
        break;
    end
    // Update buffer
    if Y is full then
        Delete p_j from Y;
        Update the SED of the neighbor points of p_j;
    end
end
Return Y;
```

specific operations of compressing and simplifying vessel trajectory data.

The STTrace algorithm incorporates a distinct calculation process for recalculating the SED values of adjacent points after removing the point with the smallest SED. Notably, unlike the SQUISH algorithm, the calculation for SED involves forming a line segment by connecting two neighboring points within the buffer, in which the vertical foot plays a crucial role.

As illustrated in Fig. 16, the algorithm flow presents a step-by-step depiction of how STTrace operates. This flow showcases the specific procedures and computations employed by STTrace to recalculate SED values while considering the formation of line segments and their associated vertical feet.

V. EVALUATION

To evaluate the compression efficiency and performance of batch and online compression algorithms, an evaluation system for the trajectory compression algorithms was utilized. The experimental program was implemented in C++ using Microsoft Visual Studio as the integrated development environment (IDE). The hardware environment for the experiments consisted of an Intel(R) Core(TM) i5-7300HQ CPU @ 2.50 GHz and 16GB of RAM.

A. COMPARISON ALGORITHM

This study focuses on two types of compression algorithms: batch and online. The batch algorithms investigated include Uniform, DP, TD-TR, ADP, and ADPS algorithms. Meanwhile, the online algorithms examined consist of OPW, OPW-TR, DR, Threshold, SQUISH, SQUISH-E, and STTrace algorithms. Each algorithm has its own unique spatio-temporal complexities and error metric method. Of particular note is the SQUISH-E algorithm, which allows for the adjustment of two parameters, i.e., compression ratio and synchronous Euclidean distance. In this experiment, the SQUISH-E algorithm was configured in two variations: 1) Setting the compression ratio to 1, denoted as SQUISH-E (sed) algorithm. 2) Setting the synchronous Euclidean distance threshold to 0, referred to as the SQUISH-E (ratio) algorithm. When the algorithm's compression ratio is set to 1, it operates as a batch compression algorithm. For a comprehensive overview of the algorithms and their specifics, please refer to Table 1, where N represents the number of points in the vessel trajectory, and m indicates the queue size utilized in the respective algorithm.

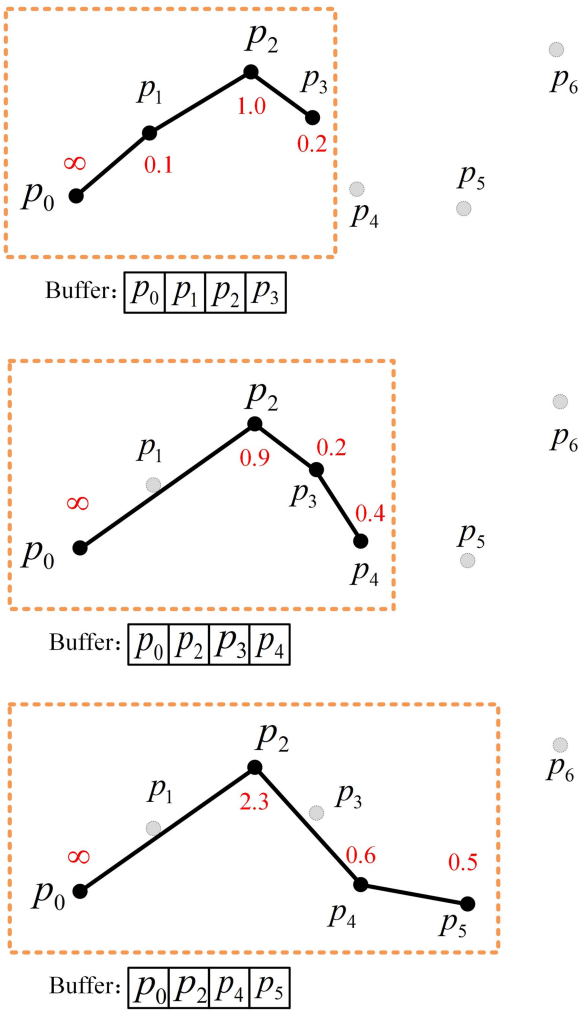

FIGURE 16. Visual illustration of STTrace algorithm.

B. MARITIME DATASETS

The currently available free open-source AIS databases encompass a variety of options, including AISHub, HELCOM, MarineCadastre, CruiseMapper, Marine Traffic, and VesselFinder, among others. Among these, the AISHub, HELCOM, MarineCadastre, and CruiseMapper databases are accessible to all users without any subscription requirement. However, accessing data from the two major AIS databases, Marine Traffic and VesselFinder, necessitates a subscription service.

The AISHub database primarily provides AIS data for vessels located in global coastal areas, while the MarineCadastre database focuses on AIS data for coastal waters within the United States. In contrast, the HELCOM database specifically includes AIS information for vessels operating in the Baltic Sea region, alongside data on pollution detection.

The three databases (CruiseMapper, Marine Traffic, and VesselFinder) serve as comprehensive sources of global vessel AIS information. Table 2 provides a detailed overview of the aforementioned maritime datasets.

TABLE 1 Comparisons of Different Trajectory Compression Algorithms Considered in Our Numerical Experiments

Algorithm	Category	Time complexity	Error metric
Uniform	Batch	$O(N)$	-
DP	Batch	$O(N^2)$	SED
TD-TR	Batch	$O(N^2)$	SED
ADP	Batch	$O(N \times logN)$	PED
APDS	Batch	$O(N \times logN)$	PED
OPW	Online	$O(N^2)$	PED
OPW-TR	Online	$O(N^2)$	SED
DR	Online	$O(N)$	Angle
Threshold	Online	$O(N^2)$	Speed,Angle
SQUISH	Online	$O(N \times logm)$	PED
SQUISH-E(sed)	Online	$O(N \times logN)$	SED
SQUISH-E(ratio)	Online	$O(N \times log(N/ratio))$	SED
STTrace	Online	$O(N^2)$	SED

C. PERFORMANCE METRICS

- Compression Ratio [77]: In the context of trajectory compression, the compression ratio is typically defined as the ratio between the number of points retained after compression and the total number of points before compression. When only considering the compression ratio, a smaller value indicates a more effective compression. However, it is important to note that as the file size decreases due to compression, the decompression process may require more time.

$$T = \frac{m}{n}, \tag{3}$$

  where T is the compression ratio, m and n, respectively, denote the total number of timestamped points for the compressed and original AIS-based vessel trajectories.
- Dynamic Time Warping (DTW) [78]: The DTW, developed based on dynamic programming, is an effective method for reducing search comparison time. Specifically, the DTW algorithm aims to determine the shortest distance between two vectors. In the case of vectors g and h in n-dimensional space, their distance can be defined as the Euclidean distance, which is the linear distance between corresponding points on the two vectors, represented as $distance(g, h) = |g - h|$. However, if the lengths of the two vectors are not equal, the previous formula cannot be used to calculate the distance between them. To address this issue, the time attribute is assigned to the elements of these vectors. Ensuring proper alignment of the time axis is crucial because it may not be apparent how the elements of the two vectors correspond. Therefore, the classical and effective DTW algorithm, which can find the optimal correspondence, provides a valuable approach to solving these problems, When using the DTW algorithm to calculate trajectory similarity, the two trajectories can be regarded as aligned time series, and the distance between them can be calculated as the similarity index. It is important to note that trajectory data may exhibit deviations due to variations

TABLE 2 Existing Maritime Datasets, Which Contain Numerous Vessel Trajectories, Have Been Exploited for the Evaluation of Trajectory Data Computing and Mining

Name	Source/Publisher	Region	Cost	Tags
AIShub [70]	Crowdsource	Global, Coastal, Coverage, Map	Free	AIS open
MarineCadastre [71]	NOAA(National Oceanic and Atmospheric Administration.) Bureau of Ocean Energy Management	US Coastal	Free	USA government AIS
HELCOM [72]	Baltic Marine, Environment	Baltic Sea	Free	Europe government AIS
CruiseMappe [73]	CruiseMapper	World	Free	AIS cruise
MarineTraffic [74]	Marine Traffic	Global	Credit or Subscription	AIS private
ExactAIS Archive [75]	ExactAIS	Global	Credit or Subscription	AIS private
VesselFinder [76]	VesselFinder	Global, Coastal	Credit or Subscription	AIS private
FleetMon [77]	FleetMon	Global	Credit or Subscription	AIS private

The reference links where these AIS datasets are available can be found in REFERENCE.

in sampling accuracy. Thus, when performing DTW calculations on trajectories, it is necessary to consider the influence of the sampling interval.

- Distance Loss Rate (DLR): DLR quantifies the proportion of length lost in the compressed trajectory relative to the total length of the original trajectory. The compressed trajectory, with a smaller length than that of the original trajectory, is calculated using a reduced set of trajectory points.
  The trajectory length, denoted as $|X|$ in this paper, is an important parameter for evaluating the trajectory characteristics. The DLR can be applied to evaluate the compression efficiency and quality of the trajectory representation. The DLR facilitates a comprehensive assessment of the information loss associated with reducing the number of trajectory points.

$$|X| = \sum_{i=1}^{N-1} distance(p_i, p_{i-1}) \tag{4}$$

  We assume that the raw and compressed trajectory sets correspond to $X = \{p_0, p_1, \ldots, p_N\}$ and $Y = \{S_0, S_1, \ldots, S_m\}$, respectively. $length_loss(LL)$ denotes the difference from the total length of the original trajectory. The length loss rate is calculated as follows.

$$DLR = \frac{|X| - |Y|}{|X|}. \tag{5}$$

- Average Running Time: It is a critical metric employed to evaluate the efficiency and performance of algorithms. It quantifies the time or number of operations required for an algorithm to execute successfully. Time complexity and space complexity are common measures used to describe and analyze the running time of an algorithm. By considering these factors, researchers can gain insights into the algorithm's computational requirements.
  A timer integrated into the computer system is utilized in this study to obtain accurate measurements. The algorithm's execution time is measured by repeatedly running it multiple times in succession, and then averaging the obtained values. This approach ensures reliable results by accounting for potential variations in runtime due to external factors.
  In the context of C++, implementing high-precision timing is achievable by the utilization of libraries specifically designed for this purpose. These libraries enable precise measurement of algorithm execution time, facilitating in-depth analysis of algorithmic performance.

D. TRAJECTORY COMPRESSION EXPERIMENTS

Each trajectory compression algorithm uses different error metrics, processing methods, and logic. Applying these algorithms, compressed trajectories can be obtained for further comparative analysis of runtime and data quality. Ensuring a consistent standard for comparing different compression algorithms presents a challenge since each method has varying error thresholds for inputs. Consequently, even if the input thresholds are identical, direct comparisons at a single level may not be feasible.

Nonetheless, the compression rate for each trajectory can be adjusted by setting a threshold. In this way, the compression rate can be adjusted (except for adaptive algorithms), and the

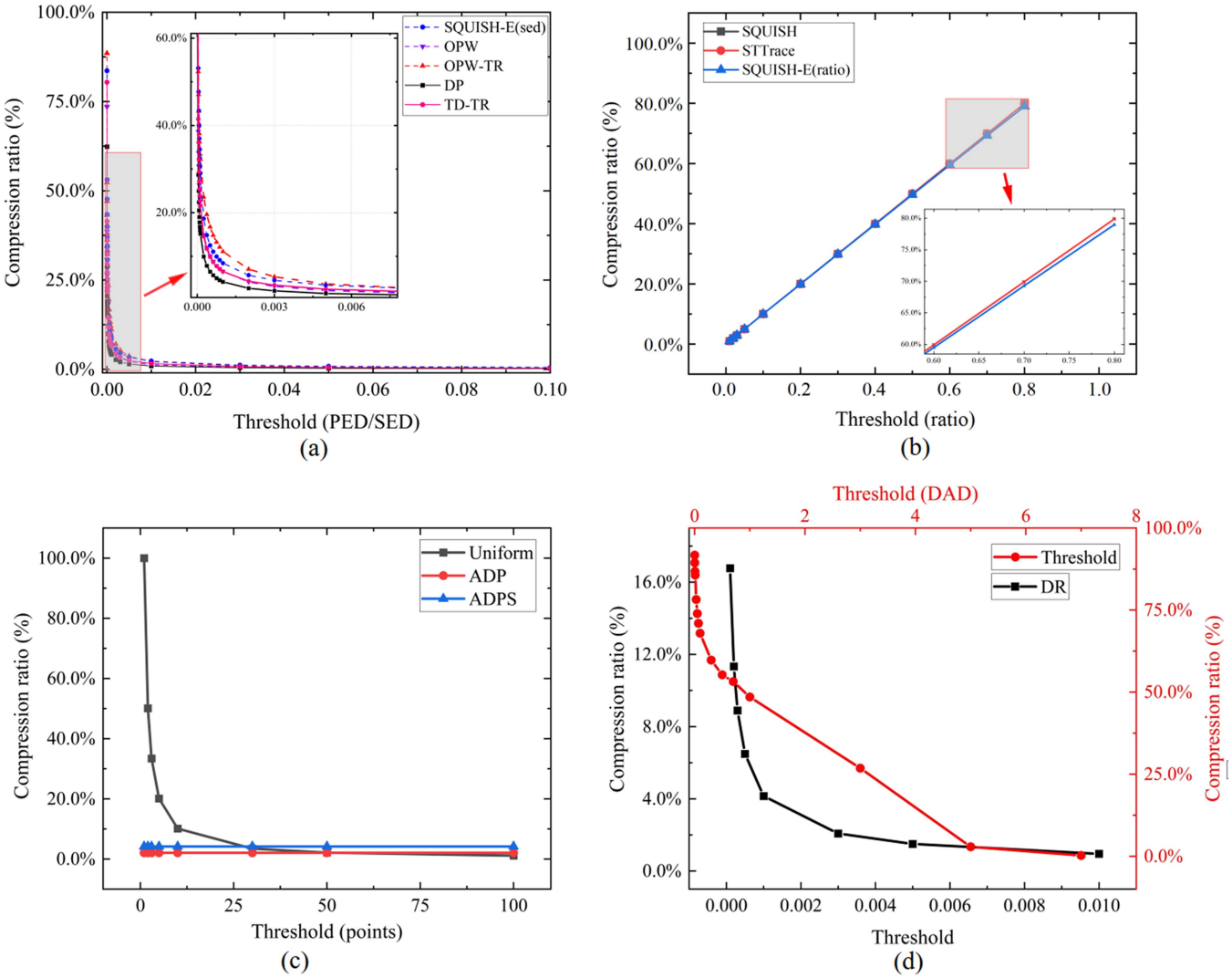

FIGURE 17. **Comparisons of compression ratios for different compression algorithms under different experimental scenarios.**

running time, DTW and DLR of various algorithms at different compression rates can be compared. Then, the advantages and disadvantages of different algorithms can be effectively evaluated and compared.

In our experiments, we utilized AIS data from the Marine Traffic dataset, specifically the data of 3148 vessels on June 30, 2022. The dataset contained a total of 7,239,758 trajectory points. To ensure data accuracy, long-stay trajectories and noisy data were eliminated, resulting in a selection of original trajectory data comprising AIS data from 2232 vessels and 2,704,676 trajectory points. This refined dataset formed the basis of our experimentation and evaluation.

1) COMPRESSION RATIO ANALYSIS

In the experiments, we evaluated the compression ratio using various compression thresholds. Apart from the adaptive compression algorithm, the different thresholds employed in other algorithms play a decisive role in determining the quality of compression. To explore this further, we examined three distinct categories of compression thresholds: SED/PED thresholds, compression ratio thresholds, and directional velocity thresholds. Each algorithm utilized a specific type of compression threshold based on its unique characteristics and requirements. By analyzing the results obtained for these different compression thresholds, we gain valuable insight into the effectiveness of the compression algorithms. This comprehensive evaluation can help analyze the strengths and weaknesses of each algorithm.

In Fig. 17(a), all five algorithms employ distance thresholds, such as PED and SED. Notably, the size of these thresholds exhibits an inverse relationship with the compression rate. Upon closer examination of the enlarged thumbnail, it becomes apparent that the OPW-TR algorithm achieves the highest compression rate, whereas the DP algorithm yields the lowest compression rate under the same threshold conditions.

Turning to Fig. 17(b), the three algorithms utilize compression rate thresholds. Here, we observe a proportional relationship between the size of the threshold and the

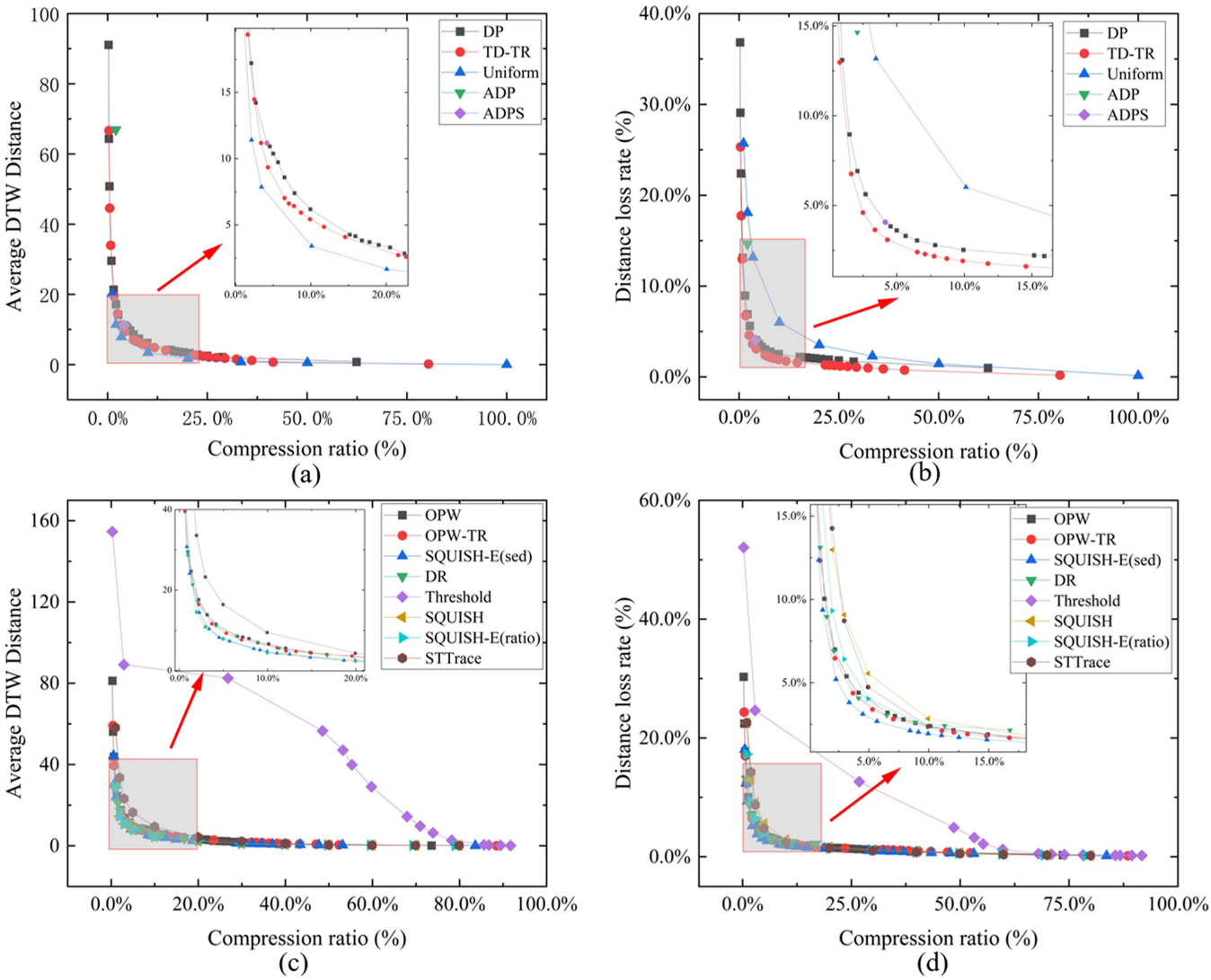

FIGURE 18. Comparisons of batch (offline) and online trajectory compression algorithms. The DTW distance and distance loss rate (DLR) are jointly selected to quantitatively evaluate the compression performance.

compression rate. Notably, in the thumbnail image, the SQUISH-E(ratio) algorithm's compression rate accuracy decreases as the threshold increases in size.

In Fig. 17(c), the remaining algorithms utilize the number of interval points as the threshold, representing the uniform algorithm. Notably, the ADP and ADPS algorithms do not require an input threshold. Consequently, we observe that the results of the Uniform algorithm follow a curve, while the results of the ADP and ADPS algorithms are depicted as straight lines.

Fig. 17(d) showcases directional thresholds. However, the specific definitions of these thresholds vary across algorithms. In this experiment, we set the speed threshold to 25 for the threshold algorithm, allowing us to investigate the impact of changing the directional threshold on the compression rate. Notably, we observe an inverse relationship between the direction threshold and the compression ratio for both algorithms.

2) COMPRESSION EFFECT EVALUATION

In the analysis of compression rates presented in Section V-D1, it is worth noting that each algorithm employs its unique compression threshold. Consequently, unifying these thresholds for a direct comparison of algorithmic effects becomes impractical. However, the compressed trajectories obtained by using each algorithm produce valuable metrics data and serve as reference points. Utilizing this information, we can compare the performance of different algorithms based on the original trajectories and the compressed trajectories, specifically focusing on metrics, such as DTW and DLR.

Furthermore, we conduct separate evaluations of batch compression algorithms and online compression algorithms to assess their individual compression effects. These evaluation results are illustrated in Fig. 18.

DTW distance is a measure of trajectory similarity. The smaller the DTW distance, the higher the similarity between

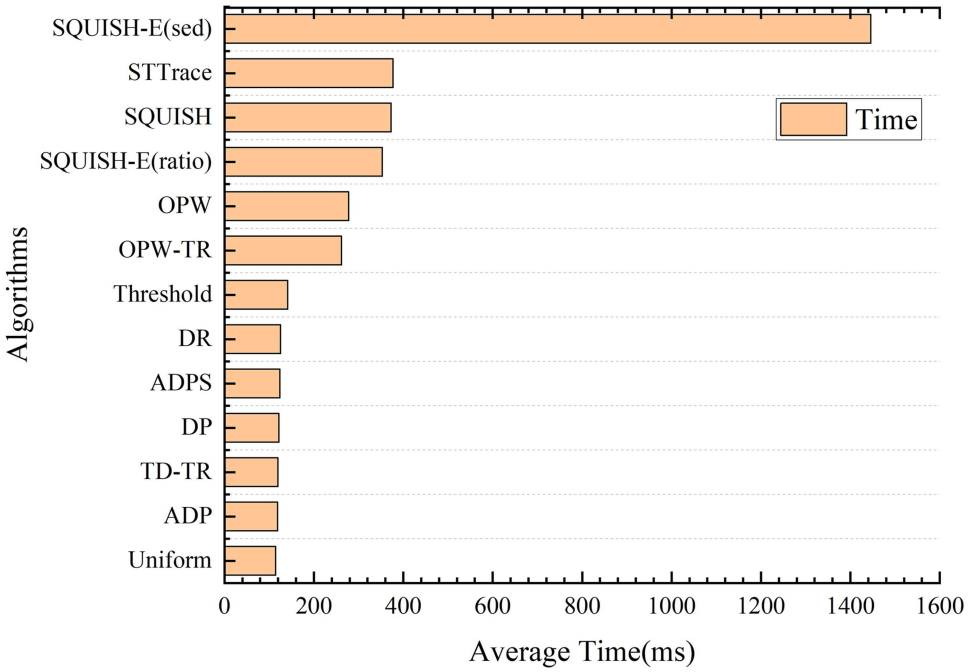

FIGURE 19. **Comparisons of computational time for different compression algorithms (ratio = 10).**

the original trajectory and the compressed trajectory. DLR distance is a quantification of the degree of trajectory distortion. The smaller the DLR distance, the lower the distortion between the original trajectory and the compressed trajectory.

Fig. 18(a) and (b) depict the effects of the offline compression algorithms, and Fig. 18(c) and (d) represent the effects of the online compression algorithms. From Fig. 18(a), we observe an inverse relationship between the compression rate and DTW distance. Under the same compression rate, the DTW distances follow the order: DP > TD-TR > Uniform. Notably, the Uniform algorithm yields the smallest DTW distance due to its retention of trajectory points through equal interval sampling. In contrast, other algorithms retain trajectory points based on thresholds related to distance and direction, resulting in uneven point retention and consequently larger DTW distances. This difference arises from the unique characteristics of each algorithm. In Fig. 18(b), the DLR value of the Uniform algorithm is significantly higher than that of other competing algorithms, indicating a poorer ability to preserve the shape of the trajectory. Additionally, we observe that the TD-TR algorithm outperforms the DP algorithm in both Fig. 18(a) and (b), suggesting that incorporating the time factor in the SED can improve trajectory preservation.

In Fig. 18(c), it can be found that the Threshold compression algorithm consistently exhibits the highest DTW values across all stages, while other online compression algorithms exhibit similar distribution patterns. Upon closer inspection, the STTrace algorithm consistently yields the highest DTW values, while the SQUISH, SQUISH-E (sed), and SQUISH (ratio) algorithms yield lower DTW values. In Fig. 18(d), the DLR value of the Threshold algorithm is also the highest among the online compression algorithms, indicating that it has the least favorable compression effect. Specifically, the SQUISH algorithm performs better, while the SQUISH-E (sed) algorithm performs less favorably. By considering both Fig. 18(c) and (d), we can conclude that the SQUISH-E (sed) algorithm achieves the best compression effect among the online compression algorithms.

3) COMPRESSION TIME ANALYSIS

In addition to compression ratio and compression effect, the running time of the compression algorithms is another important aspect to evaluate. The time performance of each compression algorithm is examined, as indicated in Fig. 19. The x-axis denotes the average time required by each algorithm to calculate individual AIS trajectories, while the y -axis represents the algorithms ranked in descending order based on their average time consumption.

Online algorithms generally exhibit longer running times compared to batch algorithms. In Fig. 19, the top eight algorithms represent online algorithms, while the bottom five correspond to offline algorithms. The algorithms can be roughly categorized into five tiers based on their running speeds. Among them, the SQUISH-E (sed) algorithm possesses the slowest running time, taking approximately 1445ms, whereas the Uniform algorithm demonstrates the fastest running time of only 114ms when compared to other algorithms. The second slowest echelon consists of the STTrace, SQUISH, and SQUISH-E (ratio) algorithms, followed by the speed balancing echelon encompassing OPW and OPW-TR algorithms. ADP, TD-TR, DP, ADPS, DR, and Threshold algorithms constitute the second fastest running echelon.

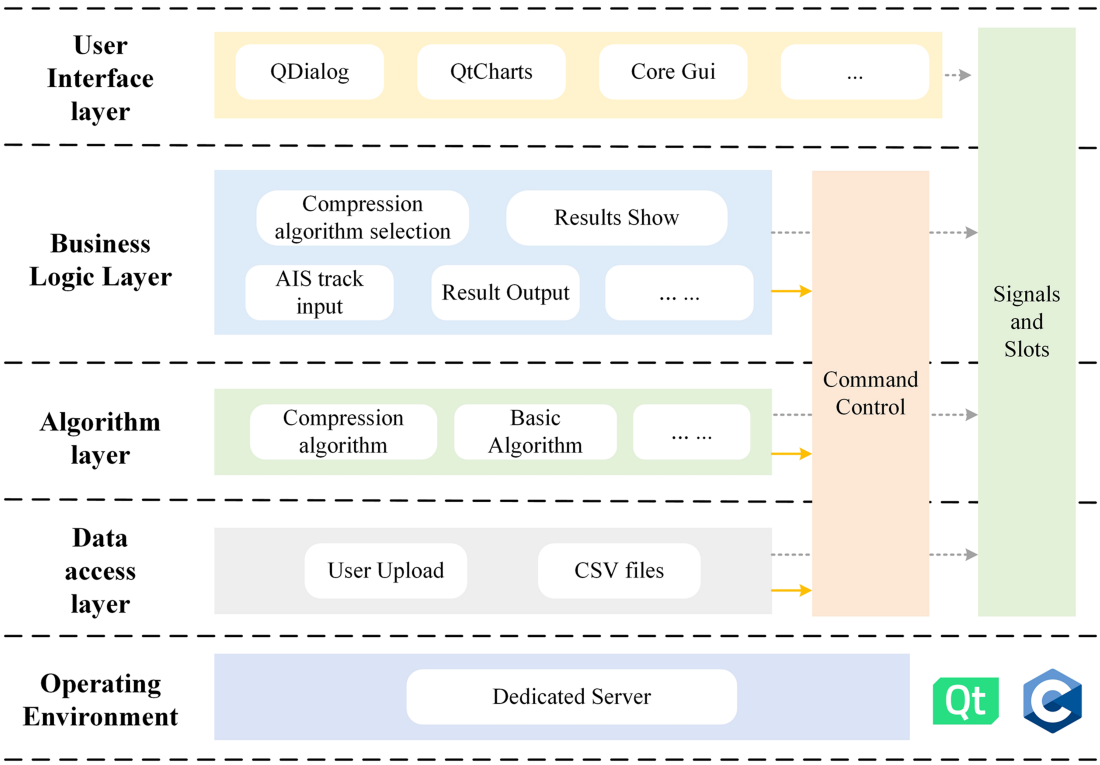

FIGURE 20. Design architecture of our AIS-based vessel trajectory compression software. This software has the capacity to handle both batch and online trajectory compression tasks.

VI. AISCOMPRESS: QT-BASED VESSEL TRAJECTORY COMPRESSION SOFTWARE

A. SOFTWARE ARCHITECTURE

The vessel trajectory compression software presented in this work primarily utilizes the classical three-layer framework of the QT system, comprising the User Interface (UI) layer, Business Logic Layer (BLL), and Data Access Layer (DAL). Furthermore, an algorithm layer is incorporated to facilitate data utilization and execute instructions from the business layer. This architectural design significantly enhances system development efficiency while ensuring robustness for system maintenance and accommodating flexible changes in requirements. Fig. 20 comprehensively illustrates the software framework composition.

The interface layer of the software adopts the QWidget class to construct the window interface, employing a control-based approach through QTableview and QtCharts. The signal and slot mechanism is utilized for information interaction, resulting in reduced coupling between system objects and facilitating rapid development, rendering, and page maintenance.

The business layer leverages the QT class library to carry out specific operations, primarily focusing on algorithm and data implementation, as well as core business logic. The well-encapsulated nature of the QT library enhances software modularity and reusability, greatly facilitating user development.

The algorithm layer utilizes C++ algorithms to execute specific operations and is primarily responsible for implementing business instructions and processing data. Powered by the C++ programming language, it exhibits strong scalability, enabling swift backend service development and API encapsulation.

The data layer is entrusted with managing the AIS data within the system, including data upload and deletion. The transmission file format follows CSV standards, defining the content of each column. This format enables fast operations, ensures cost-effectiveness, and provides user-friendly utilization.

B. SOFTWARE FEATURES

The software developed in this study encompasses five distinct functionalities: data input, path output, algorithm selection, interface display, and algorithm description. Users have the flexibility to customize input data and output paths, enabling them to select the most suitable algorithm for their needs. The software promptly presents the results on the main interface, allowing users to analyze and assess the algorithm's performance.

A broad spectrum of 12 compression algorithms can be implemented using the software. These algorithms encompass five batch compression techniques, namely Uniform, DP, TD-TR, ADP, and ADPS. Additionally, the software supports

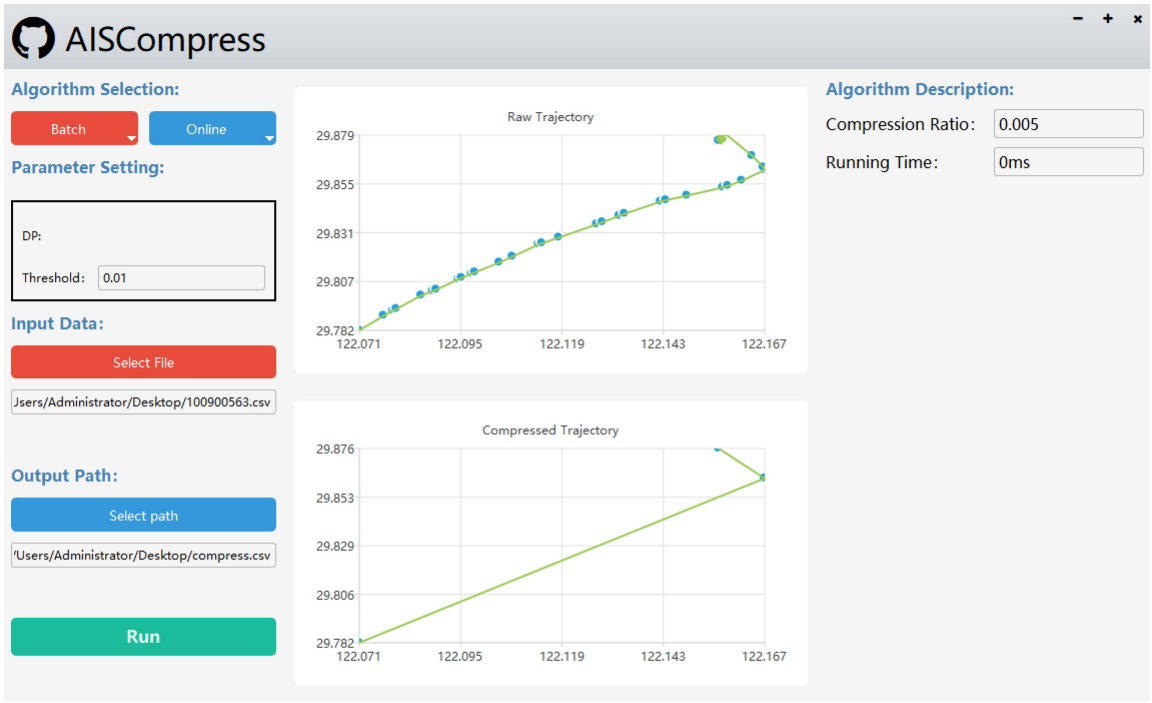

FIGURE 21. **Interface of visualizing the vessel trajectories before and after compression using our software *AISCompress*.**

seven online compression algorithms, namely OPW, OPW-TR, DR, SQUISH, SQUISH-E, Threshold, and STTrace. This comprehensive array of algorithms provides users with a variety of options to meet their trajectory compression needs.

VII. CONCLUSION AND FEATURE PERSPECTIVES

A. CONCLUSION

In this work, the progress made by scholars in trajectory compression algorithms in the past few decades is comprehensively overviewed. A compilation of prominent AIS databases worldwide, accompanied by relevant website links to facilitate further exploration by users is then presented.

Subsequently, comprehensive experiments on trajectory data compression using AIS trajectory data from the MarineCadastre database are conducted. Preliminary experimental results are analyzed, revealing that the TD-TR algorithm and SQUISH-E (sed) algorithm exhibit remarkable efficiency in compressing AIS trajectory data with varying thresholds. It is worth noting that the TD-TR algorithm proves to be particularly effective for batch processing, whereas the SQUISH-E (sed) algorithm performs well in online scenarios despite the longer computation time. The Uniform Algorithm and Threshold Algorithm are deemed unsuitable for maritime AIS trajectory data compression due to significant distance losses and large DTW matching distances.

Moreover, we introduce the OPW-TR algorithm, an improved version of the OPW algorithm, which exhibits a slightly lower compression rate but significantly reduces the distance loss rate. The ADP algorithms achieves well-balanced outcomes across various metrics and possess the advantage of adaptability, eliminating the need for threshold definition.

Finally, we develop and share an AIS-based vessel trajectory compression software on GitHub. This software mainly contains 12 different batch and online trajectory compression algorithms introduced in this work. Users can select the specific algorithm to conduct the trajectory data compression according to their needs. Batch algorithms are suitable for scenarios requiring high precision, and global optimization, such as processing historical data or conducting complex analyses. Online algorithms are better suited for handling real-time data streams, and situations requiring low-latency responses. This developed software can further enhance the convenience and accessibility for beginners or researchers involved in trajectory data compression.

B. FUTURE WORKS

In our future work, we aim to enhance the evaluation of compression effectiveness by incorporating additional measures. This will enable us to more effectively assess the compression performance of both online and batch compression algorithms under various constraints. In addition, we plan to develop trajectory compression algorithms that consider more feature information to ensure that the data features in the original trajectory are preserved as much as possible while compressing the trajectory data.

Furthermore, we intend to enhance the online trajectory compression algorithm by transitioning it into a distributed compression framework. By integrating it with a distributed trajectory database, we can directly store the original trajectory data in the distributed database after compression. This

approach holds significant potential for greatly improving compression and storage efficiency.

In summary, our future research directions in AIS trajectory compression will focus on several key aspects. The first goal is to improve compression performance, allowing for higher rates of data reduction while maintaining data integrity. Secondly, we seek to develop techniques that can effectively handle Big Data volumes associated with AIS trajectories. Thirdly, we aim to address challenges posed by noisy data to ensure reliable compression outcomes. Moreover, we plan to enhance the adaptivity of our algorithms to accommodate diverse data characteristics. Lastly, we aim to explore cross-domain applications, enabling the utilization of AIS trajectory data in various fields to meet the expanding demands of data processing tasks.

REFERENCES

[1] R. W. Liu et al., "STMGCN: Mobile edge computing-empowered vessel trajectory prediction using spatio-temporal multigraph convolutional network," *IEEE Trans. Ind. Inform.*, vol. 18, no. 11, pp. 7977–7987, Nov. 2022.

[2] P. Gloaguen, L. Chapel, C. Friguet, and R. Tavenard, "Scalable clustering of segmented trajectories within a continuous time framework: Application to maritime traffic data," *Mach. Learn.*, vol. 112, no. 6, pp. 1975–2001, 2023.

[3] X. Bai, Z. Ma, Y. Hou, Y. Li, and D. Yang, "A data-driven iterative multi-attribute clustering algorithm and its application in port congestion estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 12026–12037, Nov. 2023.

[4] D. S. Pedroche, J. G. Herrero, and J. M. M. López, "Context learning from a ship trajectory cluster for anomaly detection," *Neurocomputing*, vol. 563, 2024, Art. no. 126920.

[5] X. Zhang, X. Fu, Z. Xiao, H. Xu, and Z. Qin, "Vessel trajectory prediction in maritime transportation: Current approaches and beyond," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 19980–19998, Nov. 2022.

[6] P. Han, M. Zhu, and H. Zhang, "Interaction-aware short-term marine vessel trajectory prediction with deep generative models," *IEEE Trans. Ind. Informat.*, vol. 20, no. 3, pp. 3188–3196, Mar. 2024.

[7] E. Chondrodima, N. Pelekis, A. Pikrakis, and Y. Theodoridis, "An efficient LSTM neural network-based framework for vessel location forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 4872–4888, May 2023.

[8] L. Xie et al., "A novel model for ship trajectory anomaly detection based on Gaussian mixture variational autoencoder," *IEEE Trans. Veh. Technol.*, vol. 72, no. 11, pp. 13826–13835, Nov. 2023.

[9] J. Hu et al., "Intelligent anomaly detection of trajectories for IoT empowered maritime transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2382–2391, Feb. 2022.

[10] Y. Oh and S. Kim, "Grid-based Bayesian bootstrap approach for real-time detection of abnormal vessel behaviors from AIS data in maritime logistics," *IEEE Trans. Automat. Sci. Eng.*, early access, Nov. 07, 2023, doi: 10.1109/TASE.2023.3329041.

[11] Y. Zhang, Q. Jin, M. Liang, R. Ma, and R. W. Liu, "Vessel behavior anomaly detection using graph attention network," in *Proc. Int. Conf. Neural Inf. Process.*, 2023, pp. 291–304.

[12] M. Liang, L. Weng, R. Gao, Y. Li, and L. Du, "Unsupervised maritime anomaly detection for intelligent situational awareness using AIS data," *Knowl.-Based Syst.*, vol. 284, 2024, Art. no. 111313.

[13] C. A. G. Chávez et al., "Advancing sustainability through digital servitization: An exploratory study in the maritime shipping industry," *J. Cleaner Prod.*, vol. 436, 2024, Art. no. 140401.

[14] T. Wang, P. Cheng, and L. Zhen, "Green development of the maritime industry: Overview, perspectives, and future research opportunities," *Transp. Res. Part E: Logistics Transp. Rev.*, vol. 179, 2023, Art. no. 103322.

[15] Y. Xiao, X. Li, W. Yao, J. Chen, and Y. Hu, "Bidirectional data-driven trajectory prediction for intelligent maritime traffic," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 1773–1785, Feb. 2022.

[16] N. U. I. Hossain, N. Sakib, and K. Govindan, "Assessing the performance of unmanned aerial vehicle for logistics and transportation leveraging the Bayesian network approach," *Expert Syst. With Appl.*, vol. 209, 2022, Art. no. 118301.

[17] M. Marino et al., "New frontiers in the risk assessment of ship collision," *Ocean Eng.*, vol. 274, 2023, Art. no. 113999.

[18] Z. Xiao, X. Fu, L. Zhang, and R. S. M. Goh, "Traffic pattern mining and forecasting technologies in maritime traffic service networks: A comprehensive survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1796–1825, May 2019.

[19] A. Nurfalah, S. H. Supangkat, and E. Mulyana, "Real-time AIS anomaly detection in maritime tactical data system using AIS/radar track association approach," in *Proc. IEEE 2023 10th Int. Conf. ICT Smart Soc.*, 2023, pp. 1–6.

[20] M. Liang, J. Su, R. W. Liu, and J. S. L. Lam, "AISClean: AIS data-driven vessel trajectory reconstruction under uncertain conditions," *Ocean Eng.*, vol. 306, 2024, Art. no. 117987.

[21] M. Robards et al., "Conservation science and policy applications of the marine vessel automatic identification system (AIS)—A review," *Bull. Mar. Sci.*, vol. 92, no. 1, pp. 75–103, 2016.

[22] P. Bernabé, A. Gotlieb, B. Legeard, D. Marijan, F. O. Sem-Jacobsen, and H. Spieker, "Detecting intentional AIS shutdown in open sea maritime surveillance using self-supervised deep learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 2, pp. 1166–1177, Feb. 2024.

[23] R. W. Liu, W. Zheng, and M. Liang, "Spatio-temporal multi-graph transformer network for joint prediction of multiple vessel trajectories," *Eng. Appl. Artif. Intell.*, vol. 129, 2024, Art. no. 107625.

[24] 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, 2012.

[25] J. S. Kim, H. J. Park, W. Shin, D. Park, and S. W. Han, "WAY: Estimation of vessel destination in worldwide AIS trajectory," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 59, no. 5, pp. 5961–5977, May 2023.

[26] R. Al-Zaidi, J. C. Woods, M. Al-Khalidi, and H. Hu, "Building novel VHF-based wireless sensor networks for the internet of marine things," *IEEE Sensors J.*, vol. 18, no. 5, pp. 2131–2144, May 2018.

[27] 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.

[28] R. W. Liu, M. Liang, J. Nie, W. Y. B. Lim, Y. Zhang, and M. Guizani, "Deep learning-powered vessel trajectory prediction for improving smart traffic services in maritime Internet of Things," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3080–3094, May 2022.

[29] E. Tu, G. Zhang, L. Rachmawati, E. Rajabally, and G.-B. Huang, "Exploiting AIS data for intelligent maritime navigation: A comprehensive survey from data to methodology," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1559–1582, May 2017.

[30] L. Chen, S. Yu, Q. Chen, S. Li, X. Chen, and Y. Zhao, "5S: Design and in-orbit demonstration of a multi-functional integrated satellite-based Internet of Things payload," *IEEE Internet Things J.*, vol. 11, no. 7, pp. 12864–12873, Apr. 2024.

[31] X. Li, W. Feng, J. Wang, Y. Chen, N. Ge, and C.-X. Wang, "Enabling 5G on the ocean: A hybrid satellite-UAV-terrestrial network solution," *IEEE Wireless Commun.*, vol. 27, no. 6, pp. 116–121, Jun. 2020.

[32] D. Yang, L. Wu, S. Wang, H. Jia, and K. X. Li, "How Big Data enriches maritime research–a critical review of automatic identification system (AIS) data applications," *Transport Rev.*, vol. 39, no. 6, pp. 755–773, 2019.

[33] Z. Yan et al., "Exploring AIS data for intelligent maritime routes extraction," *Appl. Ocean Res.*, vol. 101, 2020, Art. no. 102271.

[34] 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.*, 2019, pp. 1–7.

[35] 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. 6, pp. 2218–2245, 2013.

[36] K. Patroumpas, E. Alevizos, A. Artikis, M. Vodas, N. Pelekis, and Y. Theodoridis, "Online event recognition from moving vessel trajectories," *GeoInformatica*, vol. 21, pp. 389–427, 2017.

[37] F. Xiao, H. Ligteringen, C. Van Gulijk, and B. Ale, "Comparison study on AIS data of ship traffic behavior," *Ocean Eng.*, vol. 95, pp. 84–93, 2015.
[38] Y. Zhang, A. Zhang, D. Zhang, Z. Kang, and Y. Liang, "Design and development of maritime data security management platform," *Appl. Sci.*, vol. 12, no. 2, pp. 1–18, 2022.
[39] 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.
[40] A. Makris, I. Kontopoulos, P. Alimisis, and K. Tserpes, "A comparison of trajectory compression algorithms over AIS data," *IEEE Access*, vol. 9, pp. 92516–92530, 2021.
[41] D. H. Douglas and T. 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, pp. 112–122, 1973.
[42] N. Meratnia and R. A. de By, "Spatiotemporal compression techniques for moving point objects," in *Proc. Adv. Database Technol.-EDBT 2004: 9th Int. Conf. Extending Database Technol.*, Heraklion, Crete, Greece, 2004, pp. 765–782.
[43] 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.
[44] H. Li et al., "Unsupervised hierarchical methodology of maritime traffic pattern extraction for knowledge discovery," *Transp. Res. Part C: Emerg. Technol.*, vol. 143, 2022, Art. no. 103856.
[45] Z. Wei, X. Xie, and X. Zhang, "AIS trajectory simplification algorithm considering ship behaviours," *Ocean Eng.*, vol. 216, 2020, Art. no. 108086.
[46] L. Zhao and G. Shi, "A method for simplifying ship trajectory based on improved Douglas–Peucker algorithm," *Ocean Eng.*, vol. 166, pp. 37–46, 2018.
[47] C. Tang, H. Wang, J. Zhao, Y. Tang, H. Yan, and Y. Xiao, "A method for compressing AIS trajectory data based on the adaptive-threshold Douglas-Peucker algorithm," *Ocean Eng.*, vol. 232, 2021, Art. no. 109041.
[48] R. Yan, H. Mo, D. Yang, and S. Wang, "Development of denoising and compression algorithms for AIS-based vessel trajectories," *Ocean Eng.*, vol. 252, 2022, Art. no. 111207.
[49] 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, 2019.
[50] T. Balcer, R. Szlapczynski, and T. Mestl, "Impact of trajectory simplification methods on modeling carbon dioxide emissions from ships," *Ocean Eng.*, vol. 305, 2024, Art. no. 117905.
[51] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proc. 2001 IEEE Int. Conf. Data Mining*, 2001, pp. 289–296.
[52] 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*, 2006, pp. 19–26.
[53] M. Potamias, K. Patroumpas, and T. Sellis, "Sampling trajectory streams with spatiotemporal criteria," in *Proc. IEEE 18th Int. Conf. Sci. Stat. Database Manage.*, 2006, pp. 275–284.
[54] J. Muckell, J.-H. Hwang, V. Patil, C. T. Lawson, F. Ping, and S. Ravi, "SQUISH: An online approach for GPS trajectory compression," in *Proc. 2nd Int. Conf. Comput. Geospatial Res. Appl.*, 2011, pp. 1–8.
[55] J. Muckell, P. W. Olsen, J.-H. Hwang, C. T. Lawson, and S. Ravi, "Compression of trajectory data: A comprehensive evaluation and new approach," *GeoInformatica*, vol. 18, pp. 435–460, 2014.
[56] 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, 1973.
[57] C. Chen, Y. Ding, X. Xie, S. Zhang, Z. Wang, and L. Feng, "Trajcompressor: An online map-matching-based trajectory compression framework leveraging vehicle heading direction and change," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 2012–2028, May 2019.
[58] A. Nibali and Z. He, "Trajic: An effective compression system for trajectory data," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 11, pp. 3138–3151, Nov. 2015.
[59] M. Liang, R. W. Liu, R. Gao, Z. Xiao, X. Zhang, and H. Wang, "A survey of distance-based vessel trajectory clustering: Data preprocessing, methodologies, applications, and experimental evaluation," 2024, *arXiv:2407.11084*.
[60] Y. Li, H. Li, C. Zhang, Y. Zhao, and Z. Yang, "Incorporation of adaptive compression into a GPU parallel computing framework for analyzing large-scale vessel trajectories," *Transp. Res. Part C: Emerg. Technol.*, vol. 163, 2024, Art. no. 104648.
[61] 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.
[62] W. Cao and Y. Li, "DOTS: An online and near-optimal trajectory simplification algorithm," *J. Syst. Softw.*, vol. 126, pp. 34–44, 2017.
[63] Z. Liu et al., "An online method for ship trajectory compression using AIS data," *J. Navigation*, pp. 1–22, 2024.
[64] N. Yu, X. Zhan, S. Zhao, Y. Wu, and R. Feng, "A precise dead reckoning algorithm based on bluetooth and multiple sensors," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 336–351, Jan. 2017.
[65] L. Ma, G. Shi, W. Li, and D. Jiang, "A direction-preserved vessel trajectory compression algorithm based on open window," *J. Mar. Sci. Eng.*, vol. 11, no. 12, 2023, Art. no. 2362.
[66] R. Skulstad, G. Li, T. I. Fossen, B. Vik, and H. Zhang, "Dead reckoning of dynamically positioned ships: Using an efficient recurrent neural network," *IEEE Robot. Automat. Mag.*, vol. 26, no. 3, pp. 39–51, Mar. 2019.
[67] R. Diamant and Y. Jin, "A machine learning approach for dead-reckoning navigation at sea using a single accelerometer," *IEEE J. Ocean. Eng.*, vol. 39, no. 4, pp. 672–684, Apr. 2013.
[68] 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*, vol. 25, no. 4, pp. 679–711, 2021.
[69] AIShub, "Aishub: Automatic identification system data sharing," Accessed: Jul. 10, 2023. [Online]. Available: https://www.aishub.net
[70] MarineCadastre, "Marinecadastre: National oceanographic and atmospheric administration marine cadastre," Accessed: Jul. 10, 2023. [Online]. Available: https://marinecadastre.gov
[71] HELCOM, "Helcom: Baltic marine environment protection commission - helsinki commission," Accessed: Jul. 10, 2023. [Online]. Available: https://www.helcom.fi
[72] CruiseMappe, "Cruisemappe: Interactive cruise mapping and planning," Accessed: Jul. 10, 2023. [Online]. Available: https://www.cruisemappe.com
[73] MarineTraffic, "Marinetraffic: Global ship tracking intelligence," Accessed: Jul. 10, 2023. [Online]. Available: https://www.marinetraffic.com
[74] E. Archive, "Exactais archive: Historical ais data," Accessed: Jul. 10, 2023. [Online]. Available: https://www.example.com/exactaisarchive
[75] VesselFinder, "Vesselfinder: Ais vessel tracking and maritime intelligence," Accessed: Jul. 10, 2023. [Online]. Available: https://www.vesselfinder.com/
[76] FleetMon, "Fleetmon: Global vessel tracking and maritime intelligence," Accessed: Jul. 10, 2023. [Online]. Available: https://www.fleetmon.com/
[77] F. Zhu and Z. Ma, "Ship trajectory online compression algorithm considering handling patterns," *IEEE Access*, vol. 9, pp. 70182–70191, 2021.
[78] J. Taylor, X. Zhou, N. M. Rouphail, and R. J. Porter, "Method for investigating intradriver heterogeneity using vehicle trajectory data: A dynamic time warping approach," *Transp. Res. Part B: Methodological*, vol. 73, pp. 59–80, 2015.