Ocean and Coastal Management 284 (2027) 108440



Contents lists available at ScienceDirect

Ocean and Coastal Management

journal homepage: www.elsevier.com/locate/ocecoaman



Enhancing vessel trajectory similarity measurement for maritime traffic monitoring using dual-component dynamic time warping

Yan Li [a], Lan Song [b,c,*], Bi Yu Chen [a], William Luocheng Wu [d], Zaili Yang [e], Huanhuan Li [f]

[a] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan, China
[b] Institute of Environmental Science (CML), Leiden University, the Netherlands
[c] College of Engineering, Eastern Institute of Technology, Ningbo, China
[d] Institute of Sound and Vibration Research (ISVR), University of Southampton, Southampton, UK
[e] Liverpool Logistics, Offshore and Marine (LOOM) Research Institute, Liverpool John Moores University, Liverpool, UK
[f] School of Engineering, University of Southampton, Southampton, UK

ARTICLE INFO

Keywords:
Vessel trajectory similarity
AIS data
Maritime traffic analysis
Vessel behavior monitoring
Navigational safety assessment

ABSTRACT

The Automatic Identification System (AIS) data provide valuable information for maritime traffic monitoring, vessel behavior analysis, and navigational safety assessment. Accurate measurement of vessel trajectory similarity is fundamental to these applications. However, most existing Dynamic Time Warping (DTW)-based methods apply a uniform alignment strategy to all trajectory points, overlooking the fact that vessel movement patterns are primarily determined by a limited number of structurally important points, while many other points mainly preserve trajectory continuity. As a result, conventional approaches may produce biased similarity measurements, particularly when trajectories exhibit similar overall movement patterns but differ in sampling density. To address this limitation, this study proposes a Dual-Component Dynamic Time Warping (DCDTW) framework for vessel trajectory similarity measurement. The proposed method decomposes vessel trajectories into feature points and non-feature points and applies differentiated similarity measurement strategies to each component according to their distinct contributions to trajectory structure and navigational behavior. By explicitly accounting for the heterogeneous importance of trajectory points, DCDTW better preserves trajectory shape characteristics and movement patterns than conventional DTW-based approaches. To support large-scale AIS data analysis, the proposed framework is further implemented within a Graphics Processing Unit (GPU)-based parallel computing environment, substantially improving computational efficiency. The performance of DCDTW is evaluated using real-world AIS trajectory datasets collected from Caofeidian Port and the Chengshan Jiao Promontory, two representative maritime regions with complex traffic conditions. Experimental results demonstrate that DCDTW consistently achieves higher trajectory similarity measurement accuracy than conventional DTW-based methods, while the GPU-accelerated implementation significantly reduces computational time for large-scale applications. The proposed framework provides an effective tool for identifying vessel behavior, analyzing maritime traffic patterns, detecting anomalies, and assessing navigational safety. By improving the reliability and scalability of trajectory similarity measurement, this study contributes to the development of data-driven approaches for maritime traffic management and safety governance in complex coastal and port environments.

* Corresponding author. Institute of Environmental Science (CML), Leiden University, the Netherlands.
E-mail address: l.song@cml.leidenuniv.nl (L. Song).

https://doi.org/10.1016/j.ocecoaman.2026.108440
Received 10 July 2026; Received in revised form 5 October 2026; Accepted 7 November 2026
Available online 13 November 2026
0964-5691/© 2026 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Nomenclature

Roman letters			
Variable	Definition	Variable	Definition
AIS	Automatic Identification System	DBI	Davies-Bouldin Index
ACDTW	Adaptive Constrained DTW	eDTW	EventDTW
ADP	Adaptive DP	GPU	Graphics Processing Unit
ADPS	ADP with Speed	GPGPU	General-Purpose GPU
ADPSC	ADP with Speed and Course	ITS	Intelligent Transportation Systems
COG	Course Of Ground	IoT	Internet of Things
CDC-DTW	Context Density Consistency DTW	MPDP	Multi-objective Peak DP
CPU	Central Processing Unit	MDS	Multidimensional Scaling
CUDA	Compute Unified Device Architecture	OPW	Opening Window
CFDP	Caofeidian Port	SOG	Speed Of Ground
CSJP	Chengshan Jiao Promontory	SCAADP	SOG- and COG- Aware Adaptive DP
CR	Compression Ratio	SW-DTW	Sliding Window DTW
CHI	Calinski-Harabasz Index	SQUISH	Spatial Quality Simplification Heuristic
DTW	Dynamic Time Warping	SR	Speedup Ratio
DCDTW	Dual-Component DTW	SC	Silhouette Coefficient
DDTW	Derivative DTW	TD	Triangulation Division
DDDTW	Derivative Distance DTW	TD-TR	Top-Down Time-Ratio
DP	Douglas-Peucker	TGS	Threshold-Guided Sampling
DR	Dead Reckoning	TP-DTW	Trajectory Points-based DTW
DBSCAN	Density-Based Spatial Clustering of Applications with Noise	US	Uniform Sampling

1. Introduction

The continued expansion of global trade has reinforced maritime transport as a backbone of international logistics, particularly for long-distance and intercontinental freight movement. Enhancing the efficiency, safety, and intelligence of maritime transportation systems has therefore become a priority in both academic research and operational practice (Li et al., 2024). During operation, modern vessels continuously transmit large volumes of navigational data, including positional information, speed, and heading, through the Automatic Identification System (AIS) (H. Li et al., 2023). The widespread availability of AIS data has enabled large-scale empirical studies of vessel movement patterns, traffic organization, and navigational behavior, providing a data-driven foundation for advanced maritime transport analysis (Peng et al., 2025; Yang et al., 2025).

Among the various analytical tasks enabled by AIS data, vessel trajectory similarity measurement plays a fundamental role. It supports a wide range of maritime applications, including traffic pattern analysis, route classification, vessel behavior monitoring, and anomaly detection (Xing et al., 2023). By quantifying the similarity between vessel trajectories, analysts can identify typical navigation patterns (Salvador and Chan, 2007), detect abnormal routes (Zhang et al., 2025), and group vessels with comparable movement behaviors, thereby supporting operational decision-making and maritime safety management.

Reliable vessel trajectory similarity measurement is relevant to a range of maritime traffic analysis and management tasks (Ribeiro et al., 2023; Tu et al., 2018). At the traffic-pattern level, comparison and clustering of AIS trajectories can support the identification of recurrent navigation routes, traffic corridors, and characteristic vessel movement patterns, providing useful information for understanding the organisation and use of navigational space (Rong et al., 2020). At the individual-trajectory level, comparison with historical or representative patterns can help identify trajectories that differ substantially from commonly observed behaviour and therefore warrant further examination (Pallotta et al., 2013). Trajectory similarity information may also serve as one analytical input to downstream applications such as anomaly screening, collision-risk analysis, and intelligent maritime surveillance, where vessel movement patterns need to be interpreted together with additional contextual information (Ribeiro et al., 2023). Improved trajectory similarity measurement can therefore provide a more informative basis for subsequent maritime traffic analysis and decision-support tasks. Nevertheless, these downstream applications require task-specific models and validation and are not directly evaluated in the present study.

Dynamic Time Warping (DTW) has been widely adopted as a benchmark technique for trajectory similarity measurement due to its ability to accommodate temporal misalignment between sequences (Lee and Leung, 2023). However, existing DTW-based approaches typically apply a uniform alignment strategy to all trajectory points through a global two-dimensional matching process. This assumption is ill-suited to vessel trajectories, whose overall motion patterns are primarily governed by a limited number of characteristic points, such as turning points and course-change locations. While most trajectory points ensure continuity of movement, they contribute relatively little to the overall trajectory shape. Consequently, conventional DTW-based methods may produce distorted similarity assessments, particularly when trajectories share similar global shapes but differ substantially in sampling density. Notably, several enhanced DTW variants proposed in the literature, including EventDTW (eDTW) (Jiang et al., 2020), Adaptive Constrained DTW (ACDTW) (Li et al., 2020), and Context Density Consistency DTW (CDC-DTW) (W. Li et al., 2023), remain constrained by this uniform treatment of trajectory points and therefore fail to fully address this structural limitation.

To overcome these challenges, this study proposes a Dual-Component Dynamic Time Warping (DCDTW) method for vessel trajectory similarity measurement. The core idea is to explicitly recognize the heterogeneous contributions of trajectory points by decomposing each trajectory into feature points and non-feature points, and by designing distinct similarity measurement strategies for each component. Feature points capture the macro-level geometric and behavioral structure of vessel motion, while non-feature points characterise local continuity along the trajectory. By integrating these two components, DCDTW provides a more faithful representation of vessel trajectory similarity than conventional DTW-based approaches.

A key prerequisite for implementing DCDTW is the reliable identification of trajectory feature points. Trajectory compression algorithms are naturally suited to this task, as their objective is to preserve essential geometric and kinematic characteristics with a reduced set of representative points. Although the Douglas-Peucker (DP) algorithm is widely used for trajectory compression, its reliance on manually defined thresholds limits its adaptability across diverse navigation scenarios (Zhang et al., 2024). Existing adaptive DP variants primarily focus on geometric deviation while largely neglecting dynamic navigational information, such as changes in speed and course (Li et al., 2024b; Liu et al., 2019; Zhao and Shi, 2018). To address this limitation, this study develops a Speed- and Course-Aware Adaptive Douglas-Peucker (SCAADP) algorithm. The algorithm automatically determines trajectory-specific compression thresholds by incorporating both geometric and kinematic characteristics, thereby enabling accurate and robust feature point extraction.

In addition to accuracy, computational efficiency is a critical consideration for trajectory similarity measurement, particularly when processing large-scale AIS datasets. Although several studies have proposed approximations to reduce the computational cost of DTW, most existing methods, such as FastDTW (Wu and Keogh, 2022) and SparseDTW (Al-Naymat et al., 2012), remain constrained by CPU-based serial computing frameworks. To improve scalability, this study further implements both SCAADP and DCDTW within a Graphics Processing Unit (GPU)-based parallel computing framework, substantially

accelerating similarity computation and enabling practical application to high-volume maritime data. Although this study focuses on maritime traffic, the proposed trajectory similarity framework is formulated in a generic manner and is potentially applicable to other transport modes.

The main contributions of this study are summarized as follows:

(1) A novel DCDTW framework is proposed for vessel trajectory similarity measurement. Rather than assigning equal importance to all trajectory points, DCDTW distinguishes between feature and non-feature points according to their structural significance. This dual-component representation enables the similarity measure to place greater emphasis on trajectory segments that characterise meaningful changes in vessel movement, thereby providing a more informative representation of navigational patterns and behaviour.
(2) A SCAADP algorithm is developed for trajectory feature-point extraction. In contrast to conventional Douglas–Peucker methods and adaptive variants that predominantly rely on geometric characteristics, SCAADP incorporates variations in both Speed Over Ground (SOG) and Course Over Ground (COG) when determining trajectory-specific compression thresholds. This allows the feature-extraction process to account for both the geometric structure and motion characteristics of vessel trajectories.
(3) A GPU-based parallel computing framework is developed to improve the computational scalability of the proposed methodology for large-scale AIS datasets. The parallelization strategy covers the principal computational stages, from feature-point extraction to dual-component trajectory similarity computation, thereby reducing the computational burden associated with large-scale pairwise trajectory analysis. This integrated implementation facilitates the application of the proposed methodology to computationally intensive maritime applications, including traffic-pattern analysis and vessel-behaviour analysis in complex waterways.

The remainder of this paper is organized as follows. Section 2 reviews related work on trajectory similarity measurement, trajectory compression, and computational optimization. Section 3 introduces the datasets, preprocessing procedures, and basic definitions. Section 4 presents the proposed SCAADP and DCDTW methods, along with their GPU-based implementations. Section 5 reports experimental results and comparative analyses. Finally, Section 6 concludes the paper and outlines directions for future research.

2. Literature review

This study builds upon existing research on trajectory similarity measurement, trajectory compression, and computational optimization to improve the accuracy and scalability of vessel trajectory analysis. Accordingly, this section reviews related work in four areas: trajectory similarity measurement methods, DTW and its variants, trajectory compression techniques for feature point extraction, and computational acceleration for large-scale trajectory analysis.

2.1. Review of studies on similarity measurement methods

Trajectory similarity measurement has attracted considerable attention due to its importance in maritime traffic analysis, vessel behavior monitoring, anomaly detection, and route analysis. Existing approaches can be broadly categorized into region-based, shape-based, and point-based methods.

Region-based approaches assess similarity by partitioning the spatial domain into predefined regions and comparing aggregated trajectory representations (Cignoni et al., 1998). Representative examples include triangulation-based methods, which offer relatively low computational complexity but are highly sensitive to spatial partitioning schemes and often fail to preserve detailed trajectory structure.

Shape-based methods, such as the Hausdorff distance (Taha and Hanbury, 2015) and Fréchet distance (Lyu et al., 2022), focus on the geometric similarity between trajectories. The Hausdorff distance provides computational efficiency by capturing global geometric differences, but it is generally insensitive to local structural variations along trajectories. In contrast, the Fréchet distance explicitly considers the continuity and ordering of trajectory points, offering a more faithful representation of path similarity. However, this advantage comes at the cost of high computational complexity and strong sensitivity to noise, which restricts its use in large-scale or noisy trajectory datasets.

Point-based methods compute similarity by comparing individual trajectory points, typically using Euclidean Distance. Traditional point-to-point measures require trajectories to be of equal length and are therefore unsuitable for most real-world vessel trajectories. DTW overcomes this limitation by allowing non-linear temporal alignment between trajectories of unequal length, making it one of the most widely adopted similarity measures for moving object trajectories in maritime applications.

Despite its flexibility, DTW is computationally expensive, which limits its scalability for large AIS datasets. To improve efficiency, several variants have been proposed, including FastDTW (Wu and Keogh, 2022), which applies a multi-resolution alignment strategy, and SparseDTW (Al-Naymat et al., 2012), which exploits sparsity in the distance matrix. Although these methods reduce computational burden, they remain time-consuming when applied to long trajectories or large-scale datasets. In addition, DTW is sensitive to noise (Górecki and Łuczak, 2013), prompting the development of derivative-based (Muscillo et al., 2007), sliding-window (Folgado et al., 2018), event-driven (Jiang et al., 2020), and adaptive constrained variants (Li et al., 2020) to improve robustness and alignment quality under specific conditions (W. Li et al., 2023).

Recent advances in trajectory similarity analysis have increasingly adopted learning-based approaches (Chang et al., 2024). Deep representation-learning methods, including contrastive-learning methods such as TrajCL, encode spatial and structural trajectory characteristics into compact embeddings, allowing similarity to be evaluated in a learned representation space (Chang et al., 2023; Zhu et al., 2026). Graph-based approaches incorporate relational or network structure into trajectory representations, which can be particularly useful when movement is constrained by an explicit transportation network (S. Zhou et al., 2023; Zhu et al., 2026). More recently, attention- and Transformer-based architectures have been employed to capture long-range dependencies and contextual relationships within trajectory sequences (S. Li et al., 2023; Liu et al., 2024; Xue et al., 2024). Once suitable representations have been learned, these approaches can support efficient repeated similarity queries over large trajectory collections (Chang et al., 2023; Xue et al., 2024).

DCDTW differs from these learning-based approaches in both representation and similarity computation. Rather than learning a latent embedding from training data, DCDTW operates directly on the observed trajectory information. SCAADP first distinguishes feature and non-feature points according to geometric and kinematic changes, after which DCDTW applies component-specific comparison strategies and preserves explicit correspondence between matched trajectory elements. Consequently, the resulting distance can be related directly to the underlying trajectory structure and movement characteristics rather than being determined solely in a learned latent space.

These two methodological paradigms have different application characteristics (Liu et al., 2024; Sousa et al., 2020). Learning-based methods can be advantageous when large trajectory datasets are available for representation learning and when efficient repeated retrieval from large databases is required (Xue et al., 2024). In comparison, DTW-based methods such as DCDTW are applicable without a separate model-training stage and retain explicit trajectory-alignment information (Hu et al., 2023). These properties can be useful when training data

are limited, when the method needs to be transferred to a new AIS dataset without training a task-specific similarity model, or when transparent point- or component-level correspondence is important for subsequent interpretation. DCDTW should therefore be viewed as a complementary trajectory-similarity approach rather than a replacement for learning-based methods.

Despite these methodological advances, conventional DTW-based approaches and their variants still generally apply uniform point-wise alignment without explicitly distinguishing the structural roles of different trajectory points. This assumption presents a fundamental limitation for vessel trajectory analysis. In practice, vessel movement patterns are largely governed by a limited number of feature points, such as turning points and course changes points, whereas many points primarily maintain trajectory continuity. Consequently, existing DTW-based methods may yield distorted similarity results, particularly when trajectories exhibit similar global shapes but differ substantially in point density.

To address this limitation, this study proposes a DCDTW framework that explicitly distinguishes between feature and non-feature points in similarity measurement. Furthermore, a GPU-based parallel implementation is developed to enhance computational efficiency, enabling scalable trajectory similarity analysis for large maritime datasets.

2.2. Review of studies on trajectory compression methods

Trajectory compression plays a critical role in trajectory analysis by reducing data volume while preserving essential geometric and kinematic characteristics. In the context of trajectory similarity measurement, compression is not merely a data reduction process but a means of identifying feature points that capture key navigation behaviors, such as heading changes and speed variations. Effective compression methods therefore provide a foundation for reliable similarity analysis by retaining structurally important points and removing redundant observations. Existing trajectory compression techniques can be broadly divided into offline (batch) and online methods. This section focuses on offline approaches, which are particularly relevant for large-scale historical AIS data analysis.

2.2.1. Offline compression methods

Offline compression methods operate on complete historical trajectories and exploit global information to reduce redundancy. Among these, Uniform Sampling (US) (Braverman et al., 2022) and the DP-based algorithms are the most widely used. The US algorithm compresses trajectories by retaining points at fixed spatial or temporal intervals. Although computationally efficient, this approach does not account for trajectory geometry or vessel maneuvering behavior and therefore often fails to preserve critical structural features.

In contrast, the DP algorithm simplifies trajectories by iteratively removing points with small deviations from a baseline connecting trajectory endpoints, thereby retaining key geometric features with a reduced number of points (Liu et al., 2019). Owing to its ability to preserve overall trajectory shape, DP has been extensively adopted in trajectory analysis. However, its performance is highly dependent on manually defined compression thresholds, which limits its adaptability across trajectories with diverse navigation characteristics.

To overcome this limitation, numerous DP-based extensions have been proposed that incorporate additional information beyond spatial deviation. Some studies integrate temporal constraints or kinematic attributes, such as SOG and COG, to improve compression accuracy (Meratnia and de By, 2004; Zhao and Shi, 2018). Zhou et al. (2023) proposed the Multi-objective Peak Douglas-Peucker (MPDP) algorithm. The method integrates spatial information with kinematic features, including course over ground (COG) and speed over ground (SOG), to enhance the reliability of trajectory compression. However, despite these improvements, the continued reliance on a uniform compression threshold across heterogeneous trajectories limits the adaptability of such methods and constrains their achievable accuracy. These enhancements enable better preservation of navigation behavior and can reduce compression-induced distortion. Nevertheless, most of these approaches still rely on fixed or uniformly applied thresholds, which restrict their ability to adapt to heterogeneous vessel trajectories.

Recognizing this issue, recent research has shifted towards ADP algorithms that dynamically determine compression thresholds based on trajectory-specific characteristics. Adaptive approaches typically combine geometric deviation with measures of kinematic change, such as speed variation or heading fluctuation, allowing the compression process to better reflect vessel maneuvering behavior. These methods reduce subjectivity in threshold selection and improve robustness across diverse trajectories.

Despite these advances, existing ADP-based methods often consider geometric, temporal, and kinematic features in a fragmented manner. The joint influence of spatial position, time, speed, and heading is rarely integrated in a unified framework, which may result in the loss of important navigation signatures during compression. This limitation constrains the reliability of feature point extraction and, in turn, affects downstream trajectory similarity measurement.

To address this gap, this study proposes a SCAADP algorithm that holistically integrates spatial, temporal, and kinematic information when determining compression thresholds. By jointly considering latitude, longitude, timestamp, SOG, and COG, the proposed method more accurately identifies feature points while maintaining a high compression ratio with minimal trajectory distortion. This capability provides a robust foundation for the subsequent DCDTW similarity measurement framework.

2.2.2. Online compression methods

Online trajectory compression methods operate on streaming data and rely on locally available information to reduce trajectory redundancy in real time. Unlike offline approaches, these methods do not require access to complete historical trajectories and are therefore well-suited to applications involving real-time vessel tracking and data transmission. Existing online compression techniques can be broadly grouped into four categories: the Opening Window (OPW) algorithm (Meng et al., 2017), the Dead Reckoning (DR) algorithm (Chen et al., 2010), the Spatial Quality Simplification Heuristic (SQUISH) algorithm (Muckell et al., 2011a), and the Threshold-Guided Sampling (TGS) algorithm (Potamias et al., 2006).

Opening-window-based methods identify feature points by continuously evaluating local geometric and kinematic variations within a sliding window. Representative implementations assess changes in Euclidean distance, COG, and SOG between consecutive trajectory points to detect significant manoeuvres. For example, Zhu and Ma (2021) incorporated COG deviation and SOG change rates to improve sensitivity to vessel manoeuvring behavior. These approaches are effective in capturing local trajectory dynamics but are inherently limited to short-term context.

DR-based methods predict future vessel positions by extrapolating from current state variables, such as position, speed, and heading, and retain new points only when the prediction error exceeds a predefined threshold (Klein and Diamant, 2020). This strategy achieves efficient data reduction and has been widely adopted in navigation systems. However, its performance is highly dependent on prediction accuracy and threshold selection, particularly under abrupt manoeuvres.

Buffer-based heuristic methods, such as the Spatial Quality Simplification Heuristic (SQUISH), maintain a fixed-size buffer of recent trajectory points and remove redundant points once capacity is reached. SQUISH has demonstrated strong computational efficiency and scalability for large AIS datasets and has been further extended to enforce predefined compression ratios (Muckell et al., 2011b). This approach has been widely adopted for its efficiency in processing large-scale AIS datasets. The TGS, originally proposed by Potamias et al. (2006), defines a safety region based on a vessel's current position and SOG, retaining a

trajectory point only when the predicted next state falls outside this region. With the rapid growth of global AIS data, such online compression methods have become essential for real-time maritime traffic processing. Building on this line of research, Muckell et al. (2014) extended the SQUISH algorithm to SQUISH-Extended, enabling predefined compression ratios while simultaneously reducing compression error and improving spatiotemporal efficiency. To further enhance computational performance, Han et al. (2018) developed a parallelized implementation of SQUISH-Extended, leveraging multi-core CPU architectures to significantly accelerate trajectory compression.

Online compression algorithms are better suited than offline approaches for applications that require simultaneous trajectory compression and transmission. However, their effectiveness is inherently constrained by two factors: reliance on incrementally available data points and the need to operate under strict computational budgets. These constraints introduce notable challenges. Early-stage simplification may discard critical features that are essential for accurately representing subsequent trajectory segments, while locally induced errors can propagate and accumulate over time, degrading the overall quality of the compressed trajectory.

2.3. Review of studies on GPU-accelerated computing

The DCDTW algorithm reduces computational complexity by transforming the measurement of non-feature points from two-dimensional Euclidean distance calculations to one-dimensional directed distance sequence comparisons. This transformation substantially decreases the number of costly square-root operations. Nevertheless, when applied to long trajectories and large-scale datasets, the algorithm remains computationally intensive. Although conventional CPU-based serial frameworks can satisfy the functional requirements of trajectory similarity measurement, they incur substantial execution time because trajectory points must be processed sequentially. While multi-process strategies allow multiple trajectories to be handled in parallel, scalability remains limited for datasets comprising hundreds or thousands of trajectories. Excessive process creation can significantly degrade performance, whereas insufficient parallelism fails to meet time constraints. Moreover, simply increasing the number of CPU cores leads to prohibitive computational costs without guaranteeing proportional efficiency gains. Recent studies have also demonstrated the potential of GPU-based parallel computing to accelerate computationally intensive maritime trajectory-processing tasks, including large-scale vessel trajectory

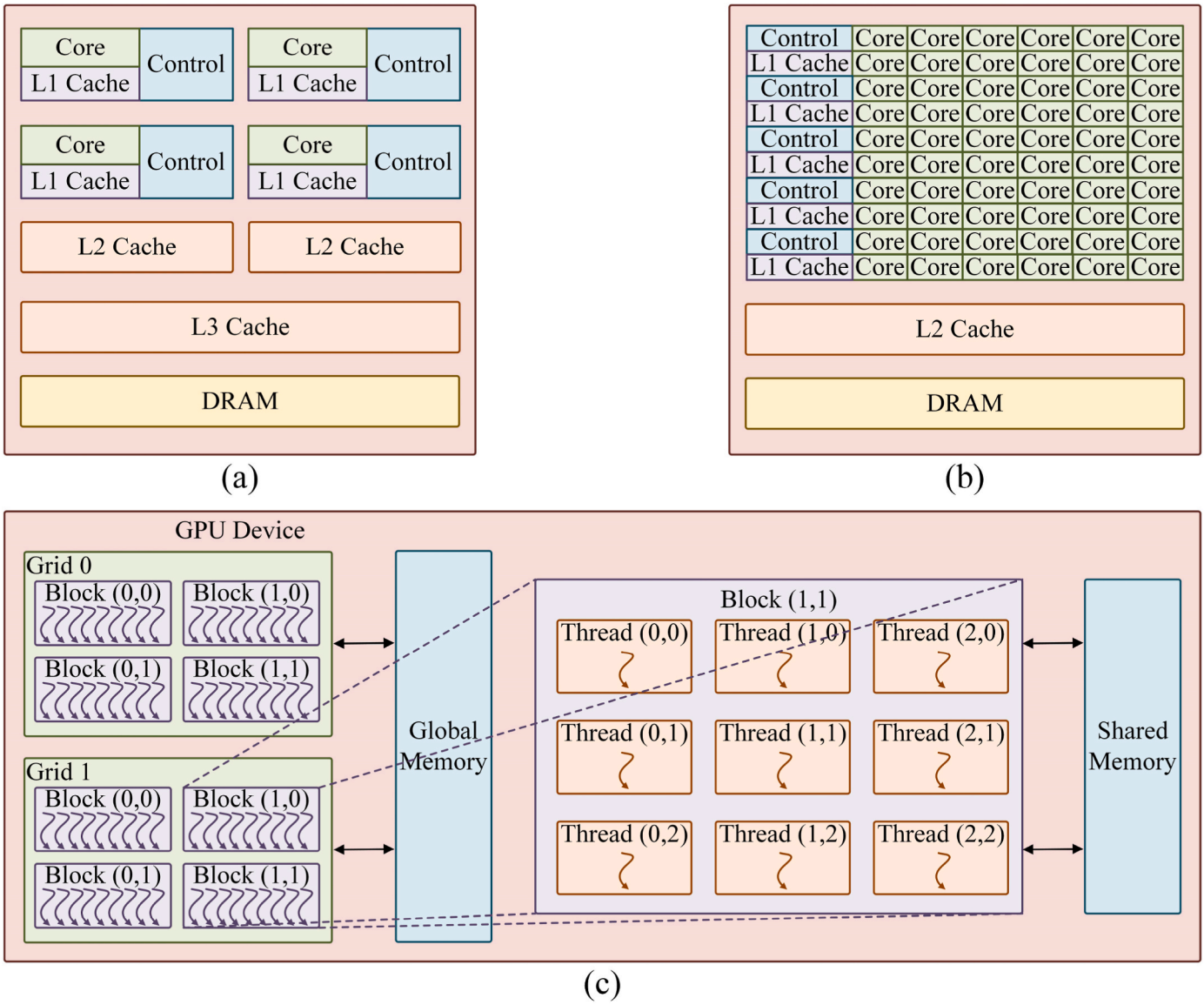

Fig. 1. Architectural comparison of CPU versus GPU computing frameworks: (a) CPU, (b) GPU, and (c) the CUDA parallel programming paradigm.

compression and AIS trajectory analysis (Huang et al., 2020; Li et al., 2024). These studies highlight the advantages of GPU architectures for handling the substantial computational requirements associated with increasingly large AIS datasets. Building on these developments, the present study integrates GPU-based parallelization with the proposed trajectory-processing methodology, covering both adaptive feature-point extraction and dual-component trajectory similarity computation. This provides the computational foundation required for applying the proposed approach to large-scale AIS trajectory datasets.

To overcome these limitations, this study further optimizes the algorithm within a GPU-based parallel computing framework. By mapping trajectory points to GPU threads, computations can be executed concurrently, substantially reducing processing time. Compared with CPUs, GPUs offer superior parallel throughput due to their architectural design (Chowdhury et al., 2022). CPUs typically comprise a limited number of cores, each supporting only a small number of threads, whereas GPUs are capable of executing thousands of threads simultaneously, enabling highly efficient data-parallel processing. The fundamental architectural differences between CPUs and GPUs are illustrated in Fig. 1(a) and (b).

Originally developed as coprocessors for real-time graphics and image rendering, GPUs have progressively evolved into general-purpose accelerators widely adopted in scientific computing (Shi et al., 2026), artificial intelligence (Zhang et al., 2023), and large-scale data analysis (Zhou et al., 2021). In modern GPU architectures, the thread constitutes the basic computational unit. Thousands of threads are organized into blocks, and multiple blocks collectively form a grid, which represents the highest level of execution hierarchy (Gilman et al., 2021). A schematic of this architecture is provided in Fig. 1(c). For trajectory-related computations, individual trajectory points can be mapped to distinct threads, or entire trajectories can be assigned to single threads, depending on the task granularity. Threads within the same block can efficiently exchange data through shared memory, whereas inter-block communication relies on global memory, which exhibits substantially higher latency (Rosen, 2013). Consequently, careful thread and memory allocation is essential; poorly designed mappings can negate the performance advantages of GPU parallelism and significantly increase runtime.

From a historical perspective, GPUs initially supported only fixed-function operations such as rasterization and texture mapping. The term “GPU” was formally introduced in 1999, but early architectures were ill-suited for general-purpose computation (Dally et al., 2021). Subsequent efforts in General-Purpose GPU (GPGPU) computing enabled limited programmability through graphics shaders, albeit with considerable implementation complexity (Maceina and Manduchi, 2017). A major turning point occurred with the introduction of unified GPU programming models, which allowed developers to write general-purpose code that executes directly on GPUs, leading to widespread adoption in large-scale data processing and scientific research (Garland et al., 2008). Later architectural advancements established hierarchical execution and memory models that further enhanced programmability and efficiency (Wittenbrink et al., 2011). In recent years, rapid advances in artificial intelligence, particularly deep learning and large language models, have further elevated the role of GPUs. Specialized GPU hardware can significantly accelerate large-scale matrix computations and data-intensive workloads (Fu et al., 2025).

2.4. Research gaps

Despite the widespread adoption of DTW and its variants for trajectory similarity measurement, several fundamental limitations restrict their effectiveness in real-world transportation applications.

(1) Undifferentiated trajectory point processing.

Despite the widespread use of DTW and its variants for trajectory comparison, a key issue is the heterogeneous information represented by different trajectory points. Conventional DTW operates on the observed point sequences through a common point-wise alignment mechanism and does not explicitly distinguish points associated with major movement changes from those describing relatively stable intermediate segments. For AIS trajectories with non-uniform sampling, this formulation can make the accumulated distance strongly influenced by the number and distribution of observations along different trajectory segments. A trajectory representation that explicitly separates major structural changes from intermediate movement may therefore provide additional information for vessel trajectory comparison. This consideration motivates the dual-component formulation adopted in DCDTW.

(2) Limited adaptivity in feature-point extraction.

The performance of feature-based similarity methods is highly dependent on the effectiveness of trajectory compression. However, most existing approaches, including the classical DP algorithm and its adaptive variants, rely on manually specified compression thresholds. Such threshold selection typically requires expert intervention and extensive trial-and-error, while failing to account for trajectory-specific kinematic characteristics, such as variations in SOG and COG. The lack of a robust, self-adaptive compression mechanism that can automatically determine optimal thresholds by jointly considering spatial and dynamic attributes represents a significant gap. It can also directly affect the reliability and generalizability of downstream similarity analysis.

(3) Excessive computational overhead from dense distance operations.

A major bottleneck in large-scale trajectory similarity analysis arises from the high computational cost of traditional DTW, driven primarily by repeated two-dimensional Euclidean distance calculations. These dense operations involve frequent multiplications and computationally expensive square-root extractions, resulting in substantial processor overhead. Existing methods offer limited capability to strategically reduce the frequency of such operations. This gap highlights the need for a computational framework that confines resource-intensive calculations to a minimal subset of informative points while employing lightweight alternatives elsewhere, thereby reducing overall computational burden without compromising similarity fidelity.

(4) Insufficient scalability for large-scale trajectory datasets.

Trajectory similarity measurement is computationally intensive, and many existing DTW-based approaches rely on CPU-based or lightly parallelized implementations. As trajectory volumes increase across transport modes such as maritime, road, or air traffic, these implementations often become impractical for large-scale analysis, with execution times that limit their use in operational or near-real-time decision support. This scalability challenge is therefore not specific to shipping but reflects a more general limitation of DTW-based trajectory similarity measurement methods when applied to high-volume trajectory data. Addressing this gap requires both a methodological design that preserves similarity accuracy and system-level optimization capable of exploiting modern parallel computing architectures to support large-scale transport analytics.

3. Preliminary

This section first introduces the definitions required for implementing the proposed DCDTW-based trajectory similarity measure (Section 3.1). Section 3.2 then describes the two real-world datasets used in the comparative experiments. Finally, Section 3.3 outlines the procedures for detecting and correcting noise in the original trajectory data to improve data quality and measurement accuracy.

3.1. Definitions

Definition 1. Raw trajectory. A vessel trajectory, denoted as *Rtra*, is defined as a sequence of AIS points with a length of R. It can be formally represented as:

$$Rtra = [RP_1, RP_2, \cdots, RP_m, RP_{m+1}, \cdots, RP_R], \tag{1}$$

$$RP_m = [tsR_m, lonR_m, latR_m, sR_m, cR_m], m = 1, 2\cdots, R. \tag{2}$$

where the *m*-th point RP_m in the raw trajectory Rtra is defined by a tuple of five dynamic attributes (tsR_m, $lonR_m$, $latR_m$, sR_m, cR_m), representing timestamp, longitude, latitude, SOG, and COG, respectively.

Definition 2. Compressed trajectory. A compressed trajectory, denoted by *Ctra* with length *C*, is derived from the raw trajectory Rtra. Its formulation representation is given by Eqs. (3) and (4).

$$Ctra = \{[CP_1, CP_2, \cdots, CP_n, CP_{n+1}, \cdots, CP_C], posL\}, \tag{3}$$

$$CP_n = [tsC_n, lonC_n, latC_n, sC_n, cC_n], n = 1, 2\cdots, C. \tag{4}$$

where CP_n denotes the *n*-th point in the compressed trajectory and is represented by a five-dimensional tuple of dynamic attributes, as defined in Eq. (4).

The vector posL is a one-hot indicator whose length equals that of the raw trajectory. Each element corresponds to a specific point in the original trajectory: a value of 1 indicates that the point is retained as a feature point by the proposed SCAADP compression algorithm, while a value of 0 denotes a discarded non-feature point. This positional indicator enables the DCDTW algorithm to explicitly distinguish between feature and non-feature points during trajectory similarity computation.

Definition 3. AIS matrix. The primary dynamic AIS data of a vessel is structured into a 5 × T matrix, formulated as follows:

$$AM_{5\times T} = \begin{bmatrix} tsR_1 & tsR_2 & \cdots & tsR_m & \cdots & tsR_T \\ lonR_1 & lonR_2 & \cdots & lonR_m & \cdots & lonR_T \\ latR_1 & latR_2 & \cdots & latR_m & \cdots & latR_T \\ sR_1 & sR_2 & \cdots & sR_m & \cdots & sR_T \\ cR_1 & cR_2 & \cdots & cR_m & \cdots & cR_T \end{bmatrix} \tag{5}$$

where T indicates the total number of trajectory points. The five variables follow the definitions given in Eq. (2). In this study, $AM_{5\times T}$ is directly used as the input to the GPU-accelerated optimization framework, thereby reducing the overhead associated with repeated data input/output operations.

Definition 4. Similarity measurement matrix. The similarity measurement matrix is defined as an N × N matrix that records the pairwise similarity values between trajectories in the dataset:

$$SM_{N\times N} = \begin{bmatrix} sm_{1,1} & sm_{1,2} & \cdots & sm_{1,N} \\ sm_{2,1} & sm_{2,2} & \cdots & sm_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ sm_{N,1} & sm_{N,2} & \cdots & sm_{N,N} \end{bmatrix} \tag{6}$$

where $sm_{i,j}$ denotes the similarity measure value between the *i*-th and *j*-th trajectories. All diagonal elements satisfy $sm_{i,i} = 0$, reflecting the fact that the similarity measure between a trajectory and itself is zero. Moreover, the matrix is symmetric, such that $sm_{i,j} = sm_{j,i}$, indicating that the similarity measure is invariant to the ordering of trajectory pairs.

3.2. Experimental dataset description

To evaluate the effectiveness of the proposed SCAADP and DCDTW algorithms, this study utilizes AIS data comprising timestamps, latitude–longitude coordinates, SOG, and COG from two representative maritime regions: Caofeidian Port (CFDP) and Chengshan Jiao Promontory (CSJP). The two regions represent contrasting geographical and navigational settings. CFDP is a port-oriented navigation area, where vessel movements are mainly associated with port approaches, arrivals and departures, and relatively constrained navigation channels. In contrast, CSJP is a coastal transit area around a prominent headland, characterized by multiple navigation corridors and more spatially distributed through-traffic patterns. As summarized in Table 1, CFDP contains 630 trajectories and 229,555 AIS points, whereas CSJP contains 1520 trajectories and 854,063 AIS points. The average number of AIS points per trajectory is 364.373 and 561.883, respectively.

These statistics indicate substantial differences in dataset scale and trajectory composition between the two study areas. However, AIS point counts are not interpreted as direct measures of vessel traffic density because they may also be influenced by trajectory duration, vessel speed, and AIS reporting intervals. The present study does not explicitly incorporate meteorological or oceanographic variables such as wind, waves, currents, or visibility; therefore, differences in physical environmental conditions between the two regions are not quantitatively evaluated. Instead, the comparison focuses on geographical and navigational characteristics that can be directly identified from the study areas and AIS trajectory distributions. The spatial distributions of vessel trajectories are shown in Fig. 2(a) and (b), and the main dataset statistics are summarized in Table 1.

The two study areas were selected to represent different maritime traffic environments. CFDP is a port area characterised by dense port-related vessel movements and frequent manoeuvring, whereas CSJP is a coastal navigation area with more diverse route patterns and vessel interactions. These differences allow the proposed methods to be evaluated across varying traffic densities, trajectory patterns, and navigational conditions.

Given that raw AIS transmissions are often affected by noise and reporting errors, the data are preprocessed prior to analysis. Following the procedures described in Section 3.3, noise detection and correction are applied to improve trajectory reliability, thereby ensuring the validity of the subsequent comparative experiments.

3.3. Raw trajectory denoising

AIS data transmitted via terrestrial base stations or satellites is often contaminated by noise arising from signal interference, transmission errors, or sensor malfunctions, which can introduce spurious artifacts into vessel trajectories. Such anomalies may significantly distort trajectory geometry and compromise the validity of similarity analysis. This section therefore presents the methods adopted for noise identification and restoration in raw AIS trajectories.

Noise detection in trajectory data commonly relies on Euclidean distance measures, making density-based clustering methods particularly suitable. Among these, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is widely used, as it classifies points into core, border, and noise categories based on neighborhood density defined by a radius parameter and a minimum point threshold (Fuentes, 2021; Kumar et al., 2025). However, conventional DBSCAN exhibits three notable limitations: (1) its performance is sensitive to manually selected hyperparameters; (2) it is effective primarily at detecting high-amplitude outliers while failing to identify low-amplitude noise points (Fig. 3a and b); and (3) it frequently misclassifies legitimate trajectory turning points as noise (Fig. 3(c)).

To overcome these limitations, this study adopts a density matrix-based adaptive DBSCAN approach developed in our previous work, the workflow of which is illustrated in Fig. 4. The core idea is to map trajectory points onto a spatial grid to construct a density matrix, followed by convolution-based analysis to evaluate direct density reachability. The procedure consists of three main steps.

Step 1: Density matrix construction. Based on the latitude-longitude boundaries of the trajectory and a predefined grid resolution, each

Table 1
Summary statistics of the two study areas (i.e., CFDP and CSJP).

Research Waters	Number of Vessel Trajectories	Number of Timestamped Points	Maximum Number of Points Per Trajectory	Minimum Number of Points Per Trajectory	Average Number of Points Per Trajectory	Standard Deviation of Points Per Trajectory	Boundary Points	Longitude (°)	Latitude (°)
CFDP	630	229555	705	51	364.373	109.167	Left Top Right Bottom	118.500 118.250	38.999 38.760
CSJP	1520	854063	2699	50	561.883	210.338	Left Top Right Bottom	122.583 123.167	37.750 37.167

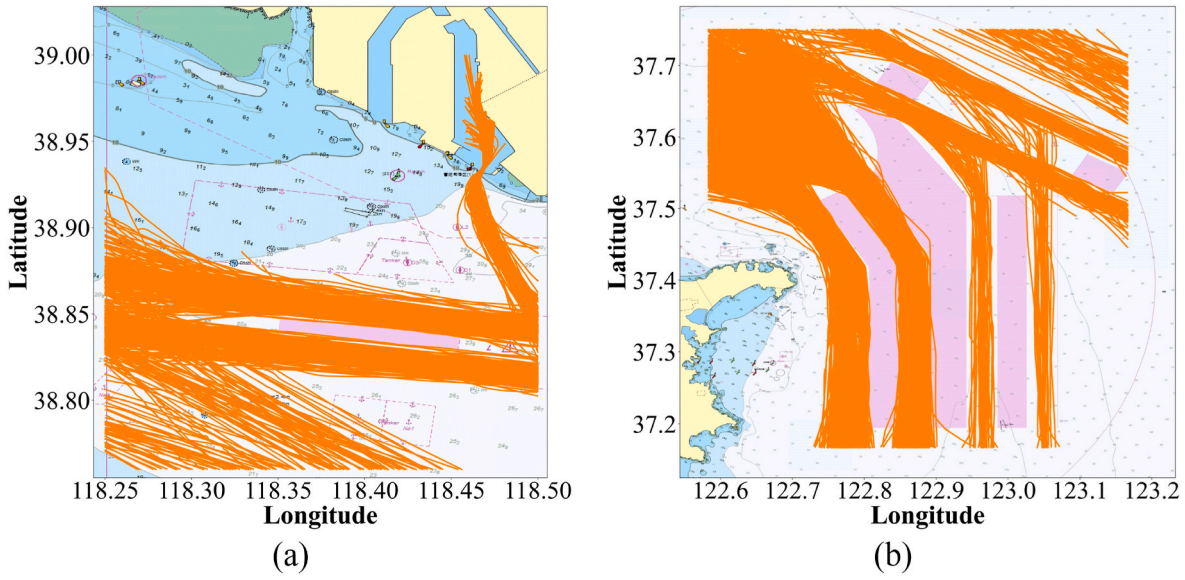

Fig. 2. Vessel trajectory datasets from two distinct maritime areas: (a) CFDP and (b) CSJP.

trajectory point is mapped to a grid cell, yielding a density matrix that represents local point concentrations (Fig. 4(a)).
Step 2: Convolution-based density enhancement. A convolution kernel of dimension $w \times w$ is applied to the density matrix to aggregate neighborhood information (Fig. 4(c)). Prior to convolution, boundary effects are handled as illustrated in Fig. 4(b). The kernel size w controls the effective neighborhood range: excessively large values may obscure local density variations and lead to missed noise detection. Accordingly, a small kernel size is preferred; in this study, $w = 3$. The resulting convolved density matrix is shown in Fig. 4(d).
Step 3: Noise identification via density reachability. Grid cells corresponding to trajectory points are examined sequentially in temporal order (Fig. 4(e)). For each point, direct density reachability between its neighborhood matrix and those of adjacent points is evaluated. If reachability exists, the point is classified as normal; otherwise, it is identified as noise. As illustrated in Fig. 4(f), the absence of overlap between the neighborhood matrix of the current point (yellow frame) and those of adjacent points (blue and brown frames) indicates a noise point, whereas intersecting neighborhoods correspond to normal trajectory points.

Following noise identification, detected noise points are repaired using a dead reckoning-based restoration method grounded in nautical navigation principles. Given the typically short reporting intervals of AIS data (ranging from seconds to tens of seconds), this approach provides reliable position estimates. The restoration process comprises three steps.

Step 1. COG conversion. The COG of the preceding normal point is converted from degrees to radians to enable trigonometric calculations, as defined in Eq. (7):

$$cRad = c \times \frac{\pi}{180} \tag{7}$$

where c refers to the COG of the normal point at the previous timestamp relative to the noise point.

Step 2. Displacement estimation. The northward (latitudinal) and eastward (longitudinal) displacement components are computed using Eqs. (8) and (9), respectively.

$$\Delta N = s \times \frac{t}{3600} \times \cos(cRad), \tag{8}$$

$$\Delta E = s \times \frac{t}{3600} \times \sin(cRad). \tag{9}$$

where s denotes the SOG of the normal point at the previous timestamp relative to the noise point. t represents the time interval between the noise point and the preceding normal point, measured in seconds. These displacements are expressed in nautical miles.

Step 3. Coordinate reconstruction. The latitude is calculated using Eq. (10), while the longitude is derived through Eqs. (11)–(13).

$$lat_{re} = lat_{nor} + \frac{\Delta N}{60}. \tag{10}$$

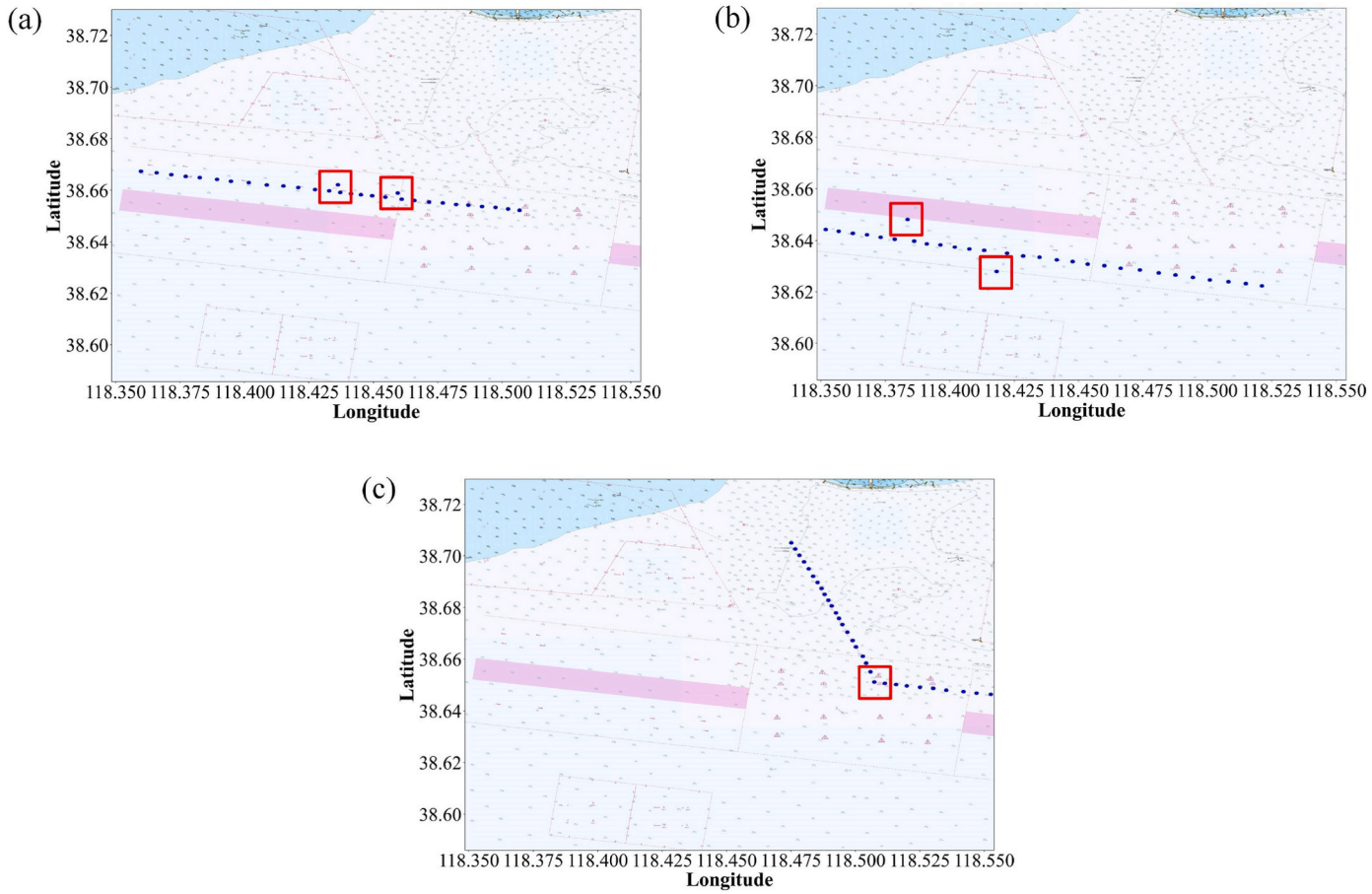

Fig. 3. Schematic illustration of vessel trajectory data affected by noise: (a) data with minor noise perturbations, (b) data with significant noise disturbances, and (c) a sudden turning point that is potentially misidentified as noise.

$$mid = \frac{lat_{re} + lat_{nor}}{2}, \tag{11}$$

$$\Delta lon = \frac{\Delta E}{60 \times \cos\left(mid \times \frac{\pi}{180}\right)}, \tag{12}$$

$$lon_{re} = lon_{nor} + \Delta lon. \tag{13}$$

where (lon_{nor}, lat_{nor}) denotes the position (latitude and longitude) of the normal point at the timestamp immediately preceding the noise point. (lon_{re}, lat_{re}) is the corrected value after noise point restoration. The conversion factor of 1/60 is applied to account for the relationship between nautical miles and degrees. To ensure accuracy in longitude estimation, the mean latitude is incorporated into the calculation.

The overall workflow integrating noise identification and restoration is summarized in Fig. 5. To assess the effectiveness of the proposed denoising strategy, representative noisy trajectories from each study area were selected for evaluation. The original trajectories plotted on nautical charts are shown in Fig. 6, while Fig. 7 presents the corresponding denoising results. These results demonstrate that the proposed approach effectively suppresses noise while preserving key trajectory features, thereby providing high-quality input data for subsequent trajectory similarity analysis.

4. Methodology

This paper proposes a novel vessel trajectory similarity measurement method based on a dual-component computational paradigm that differentiates trajectory points according to their contribution to trajectory shape. Compared with conventional DTW-based methods, the proposed approach separately treats feature and non-feature points, resulting in more accurate and efficient trajectory similarity measurement. Section 4.1 presents the overall technical framework of the proposed method. Section 4.2 details the principles of the DCDTW algorithm. Section 4.3 describes the GPU-based parallel implementation designed to substantially reduce computational runtime.

4.1. A GPU-accelerated dual-component measurement framework

The overall framework of the proposed approach is illustrated in Fig. 8. It comprises three main stages: AIS data acquisition, noise identification and restoration of raw trajectories, and GPU-accelerated trajectory similarity measurement using the proposed DCDTW algorithm.

The framework is designed to address two fundamental limitations of traditional DTW-based methods: limited measurement accuracy and poor computational scalability. In terms of accuracy, conventional approaches perform uniform point-to-point matching, which often obscures key morphological features due to the dominance of non-feature points. The proposed method overcomes this limitation by classifying trajectory points into feature and non-feature points and applying differentiated similarity measures, thereby preserving salient trajectory characteristics.

From a computational perspective, traditional DTW requires exhaustive two-dimensional Euclidean distance calculations between all point pairs, resulting in numerous costly square-root operations. In contrast, the proposed method restricts such calculations to feature points only. For non-feature points, a dimensionality reduction strategy is employed that transforms two-dimensional Euclidean distance

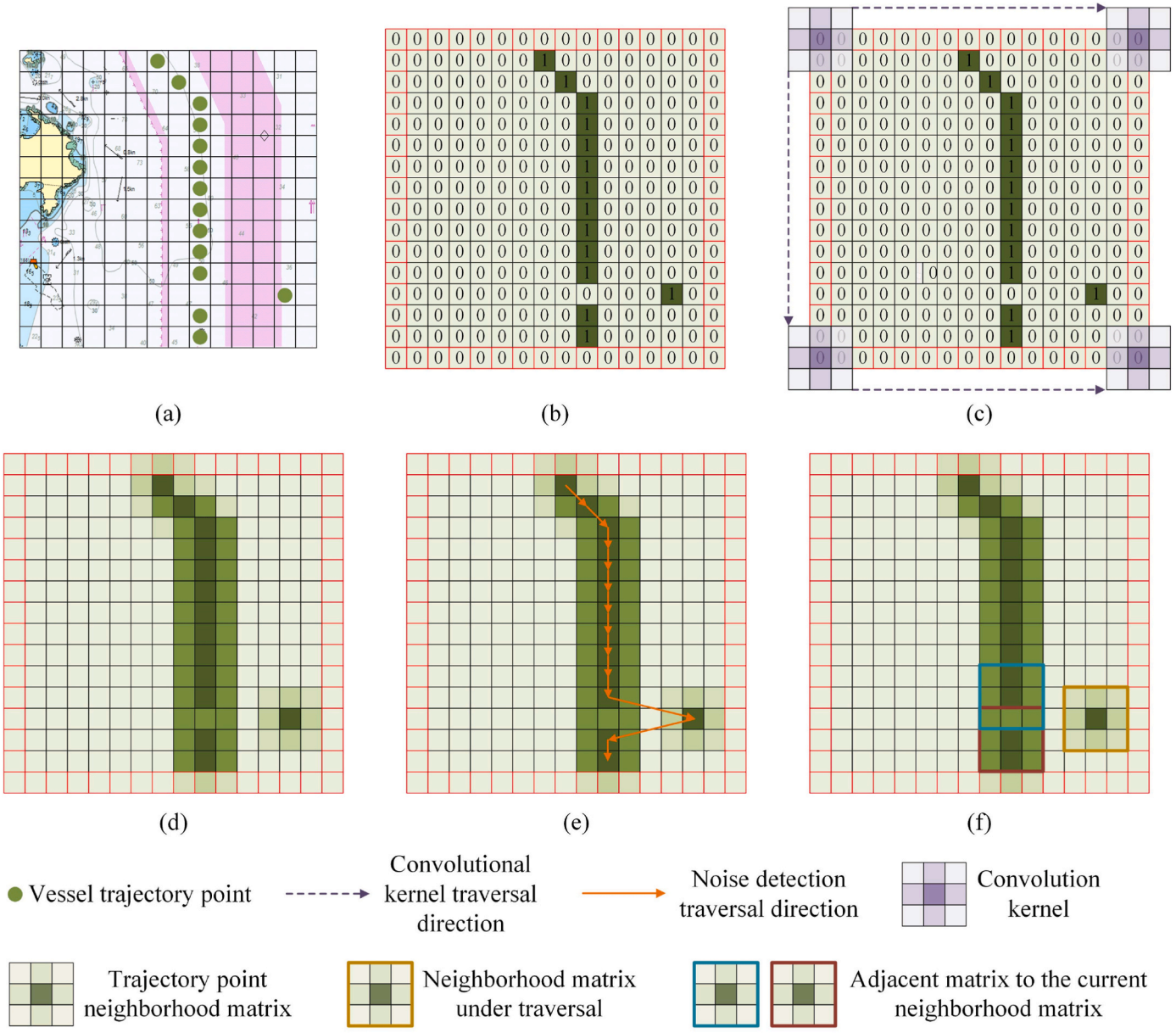

Fig. 4. Flowchart of the density matrix-based adaptive DBSCAN identification algorithm. The process includes: (a) projecting trajectory points to the initial density matrix, (b) populating with zeros to mitigate boundary effects, (c) performing convolution on the density matrix, (d) the resulting matrix after convolution, (e) traversal of all neighborhood matrices, and (f) identification of directly density-reachable noise points.

computations into one-dimensional directed distance sequence comparisons. This design significantly reduces computational overhead. To further enhance performance for large-scale trajectory datasets, the DCDTW algorithm is implemented within a GPU parallel computing framework, enabling efficient concurrent processing and substantial runtime reduction.

The potential benefit of the dual-component strategy depends on the characteristics of the trajectories being compared. It is expected to be particularly relevant when trajectories exhibit non-uniform sampling densities, unequal numbers of observations, pronounced turning or manoeuvring behaviour, or similar overall routes accompanied by different local movement characteristics. In such cases, separating feature and non-feature components provides an explicit means of representing both major structural changes and the local evolution between them.

Conversely, when trajectories have simple geometric structures, contain few distinctive manoeuvres, and are sampled at similar and relatively uniform densities, feature and non-feature components may convey less differentiated information, and the benefit of the dual-component representation relative to conventional DTW may consequently be smaller. DCDTW should therefore not be interpreted as assuming that feature points dominate trajectory similarity under all conditions. Rather, its purpose is to account explicitly for the heterogeneous structural roles of trajectory observations when such heterogeneity is present.

4.2. *The proposed DCDTW-based vessel trajectory measurement method*

The section details the implementation of the proposed DCDTW method. Since accurate identification of feature points is a prerequisite for differentiated similarity measurement, Section 4.2.1 first introduces the proposed SCAADP algorithm for trajectory compression and key-point extraction. Section 4.2.2 then presents the DCDTW similarity computation based on the extracted feature and non-feature points.

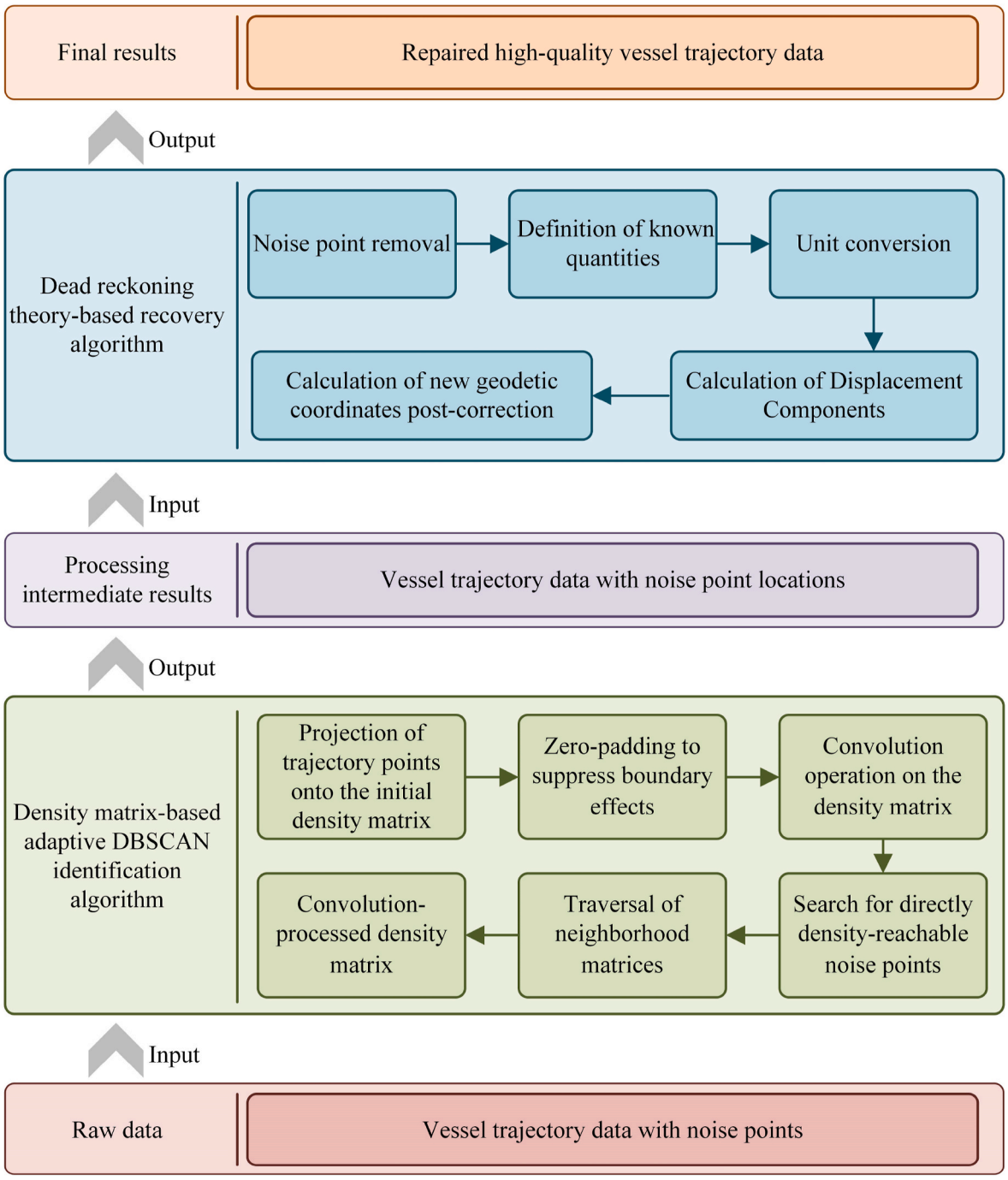

Fig. 5. A synergistic framework for noise identification and data restoration.

4.2.1. Vessel trajectory key points extraction via SCAADP

A core challenge of DCDTW is the reliable separation of feature and non-feature points. This is addressed through trajectory compression, where retained points are regarded as feature points that capture the essential geometric and kinematic characteristics of vessel motion.

The DP algorithm is a classical trajectory compression method that recursively identifies feature points based on a distance threshold, as illustrated in Fig. 9. Although effective, DP relies on a manually specified global threshold, which limits its adaptability across trajectories with heterogeneous motion patterns. To overcome this limitation, adaptive DP (ADP) variants have been proposed, where trajectory-specific thresholds are automatically derived from geometric properties (Liu et al., 2019). While effective for most linear and curved trajectories, this method fails when curved trajectories contain near-linear sub-segments. As shown in Fig. 10(b), the intermediate point P8 in the P7-P9 segment is redundant, and including its distance inflates the compression threshold.

To address the limitations of ADP, weighted thresholding approaches have been proposed in which trajectory points contribute unequally to compression threshold estimation. ADPS (Li et al., 2022) and ADP with Speed and Course (ADPSC) (Li et al., 2024) implement this idea by weighting point-to-baseline distances using normalized rates of change in SOG. ADPSC further improves physical interpretability by decomposing SOG into components parallel and perpendicular to the trajectory baseline and using only the perpendicular component to characterise lateral deviation.

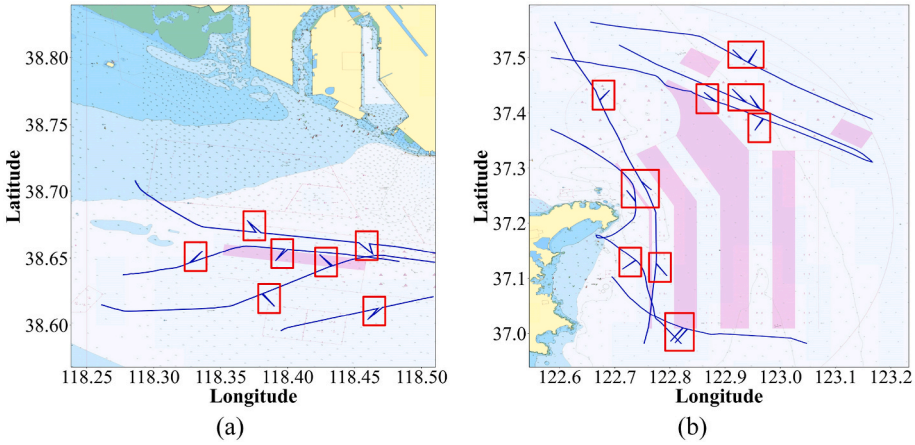

Fig. 6. The visualization depicts raw trajectory data with noise from two study areas, (a) CFDP and (b) CSJP, with red rectangles highlighting representative noisy segments.

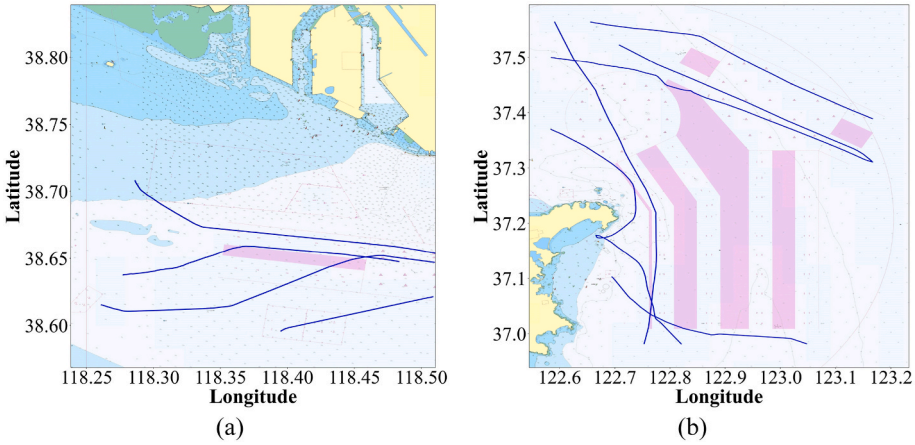

Fig. 7. Distribution of cleaned vessel trajectories across the two study areas: (a) CFDP and (b) CSJP.

Despite these improvements, ADPSC does not explicitly incorporate COG dynamics, using COG only to assist SOG decomposition. Consequently, points in straight-line segments with constant COG may still be assigned non-zero weights, even when they do not influence trajectory shape.

The proposed SCAADP algorithm addresses this limitation by jointly incorporating the rates of change of both SOG and COG into threshold weighting, as illustrated in Fig. 10(c). SOG is decomposed directly in the original coordinate system, and changes along both axes are combined to form a unified dynamic weight. Using the Pythagorean theorem, the contributions of the orthogonal SOG components are integrated to derive a comprehensive weight for each point. This design enables more accurate and robust feature-point extraction across heterogeneous vessel trajectories.

Based on the above analysis, the contribution of each trajectory point to the compression threshold is determined by the rates of change in SOG components and COG. These dynamic features are therefore integrated to compute an adaptive compression threshold, as illustrated in Fig. 11.

(1) *Contribution ratio of SOG variation.* As shown in Fig. 11(a), the SOG of each trajectory point is decomposed into x- and y-direction components using the corresponding COG, with appropriate angle conversion to account for quadrant-specific orientation. The detailed decomposition procedure is provided in Algorithm 1.

Algorithm 1. Computation of SOG Components (CoSC)

Input: s, c//s and c represent the SOG and COG of the trajectory points, respectively.
Output: sX, sY//sX and sY represent the components of the trajectory point's SOG along the x-axis and y-axis, respectively.

if $0 \le c \le 90$ **then**
  cNew = 90 - c;
  $sX = s \times \cos\left(cNew \times \frac{\pi}{180}\right)$;
  $sY = s \times \sin\left(cNew \times \frac{\pi}{180}\right)$;
else if $90 < c \le 180$ **then**
  cNew = c - 90;
  $sX = s \times \cos\left(cNew \times \frac{\pi}{180}\right)$;
  $sY = -s \times \sin\left(cNew \times \frac{\pi}{180}\right)$;
else if $180 < c \le 270$ **then**
  cNew = 270 - c;
  $sX = -s \times \cos\left(cNew \times \frac{\pi}{180}\right)$;
  $sY = -s \times \sin\left(cNew \times \frac{\pi}{180}\right)$;
else if $270 < c < 360$ **then**
  cNew = c - 270;
  $sX = -s \times \cos\left(cNew \times \frac{\pi}{180}\right)$;
  $sY = s \times \sin\left(cNew \times \frac{\pi}{180}\right)$;
end if

Based on Algorithm 1, the decomposed SOG components along the x-axis and y-axis for each trajectory point are obtained and stored in

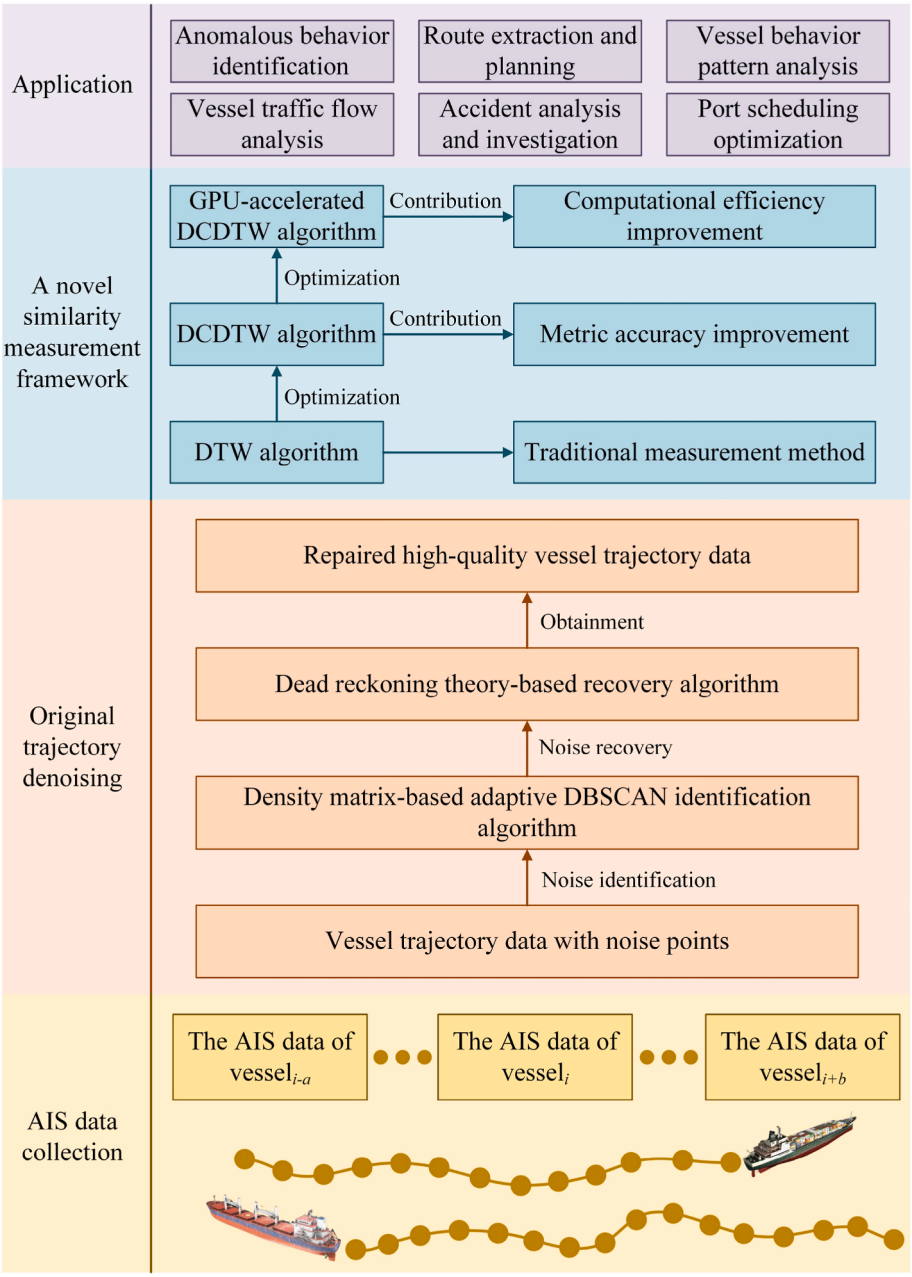

Fig. 8. Schematic diagram of the GPU-parallelized DCDTW algorithm.

vectors *sXVec* and *sYVec* of length *l*, where *l* denotes the total number of trajectory points. To support subsequent computations, AIS timestamp strings are converted into numerical values (in seconds) using Eq. (14).

$$ts = 86400 \times \mathrm{d}_u + 3600 \times ho_u + 60 \times mi_u + se_u \tag{14}$$

where d_u, ho_u, mi_u, and se_u denote the elapsed number of days from a fixed reference date, hour, minute, and second of the uu-th trajectory point, respectively. The resulting continuous timestamp ts_u is stored in the time vector *tsVec*. The time interval between two consecutive trajectory points is calculated as $\Delta t_u = ts_u\text{-}ts_{u\text{-}1}$. This formulation correctly handles trajectories crossing midnight or spanning multiple calendar dates. SOG is originally reported in knots (nautical miles per hour) and is therefore converted to metres per second using the standard conversion (1 nautical mile = 1852 m).

$$s_m = s_{nm} \times \frac{1852}{3600} \tag{15}$$

Subsequently, the rates of change of the SOG components along the *x*-axis and *y*-axis for each trajectory point are calculated separately. For the *u*-th trajectory point ($u = 2, \ldots, l$), the rate of change along the *x*-axis is given by

$$sXRate_u = \frac{|sX_u - sX_{u-1}|}{ts_u - ts_{u-1}}, u = 2, 3, 4, \cdots, l \tag{16}$$

where sX_u and ts_u represent the SOG component along the *x*-axis and the timestamp of the *u*-th trajectory point, respectively. An analogous formulation is used to compute the rate of change along the *y*-axis by substituting sY_u and sY_{u-1} into Eq. (16). The resulting rates are stored in vectors *sXRaVec* and *sYRaVec*, each of length *l*-1.

To avoid numerical instability caused by cancellation between positive and negative change rates, the contribution ratios are calculated

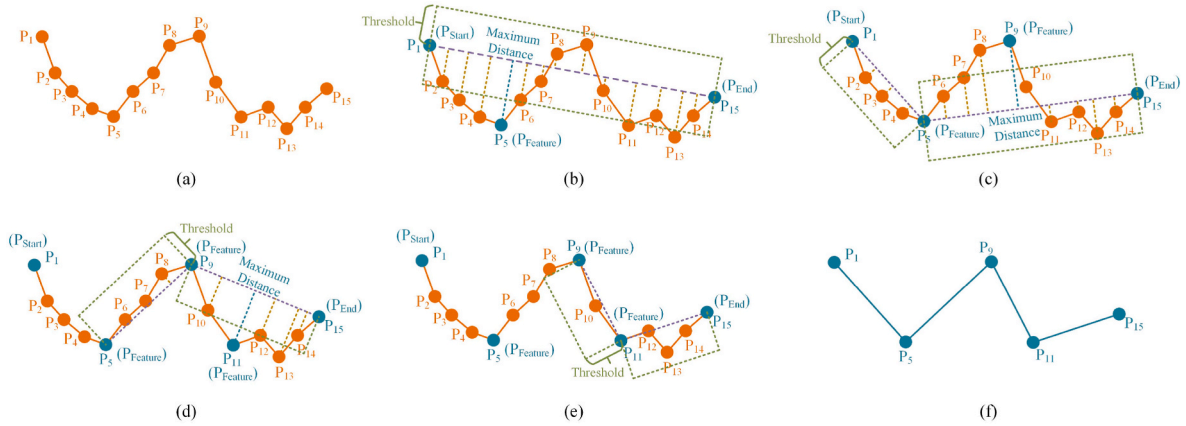

Fig. 9. Workflow of the original DP compression algorithm: (a) original trajectory, (b) selecting feature points by threshold, (c) segmenting trajectory and searching for points, (d, e) iteratively identifying new points, and (f) final compressed trajectory.

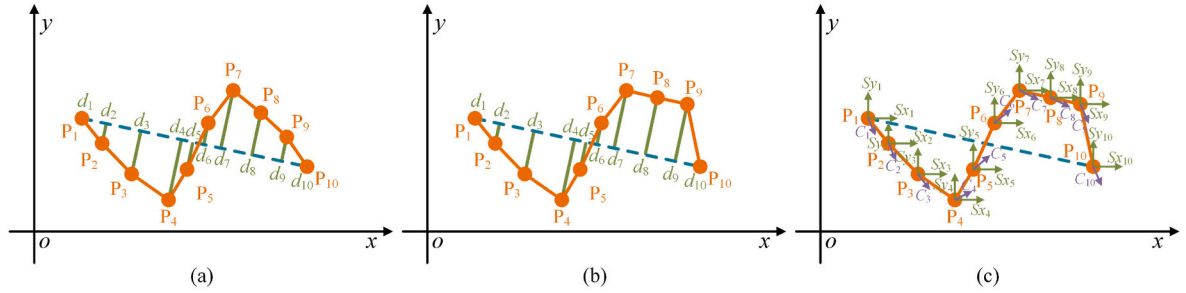

Fig. 10. Workflow for trajectory analysis: (a) and (b) distribution of midpoint-to-baseline distances for two trajectories, and (c) process of SOG vector decomposition.

using the absolute values of the corresponding rates of change. Specifically, the absolute rates in *sXRaVec* and *sYRaVec* are summed to obtain *sXRaSum* and *sYRaSum*, respectively:

$$sXRaSum = \sum_{u=2}^{l} |sXRate_u|, sYRaSum = \sum_{u=2}^{l} |sYRate_u| \tag{17}$$

A small numerical tolerance ε, is introduced to avoid division by zero or numerical instability when the corresponding sum approaches zero. The contribution ratios of the intermediate trajectory points along the x and y-axis directions are therefore calculated as:

$$\begin{cases} sYChaCon_u = \begin{cases} \dfrac{|sYRate_u|}{sYRaSum}, sYRaSum \geq \varepsilon, \\ 0, sYRaSum < \varepsilon, \end{cases} \\ sXChaCon_u = \begin{cases} \dfrac{|sXRate_u|}{sXRaSum}, sXRaSum \geq \varepsilon, \\ 0, sXRaSum < \varepsilon, \end{cases} \end{cases} \quad u = 2, 3, 4, \cdots, l-1 \tag{18}$$

Only intermediate trajectory points ($u = 2, \ldots, l$-1) participate in this calculation, since the start and end points lie on the baseline and thus contribute zero distance. The resulting contribution ratios are stored in vectors *sXChCoVec* and *sYChCoVec*, each of length *l*-2.

(2) *Contribution ratio of COG change.* The computation for the COG change contribution ratio follows the same framework as that for SOG (Fig. 11(b)), except that no vector decomposition is required. The COG values of all trajectory points are first stored in a vector *cVec* of length *l*.

Because COG is a circular variable, the change between two consecutive trajectory points is calculated using the minimum signed angular difference rather than direct subtraction. Specifically, the raw difference COG_u-COG_{u-1} is first calculated. If this difference is greater than 180°, 360° is subtracted; if it is less than −180°, 360° is added; otherwise, the original difference is retained. This treatment avoids artificial discontinuities at the 0°/360° boundary. For example, a change from 359° to 1° is correctly represented as 2° rather than 358°. The corrected angular difference is divided by the corresponding time interval to obtain the COG change rate for $u = 2, \ldots, l$, and the resulting values are stored in *cRaVec* of length l−1. Consistent with the SOG calculation, the absolute values of the COG change rates are used to avoid cancellation between positive and negative changes. The sum of the absolute COG change rates is calculated as:

$$cRaSum = \sum_{u=2}^{l} |sXRate_u| \tag{19}$$

$$cChaConu = \begin{cases} \dfrac{|sRate_u|}{sRaSum}, RaSum \geq \varepsilon, \\ 0, RaSum < \varepsilon, \end{cases} u = 2, 3, 4, \cdots, l-1 \tag{20}$$

The start and end points are excluded because their perpendicular distances to the baseline are zero. The resulting contribution ratios are stored in *cChCoV* of length *l-2*.

$$d_u = \frac{|(y_e - y_u) \times (x_s - x_u) - (y_s - y_u) \times (x_e - x_u)|}{\sqrt{(x_e - x_s)^2 + (y_e - y_s)^2}}, u = 2, 3, \cdots, l-1 \tag{21}$$

Consistent with the SOG calculation, the absolute values of the COG change rates are used to avoid cancellation between positive and negative changes. Their sum is denoted by *cRaSum*, and the contribution ratio *cChaConu* of each intermediate point ($u = 2, \ldots, l$−1) is obtained by normalizing its absolute COG change rate by *cRaSum*. When *cRaSum* approaches zero, the corresponding contribution ratios are set to zero.

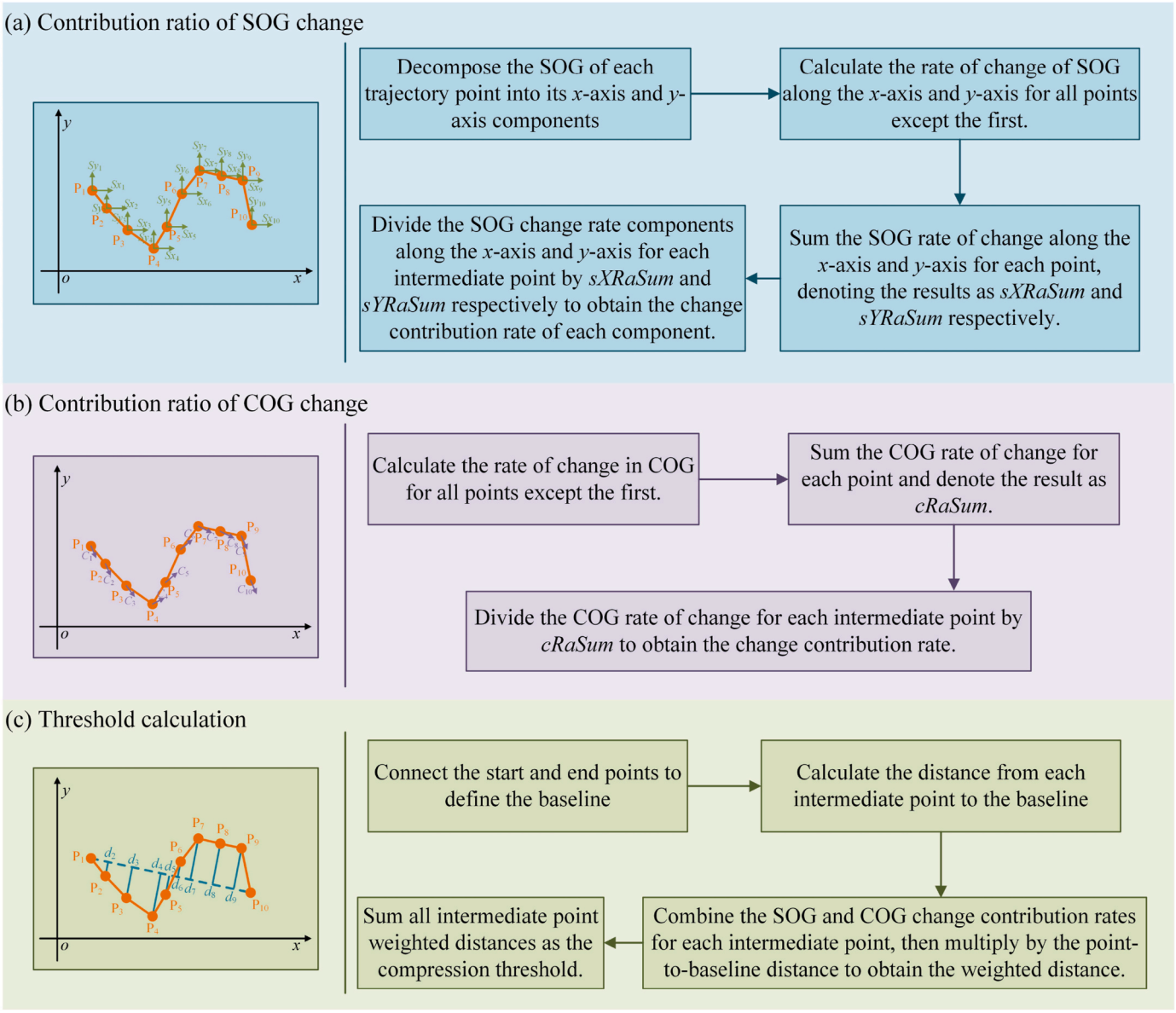

Fig. 11. Key components of threshold calculation: (a) contribution ratio of SOG change, (b) contribution ratio of COG change, and (c) threshold calculation.

The start and end points are excluded because their distances to the baseline are zero. The resulting contribution ratios are stored in *cChCoVec* of length $l-2$.

(3) *Threshold calculation.*

This step determines the compression threshold through a weighted aggregation of the distances from intermediate trajectory points to the baseline. Before calculating the perpendicular distances, the WGS-84 longitude and latitude coordinates are transformed into an appropriate local metric projected coordinate system for each study area. The resulting planar coordinates are expressed in metres, ensuring that the perpendicular distances and compression thresholds have consistent physical units. The perpendicular distance of the u-th intermediate point to the baseline is computed as:

$$d_u = \frac{|(y_e - y_u) \times (x_s - x_u) - (y_s - y_u) \times (x_e - x_u)|}{\sqrt{(x_e - x_s)^2 + (y_e - y_s)^2}}, u = 2, 3, \cdots, l-1 \quad (22)$$

where (x_s, y_s) and (x_e, y_e) represent the start and end points of the trajectory, respectively, and (x_u, y_u) is the u-th intermediate point. Only intermediate points are considered, as the distances from the start and end points to the baseline are zero. The resulting distances form a vector *DV* of length l-2.

The compression threshold is then obtained by combining these distances with the corresponding dynamic contribution rates. Specifically, the threshold is computed as:

$$thre = \sum_{u=2}^{l-1} d_u \times cChaCon_u \times \sqrt{(sXChaCon_u)^2 + (sYChaCon_u)^2}, u = 2, 3, \cdots, l-1 \quad (23)$$

where $cChaCon_u$ denotes the contribution rate of COG change, and $sXChaCon_u$ and $sYChaCon_u$ are the contribution rates of the SOG change along the x-axis and y-axis directions, respectively. From a physical perspective, Eq. (23) adjusts the compression threshold according to both the geometric and kinematic characteristics of vessel motion. Trajectory points associated with larger changes in speed or course, or greater geometric deviation, receive greater influence in the weighting

process, reflecting their stronger indication of a change in vessel movement. By contrast, points within relatively stable motion segments have less influence. The resulting adaptive threshold therefore allows SCAADP to retain points that are more informative of changes in navigational behaviour while permitting greater compression in relatively uniform trajectory segments.

This weighted formulation ensures that intermediate points with stronger geometric and dynamic influence contribute more to threshold estimation, while points exhibiting negligible variation in speed or heading are effectively excluded. The resulting threshold is then used by the SCAADP algorithm to distinguish between feature and non-feature points during iterative trajectory compression.

4.2.2. Calculation of dual-component similarity

Trajectory similarity measurement provides the basis for clustering by quantifying differences in vessel navigation patterns. DTW is widely used for this purpose because it accommodates temporal misalignment and varying trajectory lengths, enabling robust similarity assessment between unaligned trajectories. The overall DTW process is illustrated in Fig. 12.

The dual-component design of DCDTW is based on the different structural roles of trajectory points rather than on an assumption that feature points are universally more important than non-feature points. Feature points correspond to locations at which pronounced geometric or kinematic changes occur, such as substantial changes in direction or speed, and therefore characterise major changes in vessel movement. Non-feature points preserve the intermediate evolution of the trajectory and describe local movement continuity between successive feature points. Both components contain information relevant to trajectory comparison, but they describe different aspects of vessel motion.

This distinction is particularly relevant to AIS trajectories because their observations can be dense and non-uniformly distributed in time and space. Under a conventional point-wise DTW formulation, each observation contributes to the accumulated alignment cost. A densely sampled trajectory segment may therefore contribute a relatively large number of matching terms even when the underlying vessel movement changes only gradually. Meanwhile, a manoeuvring segment containing an important change in trajectory structure may be represented by comparatively fewer observations. Treating all observations through a single representation can consequently make the resulting distance sensitive not only to movement characteristics but also to the distribution of sampled points. DCDTW addresses this issue by separating feature and non-feature components, evaluating them according to their respective structural roles, and integrating the resulting components into the final trajectory-distance measure. The purpose is therefore not to discard non-feature information, but to preserve both major structural characteristics and local movement evolution within a unified similarity framework.

Specifically, assume there are two trajectory sequences, $TraObM = \{tom_1, tom_2, \ldots, tom_i, \ldots, tom_m\}$ and $TraObS = \{tos_1, tos_2, \ldots, tos_j, \ldots, tos_s\}$, with lengths of m and s, respectively, where tom_i and tos_j denote the trajectory points in two-dimensional space. The DTW distance $D_{dtw}(TraObM, TraObS)$ quantifies trajectory similarity, with smaller values indicating greater similarity.

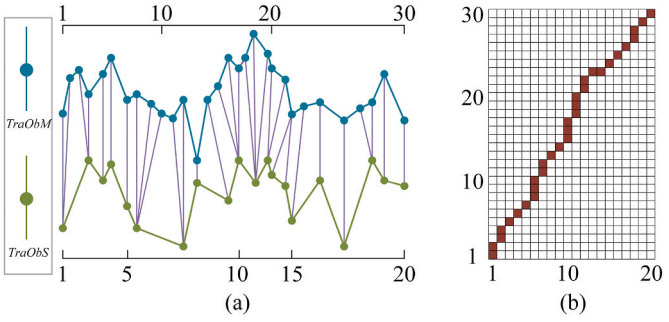

Fig. 12. DTW-based similarity analysis for sequences of different lengths: (a) alignment of the two sequences, and (b) corresponding warping path.

First, pairwise distances between trajectory points are computed to form a distance matrix $disM$ of size $m \times s$. The value of each element in $disM(i, j)$ is calculated using the following formula:

$$disM(i,j) = cauD\left(tom_i, tos_j\right), \tag{24}$$

$$cauD\left(tom_i, tos_j\right) = \sqrt{\left(lonm_i - lons_j\right)^2 + \left(latm_i - lats_j\right)^2}. \tag{25}$$

with ($lonm_i$, $latm_i$) and ($lons_j$, $lats_j$) denoting the longitude and latitude of the i-th point in $TraObM$ and the j-th point in $TraObS$.

Subsequently, a cumulative cost matrix coM of size $(m+1)\times(s+1)$ is constructed and initialized as

$$coM(0,0) = 0, \tag{26}$$

$$coM(i,0) = \infty, i = 1,2,3,\cdots m, \tag{27}$$

$$coM(0,j) = \infty, j = 1,2,3,\cdots, s. \tag{28}$$

The remaining entries of matrix coM are computed via dynamic programming:

$$coM(i,j) = disM(i,j) + \min\begin{cases} coM(i-1,j-1) \\ coM(i-1,j) \\ coM(i,j-1) \end{cases} \tag{29}$$

which accumulates the minimum alignment cost along the optimal warping path. The final DTW distance is obtained from coM (m, s).

Eq. (29) defines a recursive dynamic programming process in which each element $coM(i, j)$ captures the minimum cumulative cost from the starting point pair (tom_1, tos_1) to (tom_i. tom_j). The final similarity measure between trajectories $TraObM$ and $TraObS$ is given by the bottom-right entry coM (m, s).

Trajectory similarity measurement provides the foundation for clustering by quantifying similarities and differences in vessel navigation patterns. DTW is widely used for this purpose because it accommodates temporal misalignment and variable sampling rates. However, its uniform treatment of all trajectory points under a global matching scheme often leads to distortion, as critical geometric features (e.g., turning points) are diluted by numerous non-feature points.

The distinction between feature and non-feature points is motivated by their different structural roles in vessel trajectories. Feature points, such as turning points and locations associated with pronounced changes in course or speed, contain concentrated information about major geometric changes and navigational transitions. In contrast, non-feature points mainly preserve local continuity along relatively stable trajectory segments. Because AIS trajectories are often densely and unevenly sampled, uniformly accumulating point-wise distances may cause numerous structurally redundant points to dominate the DTW cost and weaken the influence of structurally informative points. DCDTW therefore does not assume that non-feature points are unimportant; rather, it models the two components separately so that global trajectory structure and local continuity can both contribute to the final similarity measure through appropriate representations.

To overcome this limitation, this study proposes the DCDTW algorithm, which introduces a paradigm of trajectory point separation and differentiated measurement. Trajectory points are classified into feature points and non-feature points based on their contribution to trajectory shape, and distinct similarity measures are applied to each category. The overall procedure is illustrated in Fig. 13 and summarized below.

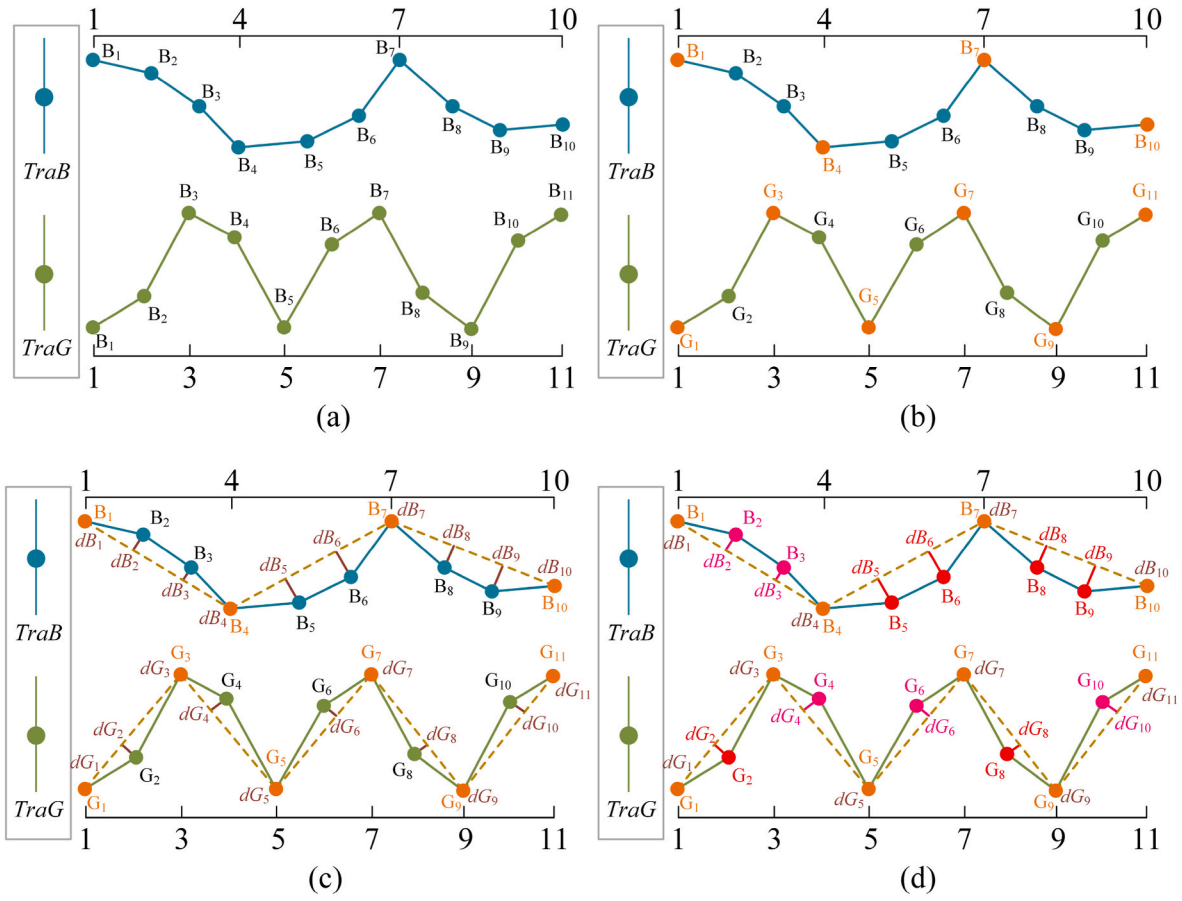

Fig. 13. Schematic diagram of DCDTW-based similarity measurement: (a) original trajectories, (b) SCAADP-based key point detection, (c) segment-wise baseline construction and midpoint-to-baseline distance calculation, and (d) assignment of signed distances for similarity computation.

Step 1. Feature point extraction.

Characteristic points are extracted using the proposed SCAADP algorithm, which iteratively removes redundant points while preserving key geometric features (Section 4.2.1). Given two trajectories, $TraB = \{B_1, B_2, B_3, \ldots, B_{10}\}$ and $TraG = \{G_1, G_2, G_3, \ldots, G_{11}\}$, the resulting feature-point sequences are denoted as $TraBFea = \{B_1, B_4, B_7, B_{10}\}$ and $TraGFea = \{G_1, G_3, G_5, G_7, G_9, G_{11}\}$, respectively (Fig. 13(b)).

Step 2. Feature-point similarity. The similarity between feature-point sequences is computed directly using the standard DTW formulation (Eqs. (24)–(29)), yielding the feature-based distance $D_{fea}(TraBFea, TraGFea)$.

Step 3. Non-feature-point similarity. Using the extracted feature points, each trajectory is segmented into sub-trajectories. Taking trajectory *TraB* as an example, the mathematical representations after segmentation using feature points are given in Eqs. (30) and (33).

For each trajectory sub-segment, a baseline is defined by connecting its start and end points. The perpendicular distance from each intermediate point to its corresponding baseline is computed using Eq. (21), as illustrated in Fig. 13(c). By definition, the distances of the start and end points are set to zero. For the two example trajectories *TraB* and *TraG*, the distances of all points to their respective baselines are assembled into two one-dimensional vectors, denoted as *TraBdVec* and *TraGdVec*.

$$TraB = \{subTraB_1, subTraB_2, subTraB_3\}, \tag{30}$$

$$subTraB_1 = \{B_1, B_2, B_3, B_4\}, \tag{31}$$

$$subTraB_2 = \{B_4, B_5, B_6, B_7\}, \tag{32}$$

$$subTraB_3 = \{B_7, B_8, B_9, B_{10}\}. \tag{33}$$

For instance, in *TraBdVec*, the entries corresponding to feature points (e.g., the 1st, 4th, 7th, and 10th points) are zero. These vectors compactly represent the deviations of non-feature points from their local baselines. Although one-dimensional DTW can be directly applied to these distance sequences, using unsigned distances captures only the magnitude of deviation and ignores directional information. Consequently, non-feature points located on opposite sides of a baseline but at equal distances would be treated as identical, leading to potential similarity distortion. To resolve this issue, the relative position of each non-feature point with respect to its baseline is determined using a vector cross-product, as defined in Eq. (34).

$$cro = (lon_e - lon_s) \times (lat - lat_s) - (lat_e - lat_s) \times (lon - lon_s) \tag{34}$$

where (lon_s, lat_s) and (lon_e, lat_e) correspond to the coordinates of the start and end points of the trajectory segment, respectively, while (lon, lat) denotes the coordinates of any non-feature point.

The sign of the cross-product value *cro* indicates the side of the baseline on which the point lies: a positive value corresponds to the left side (with respect to the start-to-end direction), and a negative value corresponds to the right side. Accordingly, the perpendicular distance is assigned a positive or negative sign.

This signed distance encoding, illustrated in Fig. 13(d), updates the elements of *TraBdVec* and *TraGdVec*. Applying one-dimensional DTW to these signed sequences yields the non-feature similarity measure

$D_{nfea}(TraBdVec,\ TraGdVec)$, which captures both the magnitude and directional consistency of deviations between the two trajectories.

Step 4. The overall trajectory similarity is obtained by combining the feature-point similarity from Step 2 with the non-feature-point similarity from Step 3, as expressed by the following function:

assignment of signed distances for similarity computation.

$$D_{dcdtw}(TraB, TraG) = D_{fea}(TraBFea, TraGFea) + D_{nfea}(TraBdVec, TraGdVec) \quad (35)$$

From a physical perspective, the two components in Eq. (35) describe vessel trajectory similarity at complementary levels. The feature-point component captures the principal geometric structure and major changes in vessel motion, whereas the non-feature-point component describes the local evolution of the trajectory between adjacent feature points. Combining the two components allows DCDTW to account for both the dominant trajectory structure and the local movement characteristics that may be lost when similarity is evaluated using feature points alone.

Compared with the original DTW algorithm, the proposed DCDTW method offers substantially higher computational efficiency. Conventional DTW computes pairwise two-dimensional Euclidean distances between all trajectory points, resulting in a large number of computationally expensive operations, particularly square root evaluations. In contrast, DCDTW confines two-dimensional distance calculations to feature points only, while representing non-feature points as one-dimensional signed distance sequences. This dimensionality reduction markedly decreases the number of square root operations and, consequently, the overall computational cost.

4.3. GPU-accelerated DCDTW measurement method

To enable efficient processing of large-scale vessel trajectory data, both SCAADP and DCDTW are implemented within a GPU parallel computing framework. Section 4.3.1 describes the parallelization strategy for trajectory compression using SCAADP, while Section 4.3.2 presents the GPU implementation of the DCDTW similarity measurement.

The GPU implementation uses different thread-mapping strategies for SCAADP and DCDTW according to their computational characteristics. In SCAADP, trajectory-point-level calculations are mapped to parallel GPU threads so that the geometric and kinematic quantities required for feature-point extraction can be evaluated concurrently. In DCDTW, each trajectory pair is assigned to an independent GPU thread, with each thread responsible for computing the corresponding element of the trajectory similarity matrix. For data organisation, the five AIS attributes are stored in a unified matrix, where T denotes the total number of trajectory points. Additional marker vectors are used to record trajectory membership, feature-point status, and trajectory boundaries. This organisation provides direct indexing of the trajectory data required by each thread and supports parallel processing of the SCAADP and DCDTW computations.

4.3.1. Parallelization of the SCAADP method

Trajectory compression using SCAADP is parallelized by mapping individual trajectory points to independent GPU threads, thereby eliminating the sequential bottlenecks inherent in CPU-based implementations. This fine-grained parallelism enables simultaneous computation across large numbers of trajectory points and significantly improves scalability.

The GPU implementation addresses three key design considerations. First, an efficient data structure is required to support a single bulk transfer of trajectory data from host memory to device memory, while preserving both trajectory membership and point ordering. Second, dedicated kernel functions must be designed to compute trajectory-specific compression thresholds and to coordinate execution across kernels. Third, additional kernel functions are required to perform the compression itself, using the precomputed thresholds to identify feature and redundant points in parallel.

To support these requirements, all trajectory attributes, including timestamp, longitude, latitude, SOG, and COG, are consolidated into a $5\times T$ matrix, referred to as the AM matrix, where T denotes the total number of trajectory points. This structure, defined in Eq. (5) and illustrated in Fig. 14, enables efficient memory access and minimizes data transfer overhead during GPU execution.

To encode trajectory membership and compression results, a Point marker vector (PMV) of length R is introduced. Each element in the PMV identifies both the trajectory to which a point belongs and its status after compression. Positive values indicate feature points, whereas negative values denote redundant points. For example, given three trajectories with 3, 5, and 7 points, respectively, the PMV is initialized as follows:

$$PMV = [1, -1, 1, 2, -2, -2, -2, 2, 3, -3, -3, -3, -3, -3, 3] \quad (36)$$

After the compression process, all points in the first trajectory are identified as feature points, whereas only the third point in the second trajectory and the fourth point in the third trajectory are selected as feature points. Accordingly, the PMV vector is updated as follows:

$$PMV = [1, 1, 1, 2, -2, 2, -2, 2, 3, -3, -3, 3, -3, -3, 3] \quad (37)$$

In the GPU execution model, each trajectory point, represented by its timestamp, spatial coordinates, SOG, and COG, is assigned to an individual thread. A vector-based thread organization is therefore adopted, as illustrated in Fig. 15(a), ensuring a direct and efficient mapping between data elements and GPU threads. This design fully exploits GPU parallelism and provides the computational foundation for high-throughput trajectory compression using SCAADP.

$$TMV = [0, 3, 8, 15] \quad (38)$$

This section describes the GPU-based parallel computation of trajectory compression thresholds. Following the workflow in Section 4.2.1, the computation is decomposed into six stages, each implemented as a dedicated kernel function. The overall workflow is illustrated in Fig. 16(a).

The previously defined marker vector (PMV) identifies the trajectory membership of each point and whether it is retained as a feature point, but it does not explicitly encode trajectory boundaries. To enable parallel identification of trajectory start and end points, an additional Trajectory Marker Vector (TMV) is introduced. Using the example of three trajectories, the TMV is initialized as shown below:

For example, if a thread with index 7 corresponds to the eighth trajectory point, the algorithm first retrieves the absolute value $|PMV[7]| = 2$. This value is then used with the TMV vector to identify trajectory boundaries: the point is the start of the second trajectory if the thread index equals $TMV[1]$, or its end if the index equals $TMV[2]$ - 1. Based on this indexing scheme, six kernel functions are constructed to compute trajectory-specific compression thresholds in parallel.

(1) ***kernel*$_1$**: This kernel computes the geometric and kinematic quantities required for threshold estimation. Specifically, it calculates the perpendicular distance from each trajectory point to its corresponding baseline and decomposes the SOG into its x- and y-axis components. The distances are stored in the vector DV, while the decomposed SOG components are stored in $sXVec$ and $sYVec$, respectively. All three vectors have a length equal to the total number of trajectory points.

Unlike the serial implementation, where distances for start and end points are excluded, the GPU implementation retains these zero-valued entries in DV. This design preserves a one-to-one correspondence

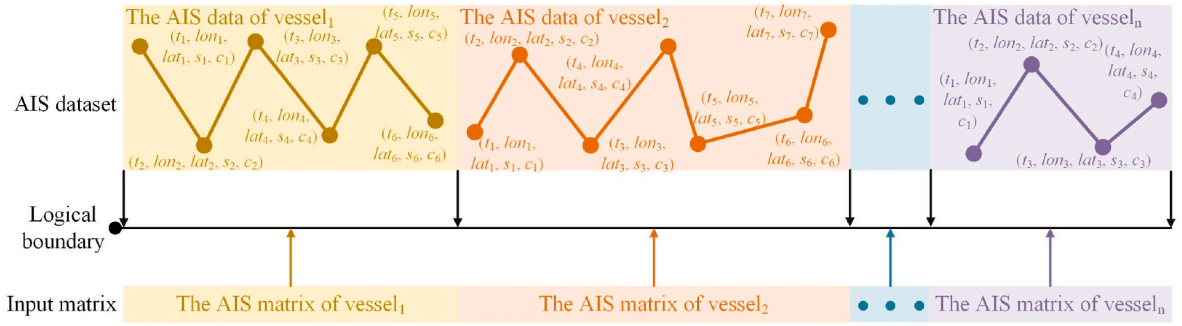

Fig. 14. Optimized memory access patterns for AIS data to maximize bandwidth efficiency.

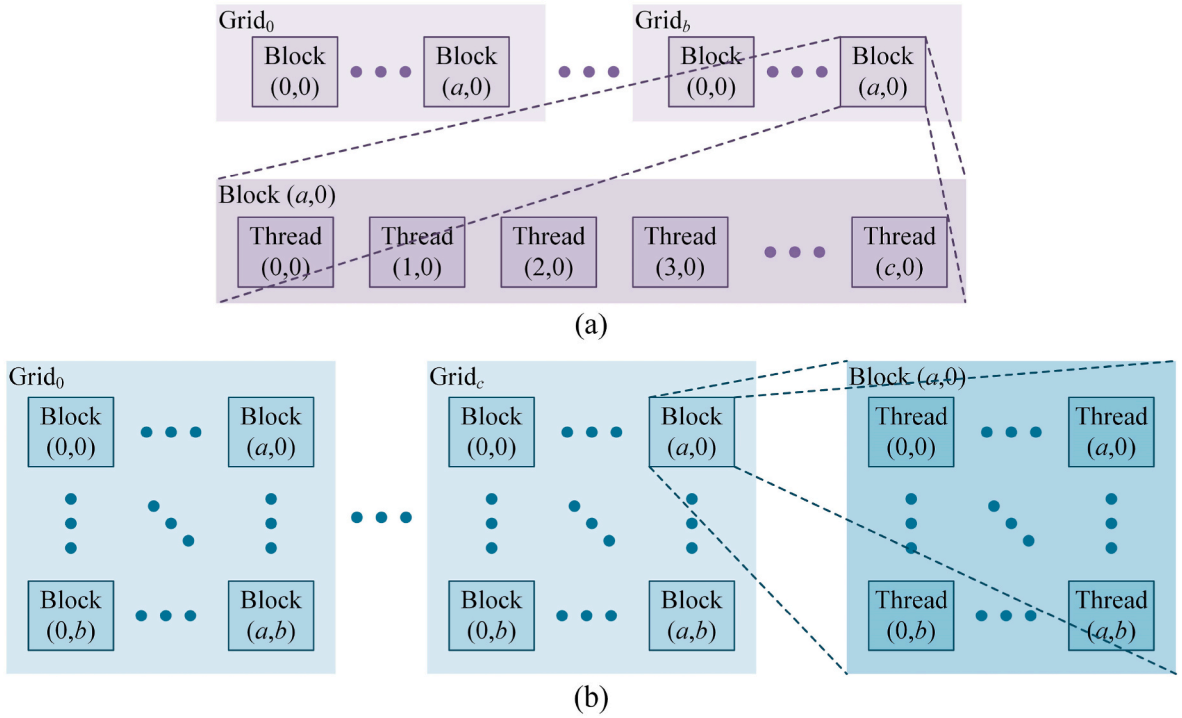

Fig. 15. Design schematic for the parallel thread distribution in computing the (a) trajectory vector and (b) similarity matrix.

between trajectory points and vector indices, simplifying memory access and thread indexing. The execution flow of kernel1 is summarized in Algorithm 2.

Algorithm 2. *kernel*$_1$

Input: *AM*, *PMV*, TMV, *DV*, *sXVec*, sYVec//All elements in the initialized vectors *DV*, *sXVec*, and sYVec are set to 0.
Output: *DV*, *sXVec*, sYVec
1. *th* = blockIdx.x × blockDim.x + threadIdx.x;//It denotes the thread index and simultaneously corresponds to the th trajectory point in the dataset.
2. *staPos* = *TMV*[*PMV*[*th*] – 1];//The position of the starting point of the parent trajectory.
3. *endPos* = *TMV*[*PMV*[*th*]] - 1;//The position of the ending point of the parent trajectory.
4. *lon* = *AM*[1,:][*th*], lat = *AM*[2,:][*th*], s = *AM*[3,:][*th*], c = *AM*[4,:][*th*];//*lon*, *lat*, *s*, and c correspond to the longitude, latitude, SOG, and COG of the th trajectory point that we are processing.
5. lon_s = *AM*[1,:][*staPos*], lat_s = *AM*[1,:][*staPos*];//lon_s and lat_s are the longitude and latitude coordinates of the starting point in the trajectory corresponding to the current trajectory point, respectively.
6. lon_e = *AM*[1,:][*endPos*], lat_e = *AM*[1,:][*endPos*];//lon_e and lat_e denote the longitude and latitude coordinates of the trajectory's endpoint, respectively.

(*continued on next column*)

(*continued*)

7. *sX*, sY = CoSC(*s*, c);//Building upon the SOG decomposition results for trajectory points, as described in Algorithm 1 of Section 4.2.1.
8. *sXVec*[*th*] = *sX*, *sYVec*[*th*] = *sY*;
9. d = cauDis(*lon*, *lat*, lon_s, lat_s, lon_e, lat_e);//The function cauDis is defined by Eq. (21) presented in Section 4.2.1.
10. *DV*[*th*] = *d*;

(2) ***kernel*$_2$**: This kernel computes the rates of change of motion parameters for trajectory points. Specifically, it calculates the rates of change of the SOG components along the *x* and *y* axes (*sXRate*, *sYRate*) and the rate of change for the COG (*cRate*) for all points except the first point of each trajectory. The resulting values are stored in the vectors *sXRaVec*, *sYRaVec*, and *cRaVec*, respectively, each with length equal to the total number of trajectory points.

In the serial algorithm of Section 4.2.1, the first point of each trajectory is excluded from the aforementioned calculations. In the GPU parallel implementation, however, the corresponding entries in *sXRaVec*, *sYRaVec*, and *cRaVec* are set to zero. This preserves index alignment

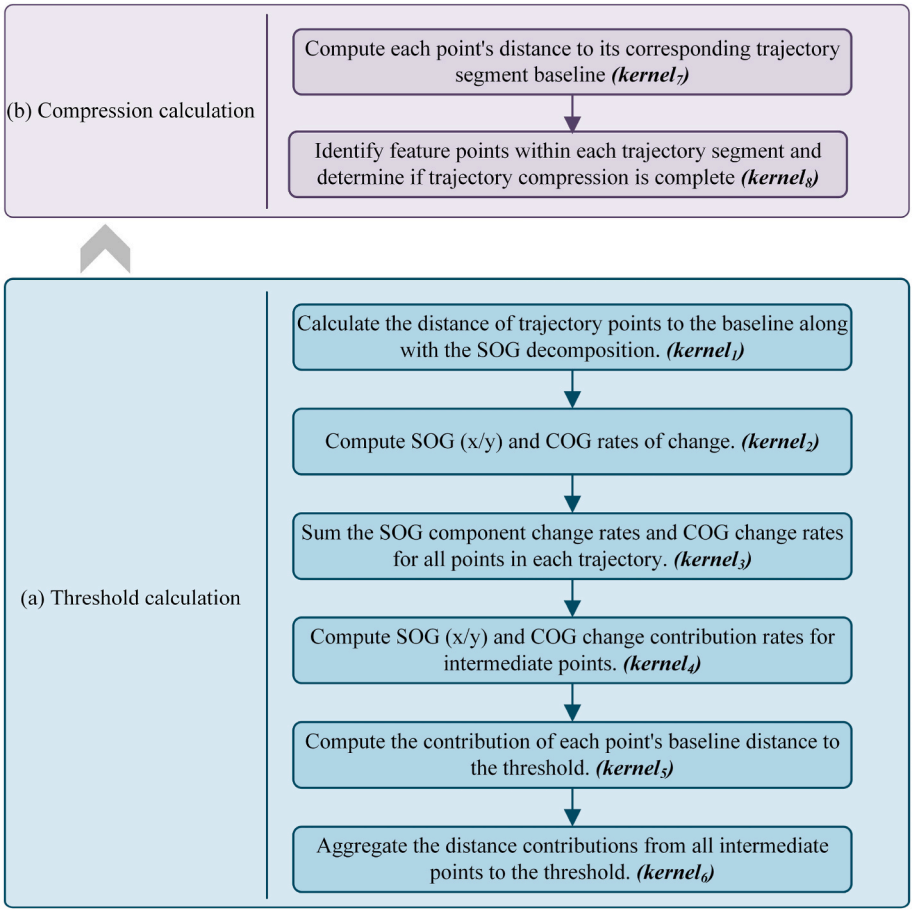

Fig. 16. Key components of the GPU-accelerated SCAADP framework: (a) threshold calculation, and (b) compression calculation.

between vectors and trajectory points, simplifying memory access and parallel computation. The execution flow of *kernel*$_2$ is summarized in Algorithm 3.

Algorithm 3. *kernel*$_2$

Input: *AM*, *sXVec*, *sYVec*, *sXRaVec*, *sYRaVec*, cRaVec//All elements in the initialized vectors *sXRaVec*, *sYRaVec*, and cRaVec are set to 0.
Output: *sXRaVec*, *sYRaVec*, cRaVec
1. th = blockIdx.x × blockDim.x + threadIdx.x;
2. $staPos = TMV[PMV[th] - 1]$;
3. $ts_{th} = AM[0,:][th]$, $ts_{th\text{-}1} = AM[0,:][th\text{-}1]$;//$ts_{th}$ and $ts_{th\text{-}1}$ correspond to the timestamps of the current and the previous trajectory point, respectively.
4. **if** th ! = staPos **then**//Skip the first point of each trajectory in the computation.
5. $sXRate = \frac{|sXVec[th] - sXVec[th-1]|}{ts_{th} - ts_{th-1}}$;
6. $sYRate = \frac{|sYVec[th] - sYVec[th-1]|}{ts_{th} - ts_{th-1}}$;
7. $cRate = \frac{|cVec[th] - cVec[th-1]|}{ts_{th} - ts_{th-1}}$;
8. $sXRaVec[th] = sXRate$; $sYRaVec[th] = sYRate$; $cRaVec[th] = cRate$;
9. **end if**

(3) ***kernel*$_3$**: The kernel calculates trajectory-level cumulative sums of motion change rates using the outputs from *kernel*$_2$. For each trajectory, it accumulates the rates of change of the SOG components (*sXRate*, *sYRate*) and the COG *cRate*, producing the sums *sXRaSum*, *sYRaSum*, and *cRaSum*, respectively.

The resulting trajectory-specific sums are stored in the vectors *sXRaSumVec*, *sYRaSumVec*, and *cRaSumVec*, each with length equal to the number of trajectories in the dataset. The execution flow of *kernel*$_3$ is provided in Algorithm 4.

Algorithm 4. *kernel*$_3$

Input: *TMV*, *sXRaVec*, *sYRaVec*, *cRaVec*, *sXRaSumVec*, *sYRaSumVec*, and cRaSumVec//All elements in the initialized vectors *sXRaSumVec*, *sYRaSumVec*, and cRaSumVec are set to 0.
Output: *sXRaSumVec*, *sYRaSumVec*, and cRaSumVec
1. $thTra$ = blockIdx.x × blockDim.x + threadIdx.x;//The thread index thTra here corresponds to the index of a trajectory, whereas th in Algorithms 3 and 4 maps to the indices of trajectory points.
2. sXRaSum = 0, sYRaSum = 0, cRaSum = 0;//Set to initial values.
3. **for** v = $TMV[thTra]$: $TMV[thTra+1]$ **do**
4. sXRaSum = sXRaSum + $sXRaVec[v]$;
5. sYRaSum = sYRaSum + $sYRaVec[v]$;
6. cRaSum = cRaSum + $cRaVec[v]$;
7. $sXRaSumVec[thTra] = sXRaSum$;
8. $sYRaSumVec[thTra] = sYRaSum$;
9. $cRaSumVec[thTra] = cRaSum$;
10. **end for**

(4) ***kernel*$_4$**: Using the outputs from *kernel*$_2$ and *kernel*$_3$, this kernel computes the contribution rates of motion changes for all trajectory points. Specifically, it calculates the contribution rates of the SOG components (*sXChaCon*, *sYChaCon*) and the COG (*cChaCon*) for each intermediate trajectory point. These results are then stored in the vectors *sXChCoVec*, *sYChCoVec*, and *cChCoVec*, each with length equal to the total number of trajectory points.

In the serial implementation (Section 4.2.1), head and tail points are

excluded from these calculations. In the GPU implementation, the corresponding vector entries are assigned zero values to maintain index alignment and enable efficient parallel access. The execution flow of $kernel_4$ is provided in Algorithm 5.

Algorithm 5. $kernel_4$

Input: *PMV*, *TMV*, *sXRaVec*, *sYRaVec*, *cRaVec*, *sXRaSumVec*, *sYRaSumVec*, *cRaSumVec*, *sXChCoVec*, *sYChCoVec*, and cChCoVec//All elements in the initialized vectors *sXChCoVec*, *sYChCoVec*, and cChCoVec are set to 0.
Output: *sXChCoVec*, *sYChCoVec*, and cChCoVec
1. th = blockIdx.x × blockDim.x + threadIdx.x;
2. $staPos = TMV[PMV[th] - 1]$;
3. $endPos = TMV[PMV[th]\ ] - 1$;
4. **if** th ! = staPos && th ! = endPos **then**//Skip the starting and ending points of the trajectory.
5. $sXChCoVec[th] = \frac{sXRaVec[th]}{sXRaSumVec[|PMV[th]|]}$;
6. $sYChCoVec[th] = \frac{sYRaVec[th]}{sYRaSumVec[|PMV[th]|]}$;
7. $cChCoVec[th] = \frac{cRaVec[th]}{cRaSumVec[|PMV[th]|]}$;
8. **end if**

(5) ***kernel*$_5$**: This kernel computes the weighted contribution of each point's baseline distance to the compression threshold. Based on Eq. (22), it integrates the contribution rates produced by $kernel_4$ with the baseline distance vector *DV* generated by $kernel_1$. The resulting weighted distances are stored in the vector *weiDisVec*, which has length equal to the total number of trajectory points. Entries corresponding to trajectory start and end points are set to zero. The computational procedure is detailed in Algorithm 6.

Algorithm 6. $kernel_5$

Input: *DV*, *sXChCoVec*, *sYChCoVec*, *cChCoVec*, and weiDisVec//All elements in the initialized vector weiDisVec is set to 0.
Output: weiDisVec
1. th = blockIdx.x × blockDim.x + threadIdx.x;
2. $staPos = TMV[PMV[th] - 1]$;
3. $endPos = TMV[PMV[th]\ ] - 1$;
4. **if** th ! = staPos && th ! = endPos **then**
5. $weiDisVec[th] = DV[th] \times cChCoVec[th] \times \sqrt{(sXChCoVec[th])^2 + (sYChCoVec[th])^2}$;
6. **end if**

(6) ***kernel*$_6$**: In the final stage, this kernel computes the compression threshold for each trajectory by summing the weighted distances of its constituent points from *weiDisVec*. The resulting thresholds are stored sequentially, according to trajectory order, in the vector *threVec*. The execution flow of $kernel_6$ is described in Algorithm 7.

Algorithm 7. $kernel_6$

Input: *TMV*, *weiDisVec*, and threVec//All elements in the initialized vector threVec is set to 0.
Output: weiDisVec
1. $thTra$ = blockIdx.x × blockDim.x + threadIdx.x;
2. $threData = 0$;//Set to initial values.
3. **for** v = $TMV[thTra]$: $TMV[thTra+1]$ **do**
4. $threData$ = threData + $weiDisVec[v]$
5. **end for**
6. $threVec[thTra] = threData$;

After threshold computation, the remaining challenge is to efficiently perform parallel iterative extraction of feature points. Building on our previous work, which demonstrated the feasibility of GPU-based parallel iteration for trajectory compression, this study extends that approach and details its implementation.

The parallel iterative process consists of two key operations. First, the distance from each trajectory point to the baseline of its current trajectory segment is dynamically updated. This dynamic behavior arises because newly identified feature points in each iteration subdivide existing segments, generating new baselines for subsequent iterations. Second, the maximum point-to-baseline distance within each segment is compared against the compression threshold to determine whether additional feature points should be extracted. An iteration termination condition is defined such that the process stops when no new feature points are identified.

As illustrated in Fig. 16(b), this iterative procedure is implemented using two kernel functions. The first of these is described below.

(7) ***kernel*$_7$**: The kernel computes the distance from each trajectory point to the baseline of its corresponding trajectory segment during the iterative search for new feature points. While this operation resembles the baseline-distance computation in $kernel_1$, a key distinction is that the computation here is dynamic. In each iteration, newly identified feature points redefine the segment boundaries, requiring recalculation of baselines and point-to-baseline distances.

Accordingly, $kernel_1$ can be viewed as a special case of $kernel_7$, applied only in the initial iteration. To efficiently identify the start and end points of dynamically updated trajectory segments, a two-pointer strategy is employed, as detailed in Algorithm 8.

Algorithm 8. Dynamic Start-End Point Detection

Input: *PMV*, th//th is the thread index and simultaneously corresponds to the th trajectory point in the dataset.
Output: *lp*, rp//lp and rp represent the locations of the start and end points, respectively, within AM for the newly generated trajectory segment.
1. lp = th − 1, rp = th + 1;//Initialization.
2. $laValue = PMV[th]$;
3. **while** ($PMV[lp]$! = *-laValue*) || ($PMV[rp]$! = *-laValue*) **do**
4. **if** $PMV[lp]$! = *-laValue* **then**
5. lp = lp − 1;
6. **end if**
7. **if** $PMV[rp]$! = *-laValue* **then**
8. rp = rp − 1;
9. **end if**
10. **end while**

The seventh kernel integrates the point-to-baseline distance computations described in Algorithms 2 and 8 into a unified implementation, enabling dynamic distance evaluation during iterative feature-point extraction.

(2) ***kernel*$_8$**: This kernel governs the iterative detection and termination process through coordinated updates of the marker vectors *PMV* and *TMV*. It performs two key functions. First, based on the distance results produced by $kernel_7$, it identifies newly detected feature points within trajectory segments and updates the corresponding entries in *PMV* from negative to positive values. Second, it determines whether compression for a given trajectory has converged by monitoring *TMV*. Once no new feature points are detected, the corresponding *TMV* entry is updated from positive to negative, indicating termination for that trajectory.

Through dynamic sign updates in *PMV* and *TMV*, $kernel_8$ drives the iterative process, enabling continuous feature-point discovery while simultaneously controlling termination. Fig. 17 illustrates the evolution of PMV and TMV for a single trajectory. Initially, only the first and last points are marked as feature points in PMV, while all TMV entries are positive (Fig. 17(a) and (b); Eqs. (39) and (40)).

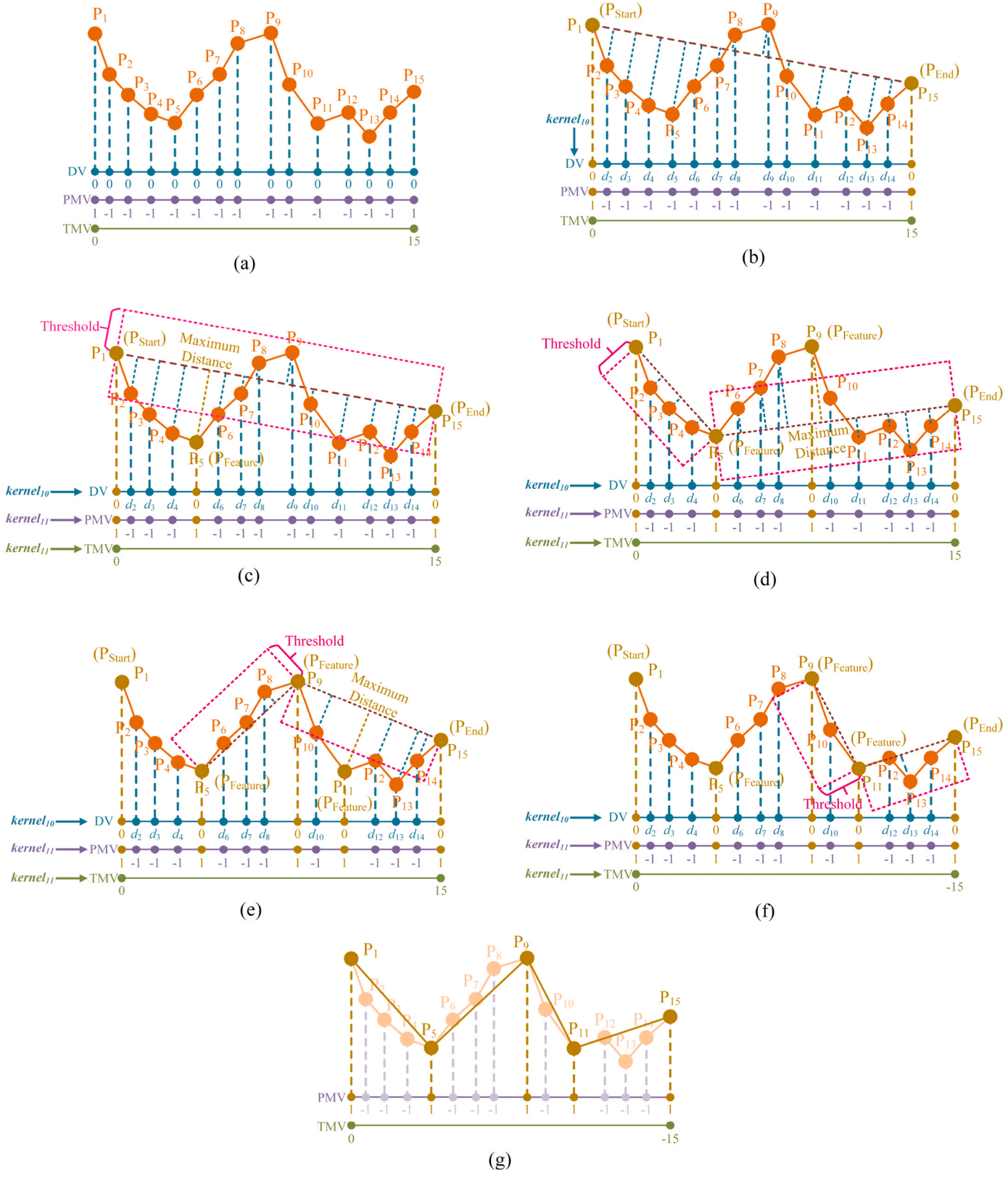

Fig. 17. Workflow of the GPU-accelerated SCAADP trajectory compression: (a) original trajectory with labels *DV*, *PMV*, and *TMV*, (b) parallel computation of *d*, (c) search for maximal *d* to identify feature points, (d) trajectory segmentation and update of new feature points, (e, f) iterative identification of feature points, and (g) final compressed trajectory.

$$PMV = [1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1], \tag{39}$$

$$TMV = [0, 15]. \tag{40}$$

During successive iterations, newly detected feature points are progressively marked in *PMV* (Fig. 17(c)–(e)). When no further feature points are identified, *PMV* remains unchanged, and all *TMV* entries are set to negative values, signaling convergence (Fig. 17(f) and (g)). The final states of *PMV* and *TMV* are given in Eqs. (41) and (42).

$$PMV = [1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1], \tag{41}$$

$$TMV = [0, -15]. \tag{42}$$

This single-trajectory example illustrates the complete evolution of the marker vectors during parallel iteration. To demonstrate multi-trajectory parallelism, Fig. 18 compares the initial and final states of *PMV* and *TMV* for three trajectories before and after parallel computation, showing that feature points across all trajectories are correctly identified in a single unified process.

4.3.2. Parallelization of dual-component similarity

The GPU implementation of the DCDTW similarity measurement is based on assigning independent trajectory pairs to parallel threads. Computation is organized around the similarity matrix*SM* (defined in Section 3.1, Eq. (6)), which stores pairwise trajectory similarity values and serves as direct input for clustering algorithms such as DBSCAN and K-means.

As illustrated in Fig. 19 (b), the thread layout is aligned with the dimensions of *SM*. Each GPU thread computes one matrix element: the thread at position [*m*, *n*] evaluates the DCDTW similarity between the (*m*+1)-th and (*n*+1)-th trajectories, as shown in Fig. 19. This design enables all trajectory pairs to be processed concurrently. In contrast to CPU-based serial execution, where similarity values are computed sequentially, the GPU framework evaluates the entire similarity matrix in parallel, yielding substantial performance gains for large-scale datasets.

Compared with SCAADP, DCDTW requires greater computational effort per thread, but its parallelization is conceptually simpler. All similarity computations are completed within a single kernel function, avoiding inter-kernel coordination. The overall parallel execution procedure is detailed in Algorithm 9.

In Algorithm 9, the device function DCDTW computes the similarity between two trajectories using the formulation described in Section 4.2.2 and Eq. (35). This encapsulated design ensures clarity, modularity, and efficient execution within the GPU parallel framework.

The main memory limitation arises from the pairwise similarity matrix, whose storage requirement grows as $O(N^2)$ with the number of trajectories. For very large AIS datasets, GPU global memory may therefore become a limiting factor. Additional computational bottlenecks include host-device data transfer, global-memory access latency, and workload imbalance caused by heterogeneous trajectory lengths. These limitations motivate future development of memory-efficient, batched, and multi-GPU implementations.

Algorithm 9. Parallel DCDTW Computation

Input: TraSet//TraSet is defined as the container of all trajectory data in the dataset, represented as {Tra_1, Tra_2, …, Tra_{tsL}}, The length of this set, denoted by *tsL*, is the total number of trajectories. Each *Tra* is also a sequence that comprises the longitude and latitude coordinates of trajectory points. Its formal definition resembles Eqs. (1) and (2) in Section 3.1, but excludes the timestamp, SOG, and COG fields present in Eq. (2).
Output: SM
1. *thx* = threadIdx.x + blockIdx.x × blockDim.x;
2. *thy* = threadIdx.y + blockIdx.y × blockDim.y;
3. Initialize *SM*[0,0] = 0;
4. *dcdtwData* = DCDTW(*TraSet*[*thx*], *TraSet*[*thy*]);
5. *SM*[*thx*, *thy*] = *dcdtwData*;

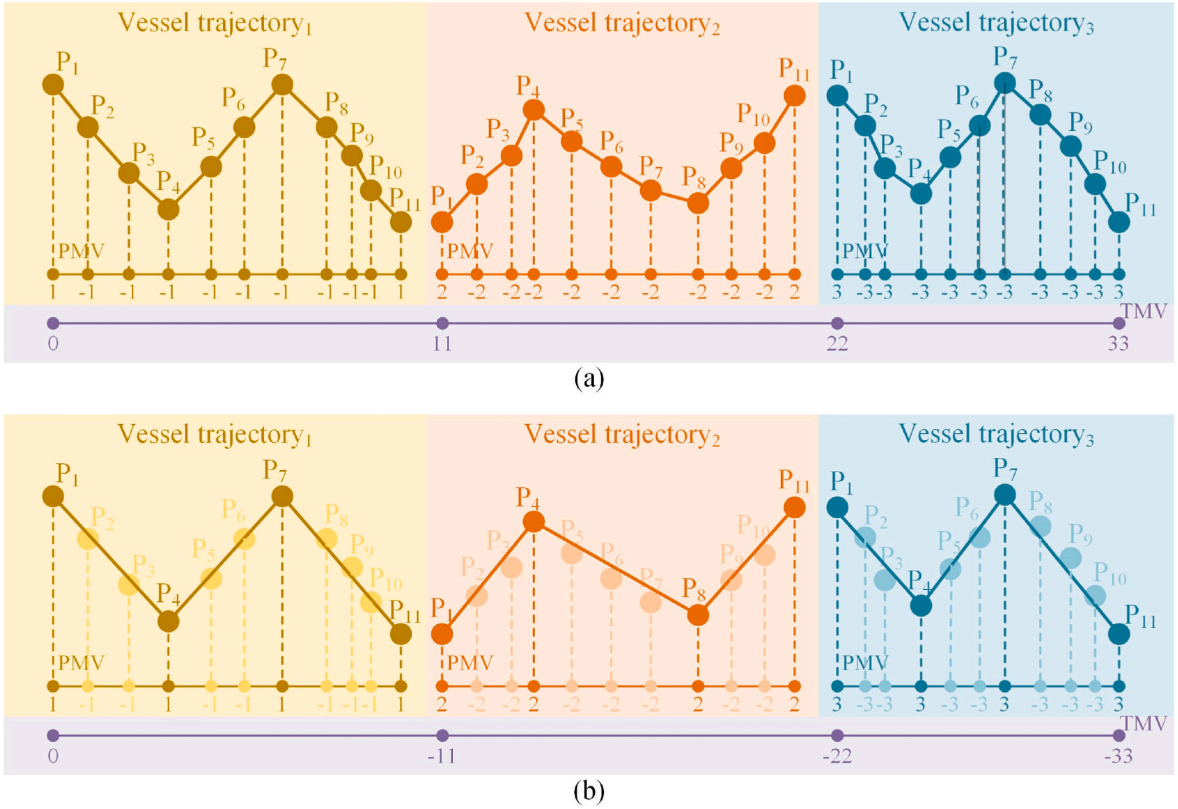

Fig. 18. State transition of the *PMV* and *TMV* labels induced by data compression: (a) pre-compression initialization, and (b) post-compression state.

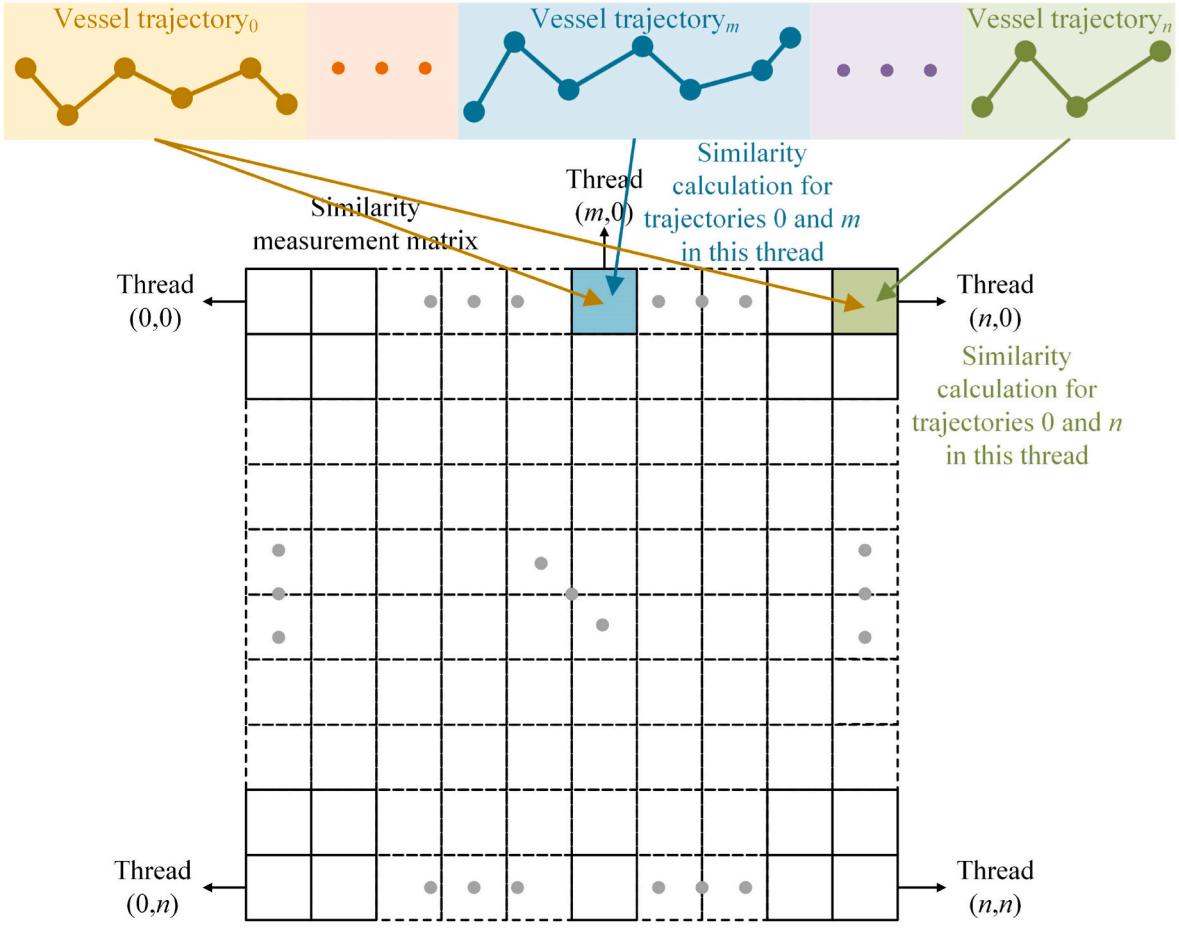

Fig. 19. Computing the parallel mapping relationship between DCDTW and the similarity measurement matrix between two trajectories.

4.3.3. Computational complexity

For two trajectories containing m and n points, conventional DTW has a time complexity of $O(mn)$. In DCDTW, the trajectories are divided into feature and non-feature components. Let m_f and n_f denote the numbers of feature points in the two trajectories, and let mn_f and nn_f denote the sizes of their corresponding non-feature representations. The computational cost of the component-wise comparisons depends on these sequence lengths and remains bounded by $O(mn)$ in the worst case. Therefore, DCDTW retains the same worst-case asymptotic complexity class as conventional DTW.

The computational distinction lies primarily in the operations required by the two components. The feature component uses two-dimensional spatial-distance calculations to characterise major trajectory structures, whereas the non-feature component is compared using one-dimensional signed-distance representations. Consequently, the latter requires simpler elementary distance operations. This reduces the arithmetic workload associated with non-feature-point comparison without changing the worst-case asymptotic complexity.

For N trajectories with an average length L, exhaustive pairwise trajectory comparison involves $O(N_2)$ trajectory pairs, resulting in a worst-case computational requirement of approximately $O(N_2L_2)$. The GPU implementation parallelises these independent trajectory-pair computations and thereby reduces wall-clock execution time through concurrent processing; it does not alter the asymptotic arithmetic complexity of exhaustive pairwise comparison.

The scalability of the GPU implementation is constrained not only by arithmetic computation but also by memory requirements and data movement. A complete pairwise trajectory-distance matrix requires $O(N_2)$ storage, which can become a limiting factor for very large trajectory collections. In addition, the trajectory data and intermediate variables required during similarity computation occupy GPU global memory. Potential performance bottlenecks include host–device data-transfer overhead, global-memory access latency, and workload imbalance caused by heterogeneous trajectory lengths and different numbers of extracted feature points.

For datasets exceeding the available GPU memory, the pairwise computation can in principle be divided into batches or matrix tiles so that only a subset of trajectory pairs is processed at one time. Such a strategy can reduce peak memory demand, although it may introduce additional data-transfer and scheduling overhead. Further scalability could also be investigated through multi-GPU or distributed implementations. These issues become increasingly important for continuous or substantially larger AIS datasets and constitute directions for future optimization of the current GPU implementation.

5. Experimental results and implications

The validation of the proposed GPU-accelerated DCDTW algorithm focuses on two primary aspects. First, both qualitative and quantitative comparisons with conventional DTW and existing optimized variants are presented, demonstrating the superior matching performance of DCDTW and the effectiveness of the proposed SCAADP-based feature extraction. Second, the computational efficiency of the parallelized implementations is quantitatively assessed by comparing their execution times with those of the corresponding serial CPU versions using the speedup ratio. All experiments are conducted on a dedicated GPU computing platform, the configuration of which is summarized in Table 2.

Table 2
Computing infrastructure.

Hardware Environment	Model	Software Environment	Version
CPU	Intel Xeon Gold 6330 × 2	Python	3.13.5
Host Memory	32G DDR4 ECC-R 2933 × 8		
GPU Global Memory	Tesla L20 × 2 48 GB × 2	CUDA	11.7

5.1. Evaluation metrics

This section defines the quantitative criteria used to evaluate the proposed SCAADP and DCDTW algorithms. Metrics for trajectory compression (feature-point extraction) and trajectory similarity measurement are presented in Sections 5.1.1 and 5.1.2, respectively.

5.1.1. Evaluation metrics for compression methods

To assess the performance of the proposed SCAADP algorithm relative to existing compression methods, three metrics are employed: Compression Ratio (CR), Trajectory Points-based DTW (TP-DTW), and Speedup Ratio (SR).

CR and TP-DTW reflect a fundamental trade-off in trajectory compression. While a higher CR indicates greater point reduction, excessive compression may distort trajectory structure. Consequently, the primary objective is to minimize TP-DTW, thereby preserving the geometric fidelity of the original trajectory, before comparing CR values across methods. In addition, SR is introduced to evaluate computational acceleration achieved by the GPU-based implementation.

CR measures the reduction in trajectory points after compression and is defined as

$$CR = 100\% \times \left(1 - \frac{Num_{com}}{Num_{raw}}\right) \tag{43}$$

where Num_{raw} and Num_{com} denote the number of trajectory points before and after compression, respectively.

TP-DTW, introduced by Li et al. (2024b), quantifies the similarity between original and compressed trajectories by computing the DTW distance between vectors of signed perpendicular distances to the baseline. The sign information preserves relative positional relationships, yielding a more sensitive measure of structural distortion caused by compression.

SR evaluates computational efficiency by comparing execution times under CPU serial and GPU parallel frameworks:

$$SR = \frac{CPUT}{GPUT} \tag{44}$$

where *CPUT* denotes the CPU processing time excluding I/O overhead, and *GPUT* includes both GPU computation time and host-device data transfer. This definition provides a comprehensive assessment of practical acceleration performance for large-scale trajectory datasets.

5.1.2. Evaluation metrics for similarity measurement

To quantitatively evaluate the effectiveness of the DCDTW algorithm in trajectory similarity measurement, three clustering metrics are employed: the Silhouette Coefficient (SC), the Calinski–Harabasz Index (CHI), and the Davies–Bouldin Index (DBI) (Ashari et al., 2023).

Trajectory similarity matrices generated by different similarity measures are used as inputs to a clustering algorithm (DBSCAN). To satisfy the input requirements of SC, CHI, and DBI, each trajectory is first embedded as a single point using Multidimensional Scaling (MDS), which directly operates on the similarity matrix. This procedure ensures methodological consistency between similarity computation and clustering evaluation.

SC measures intra-cluster cohesion and inter-cluster separation, with higher values indicating better clustering quality. CHI quantifies the ratio of between-cluster to within-cluster dispersion, where larger values denote superior separation. DBI evaluates cluster similarity, with lower values reflecting more distinct clusters. Together, these metrics provide a comprehensive assessment of similarity measurement performance.

In addition to clustering quality, the computational efficiency of DCDTW is evaluated using the Speedup Ratio (SR) to compare CPU-based and GPU-accelerated implementations.

5.2. Benchmarking methods

This section presents the baseline methods used in the comparative experiments for the SCAADP and DCDTW algorithms, which are detailed in Sections 5.2.1 and 5.2.2, respectively.

5.2.1. Compression methods for benchmarking

The performance of the proposed SCAADP algorithm is compared against four representative trajectory compression methods: the classical DP algorithm and three adaptive DP-based variants.

(a) DP (Visvalingam and Whyatt, 1990): DP recursively removes redundant points based on a predefined distance threshold. Threshold values of 0.1 m, 0.5 m, 1 m, 5 m, 10 m, 20 m, 50 m, and 100 m are tested to assess sensitivity to parameter selection.
(b) ADP (Liu et al., 2019): ADP adaptively determines a trajectory-specific compression threshold based on average point-to-baseline distance and overall trajectory morphology.
(c) ADPS (Li et al., 2022): ADPS incorporates the rate of change of SOG into threshold estimation, combining geometric and dynamic information.
(d) ADPSC (Li et al., 2024): ADPSC further integrates spatial offset weighting with SOG change rates to improve feature-point preservation.

5.2.2. Similarity measurement methods for benchmarking

To evaluate the effectiveness of the proposed DCDTW algorithm, seven representative trajectory similarity measurement methods are selected for comparative analysis. These baselines include the classical DTW algorithm and six widely used DTW-based variants that enhance alignment accuracy through shape, derivative, or contextual information.

(a) DTW (Tao et al., 2023): It measures similarity by identifying an optimal alignment path through dynamic programming, allowing temporal distortion and minimizing the cumulative distance between matched points.
(b) DDTW (Muscillo et al., 2007): It replaces raw value comparisons with first-order derivatives, enabling improved alignment based on trajectory shape characteristics.
(c) DDDTW (Górecki and Łuczak, 2013): It combines raw values and derivatives within a weighted distance function, jointly capturing magnitude differences and shape information.
(d) SW-DTW (Folgado et al., 2018): It improves robustness by integrating local amplitude and derivative features within a sliding-window distance framework.
(e) eDTW (Jiang et al., 2020): It detects salient rising and falling events and uses them to guide alignment, improving performance for trajectories with differing sampling rates.
(f) ACDTW (Li et al., 2020): It introduces an adaptive penalty mechanism to regulate many-to-one and one-to-many matches during alignment.
(g) CDC-DTW (W. Li et al., 2023): It incorporates local contextual density into weighted DTW, enhancing similarity measurement for time series with heterogeneous sampling frequencies.

5.3. Experimental results on compression methods

This section presents comparative experimental results for trajectory compression and similarity measurement. The performance of the proposed SCAADP and DCDTW algorithms is analyzed in Sections 5.3.1 and 5.3.2, respectively.

5.3.1. Effectiveness comparison

The proposed SCAADP algorithm overcomes key limitations of the classical DP method and its adaptive variants. Traditional DP relies on manually specified thresholds, which require expert tuning and fail to account for trajectory-specific kinematic characteristics, such as variations in SOG and COG. Although existing ADP-based methods adapt thresholds using geometric properties, they do not explicitly incorporate dynamic changes in SOG and COG between consecutive points.

In contrast, SCAADP determines trajectory-specific thresholds by jointly integrating geometric structure with dynamic information, enabling effective removal of redundant points while preserving critical feature points. Table 3 summarises the distribution of thresholds obtained by SCAADP across the two study areas, including the maximum, minimum, mean, and standard deviation. Overall thresholds in the CFDP region are lower than those in the CSJP region, reflecting the greater navigational complexity and prevalence of nonlinear trajectories in CSJP.

The visual comparison in Fig. 20 further demonstrates that trajectories compressed using SCAADP retain their essential structural characteristics without observable distortion when projected onto nautical charts.

This section presents a quantitative comparison between the proposed SCAADP algorithm and the four baseline methods outlined in Section 5.2.1, using the CR and TP-DTW as evaluation metrics (Section 5.1.1). The evaluation results for different compression algorithms across the two water areas are summarized in Tables 4 and 5.

For the original DP algorithm, low thresholds (0.1 m, 0.5 m, and 1.0 m) yield low TP-DTW values, indicating high trajectory fidelity, but result in poor CR due to limited removal of redundant points. Increasing the threshold to 5 m leads to a sharp increase in CR, accompanied by a substantial rise in TP-DTW, reflecting pronounced structural distortion. Beyond this threshold, CR gains plateau while TP-DTW continues to increase. These results confirm that a uniform threshold cannot effectively balance compression efficiency and trajectory fidelity across heterogeneous trajectories.

These results also illustrate the sensitivity of conventional DP to the manually specified compression threshold. As the threshold increases, the compression ratio increases, but trajectory distortion measured by TP-DTW also becomes progressively larger. In contrast, SCAADP avoids manual selection of a global threshold by determining trajectory-specific thresholds from the geometric and kinematic characteristics of each trajectory. This adaptive mechanism reduces the dependence of compression performance on manually tuned threshold parameters.

The adaptive variants ADP, ADPS, and ADPSC address this limitation by deriving trajectory-specific thresholds. ADP relies solely on geometric deviation, whereas ADPS and ADPSC incorporate dynamic information through SOG variation, yielding improved alignment with vessel motion characteristics. Building on this progression, the proposed SCAADP algorithm further integrates both SOG and COG variations to achieve more accurate threshold estimation.

Table 3
Compression thresholds derived from the SCAADP algorithm for the CFDP and CSJP study regions.

Research Waters	Maximum Threshold	Minimum Threshold	Mean Threshold	Standard Deviation Threshold
CFDP	77.517	0.128	2.984	2.312
CSJP	98.201	0.476	26.431	21.589

As shown in Tables 4 and 5, SCAADP consistently achieves the lowest TP-DTW values in both study areas, indicating superior preservation of trajectory structure. Compared with ADPSC, the reduction in CR is negligible (0.189% and 0.367%), while TP-DTW decreases substantially (8.620% and 14.796%), demonstrating a clear improvement in compression quality with minimal efficiency loss. These results highlight the benefit of jointly incorporating SOG and COG dynamics.

The parameters involved in the proposed framework can be distinguished between algorithmic quantities used in trajectory representation and implementation settings used for parallel computation. For trajectory compression, conventional DP requires a predefined global distance threshold, and its performance is sensitive to this choice. As shown in Tables 4 and 5, increasing the DP threshold progressively increases the compression ratio but also increases TP-DTW, indicating greater deviation between the compressed and original trajectories. This illustrates the inherent trade-off between compression efficiency and trajectory preservation when a fixed threshold is used.

In contrast, SCAADP does not require manual selection of a global compression threshold. For each trajectory, the compression threshold is determined adaptively from its geometric deviation together with variations in SOG and COG. Trajectories with different geometric and kinematic characteristics can therefore receive different thresholds without a dataset-level search for a fixed optimal value. The method is consequently less dependent on manual threshold selection, although its output naturally remains dependent on the observed geometric and motion characteristics from which the adaptive threshold is derived.

The GPU implementation introduces execution-related settings but does not modify the mathematical formulation of SCAADP or DCDTW. In the implemented parallelization strategy, trajectory-point-level operations required by SCAADP and trajectory-pair-level operations required by DCDTW are distributed across GPU threads, while AIS attributes and trajectory markers are organised to support parallel access. GPU execution settings influence computational efficiency and hardware utilisation but do not constitute model hyperparameters and do not alter the intended numerical definition of the resulting trajectory distances. The specific implementation settings used in the experiments are reported to facilitate reproducibility.

The DP results demonstrate a clear dependence on the prescribed compression threshold. For both study areas, increasing the threshold substantially increases the compression ratio, but this is accompanied by increasing TP-DTW values, reflecting greater trajectory distortion. The appropriate fixed threshold therefore depends on the desired balance between data reduction and trajectory preservation. SCAADP addresses this parameter-selection problem by deriving a trajectory-specific threshold from geometric and kinematic characteristics rather than applying the same manually selected threshold to all trajectories. This adaptive mechanism reduces the dependence of the compression procedure on a globally tuned distance threshold.

Tables 4 and 5 report the mean and standard deviation of TP-DTW between compressed and original trajectories in both study areas, providing an overall assessment of compression fidelity. Fig. 21 presents trajectory-level TP-DTW results, showing consistently low values across trajectories, which indicates that SCAADP preserves trajectory structure effectively after compression.

In addition to the quantitative analysis, a qualitative evaluation is conducted through visual inspection. Although Fig. 22 illustrates the overall compression results, the large number of trajectories limits detailed examination of local structural features. Therefore, four representative trajectories are selected from each study area to enable clearer comparison. The selected trajectories for the two regions are shown in Figs. 22 and 23, respectively.

Figs. 24 and 25 compare the compression results of four representative trajectories from the CFDP and CSJP regions using SCAADP and the benchmark methods. For the original DP algorithm, only results with thresholds of 0.1 m and 100 m are shown to illustrate performance

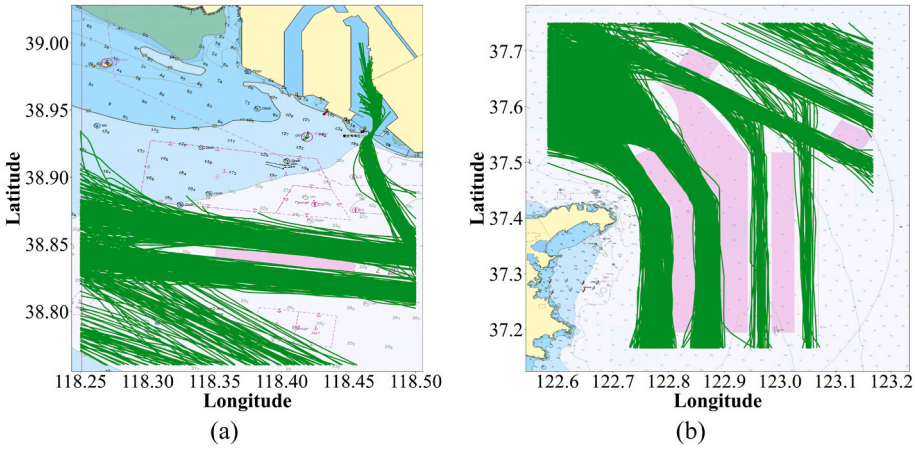

Fig. 20. Compressed trajectories (via SCAADP) in the (a) CFDP and (b) CSJP regions.

Table 4
Performance assessment of trajectory compression in CFDP, comparing the proposed SCAADP with original DP (i.e., 0.1 m, 0.5 m, 1 m, 5 m, 10 m, 20 m, 50 m, and 100 m), ADP, ADPS, and ADPSC algorithms.

Method	Threshold (m)	CR (%) ↑	TP-DTW ↓	
			Mean	Standard deviation
DP	0.100	17.298	0.014	0.017
	0.500	48.895	0.051	0.052
	1.000	64.935	0.093	0.094
	5.000	88.313	0.411	0.432
	10.000	93.173	0.695	0.746
	20.000	96.950	1.040	1.014
	50.000	98.188	1.516	1.412
	100.000	98.660	1.937	1.841
ADP	-	96.883	0.902	0.894
ADPS	-	94.472	0.737	0.793
ADPSC	-	91.166	0.522	0.690
SCAADP	**-**	**90.993**	**0.477**	**0.582**

Table 5
Performance assessment of trajectory compression in CSJP, comparing the proposed SCAADP with original DP, ADP, ADPS, and ADPSC algorithms.

Method	Threshold (m)	CR (%) ↑	TP-DTW ↓	
			Mean	Standard deviation
DP	0.100	12.330	0.021	0.113
	0.500	38.667	0.071	0.119
	1.000	55.756	0.137	0.141
	5.000	85.186	0.689	0.504
	10.000	91.412	1.266	0.901
	20.000	95.143	2.039	1.378
	50.000	97.202	3.350	2.348
	100.000	98.009	4.549	2.998
ADP	-	95.936	2.478	2.007
ADPS	-	94.092	2.052	1.834
ADPSC	-	90.129	1.176	1.021
SCAADP	**-**	**89.798**	**1.002**	**0.857**

under extreme parameter settings. Trajectories compressed using DP with a 100 m threshold, as well as those processed by ADP and ADPS, exhibit noticeable structural distortion. In contrast, trajectories compressed using DP with a 0.1 m threshold, ADPSC, and SCAADP closely preserve the original trajectory structure.

Tables 6 and 7 report the corresponding CR and TP-DTW values for the representative trajectories. Although DP with a 0.1 m threshold yields low TP-DTW values, its CR remains limited, indicating insufficient removal of redundant points. Compared with ADPSC, SCAADP achieves a higher CR with only a marginal increase in TP-DTW, while

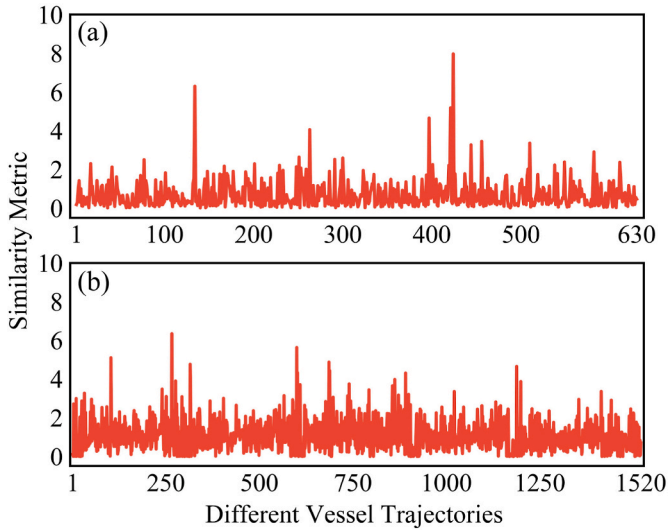

Fig. 21. Performance of the TP-DTW assessment metric for trajectory compression across the (a) CFDP and (b) CSJP regions.

maintaining structural consistency, as confirmed by Figs. 24 and 25. Taken together, the qualitative and quantitative results demonstrate that SCAADP more effectively eliminates redundant points while reliably preserving key trajectory features.

5.3.2. *Efficiency acceleration with GPU parallelization*

The computational efficiency of the proposed SCAADP algorithm under GPU parallelization is evaluated using the SR, with performance compared against a CPU-based serial implementation. Tables 8 and 9 report execution times and SR values for both platforms across the two study areas.

For the original datasets, which contain a relatively small number of trajectories (630 in CFDP and 1520 in CSJP), the observed speedups are modest (SR = 1.063 and 1.208, respectively). To assess scalability, the datasets were artificially expanded by factors of 5, 10, 50, and 100. Under these enlarged workloads, GPU acceleration becomes increasingly pronounced. In CFDP, SR increases progressively with dataset size, exceeding one order of magnitude at large scales. A similar but more pronounced trend is observed in CSJP, where SR exhibits even larger gains as the dataset size grows.

Overall, these results demonstrate that while GPU acceleration offers limited benefits for small datasets, it delivers substantial performance improvements for large-scale trajectory compression tasks, confirming

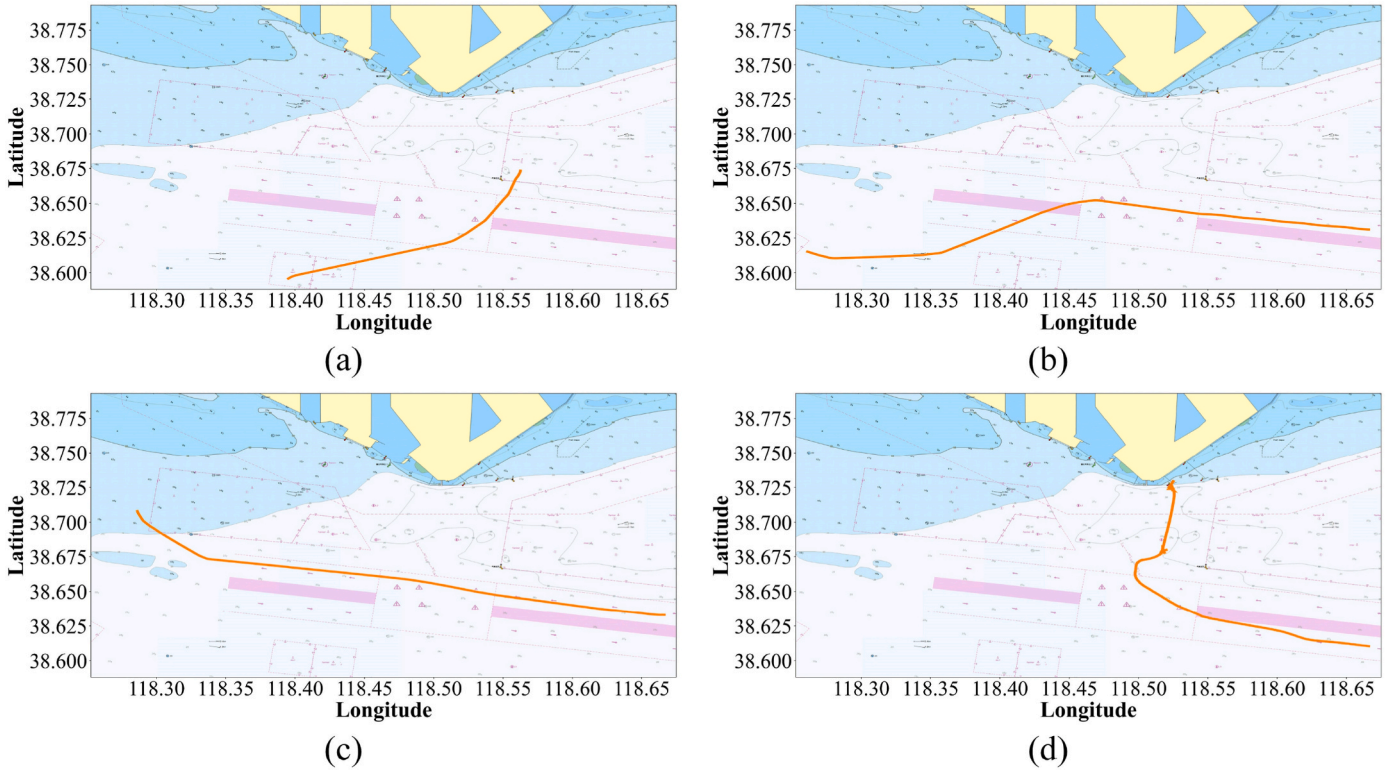

Fig. 22. Distribution characteristics of representative trajectories in the CFDP region, employed to evaluate the SCAADP algorithm's performance: (a) MMSI 229287000, (b) MMSI 232005278, (c) MMSI 232008054, and (d) MMSI 249331000.

the scalability and practical advantage of the proposed parallel implementation.

5.4. Experimental results on similarity measurement

This section evaluates the effectiveness and practical utility of the proposed DCDTW similarity measurement method. Results are reported in terms of representative trajectory comparison, large-scale clustering performance, and visual validation of clustering outcomes.

5.4.1. Effectiveness comparison

To provide an intuitive assessment of similarity measurement performance, four representative trajectories were selected from each study area, as shown in Fig. 26. These trajectories capture the dominant navigational patterns and morphological diversity in the CFDP and CSJP regions. Using one trajectory as the reference, similarity values were computed against the remaining trajectories using DCDTW and six DTW-based baseline methods.

Using the representative trajectories, a core trajectory (v_1) was selected in each study area, and similarity values between v_1 and the remaining trajectories were computed using different algorithms. The resulting similarity measures for the CFDP and CSJP areas are reported in Tables 10 and 11, respectively, enabling direct comparison across methods.

In the CFDP area, trajectories v_1 and v_2 operate within the same waterway and should therefore exhibit the smallest similarity distance. Trajectory v_3, although located in a different waterway, follows a nearly straight path similar to v_1, and its similarity distance should be only marginally larger. In contrast, trajectory v_4 represents a vessel entering the port and exhibits a markedly different morphology, implying a substantially larger distance from v_1.

Several baseline methods fail to reflect these expected relationships. In particular, eDTW, ACDTW, and CDC-DTW produce similarity values exceeding 500 for the v_1-v_2 pair, which is inconsistent with physical scale, given that inter-point distances in the WGS-84 coordinate system rarely exceed two orders of magnitude. DDDTW further exhibits disproportionate increases between trajectory pairs, while DDTW and SW-DTW also yield implausibly large values for the morphologically distinct v_1-v_4 pair. Although DTW performs comparatively better, its similarity values remain inflated. By contrast, DCDTW produces similarity measures within a reasonable order of magnitude and preserves the correct relative ordering across trajectory pairs.

A consistent pattern is observed in the CSJP area. Trajectories v_1 and v_2, which share the same shipping lane, exhibit the smallest similarity values, while trajectory v_3, located in an adjacent lane with opposite course, shows only a slight increase. Trajectory v_4, positioned in a distant lane with a distinct linear pattern, yields the largest distance. As shown in Table 11, DCDTW produces similarity values on the order of tens, whereas all baseline methods exceed 200, with eDTW and ACDTW reaching several orders of magnitude higher. Verification against the maximum inter-point distance between v_1 and v_4 (approximately 30) confirms the metric plausibility of DCDTW.

Overall, DCDTW is the only method that consistently yields similarity values with realistic magnitudes and correct relational ordering, demonstrating superior effectiveness in vessel trajectory similarity measurement.

The advantage of the dual-component strategy is expected to be more pronounced when trajectories exhibit heterogeneous sampling densities, unequal numbers of trajectory points, or distinct local manoeuvring patterns despite similar overall routes. Under such conditions, conventional DTW may accumulate excessive contributions from densely sampled but structurally redundant points. By separately modelling feature and non-feature components, DCDTW preserves the influence of major turning and manoeuvring locations while still accounting for local trajectory continuity. Conversely, for trajectories with relatively uniform sampling and simple geometric structures, the difference between DCDTW and conventional DTW may be less pronounced.

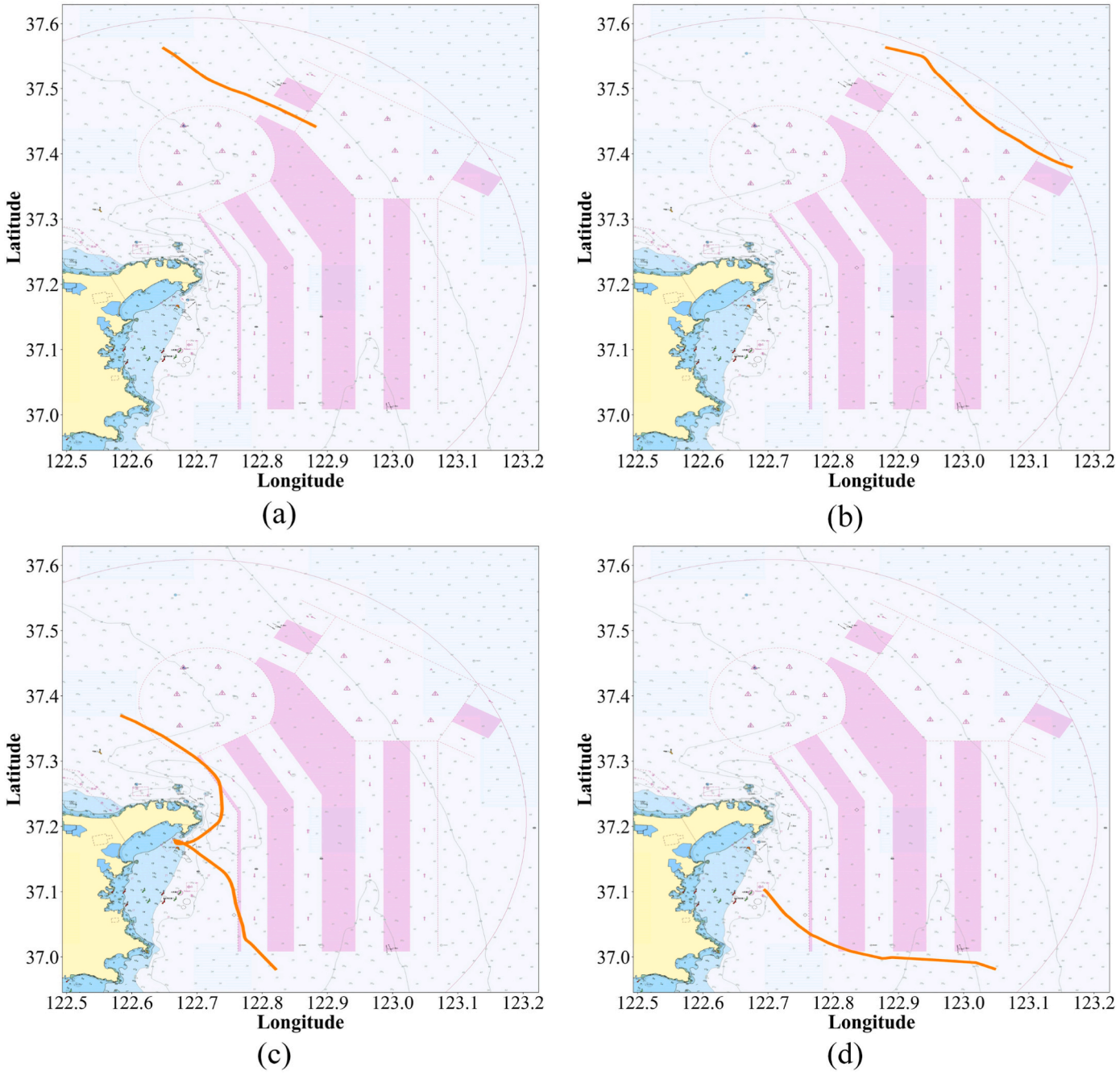

Fig. 23. Distribution characteristics of representative trajectories in the CSJP region, employed to evaluate the SCAADP algorithm's performance: (a) MMSI 210616000, (b) MMSI 229980000, (c) MMSI 312369000, and (d) MMSI 312454000.

Following the analysis of representative trajectories, the performance of the different similarity measures was further evaluated through clustering of the full datasets. The similarity matrices obtained using each method were used as input to DBSCAN, and clustering performance was evaluated using the SC, CHI, and DBI. DBSCAN was implemented in its standard form without modification. The neighborhood radius (ε) and minimum number of samples (MinPts) were set according to the parameter configuration reported by Lee et al. (2021), and the same parameter-setting procedure was applied consistently across the compared similarity measures. No additional parameter tuning was conducted for individual methods. The parameter values used in the experiments are now explicitly reported in the revised manuscript to facilitate reproducibility. The clustering results for CFDP and CSJP are presented in Tables 12 and 13, respectively.

In the CFDP area, DCDTW achieves the highest SC and CHI values and the lowest DBI, outperforming DTW, which ranks second. Compared with DTW, DCDTW improves SC and CHI by 49.358% and 46.443%, respectively, while reducing DBI by 36.783%. A consistent pattern is observed in the CSJP area, where DCDTW again attains the best performance across all three metrics, with improvements of 43.973% (SC) and 31.391% (CHI) and a reduction of 43.858% in DBI relative to the second-best method (DDDTW). These results confirm that DCDTW delivers superior clustering performance among all compared similarity measures.

Building on the representative trajectory comparisons, the overall behaviour of the similarity measures is further examined through clustering analysis using the complete datasets. The trajectory-distance matrices generated by the different methods are used as inputs to

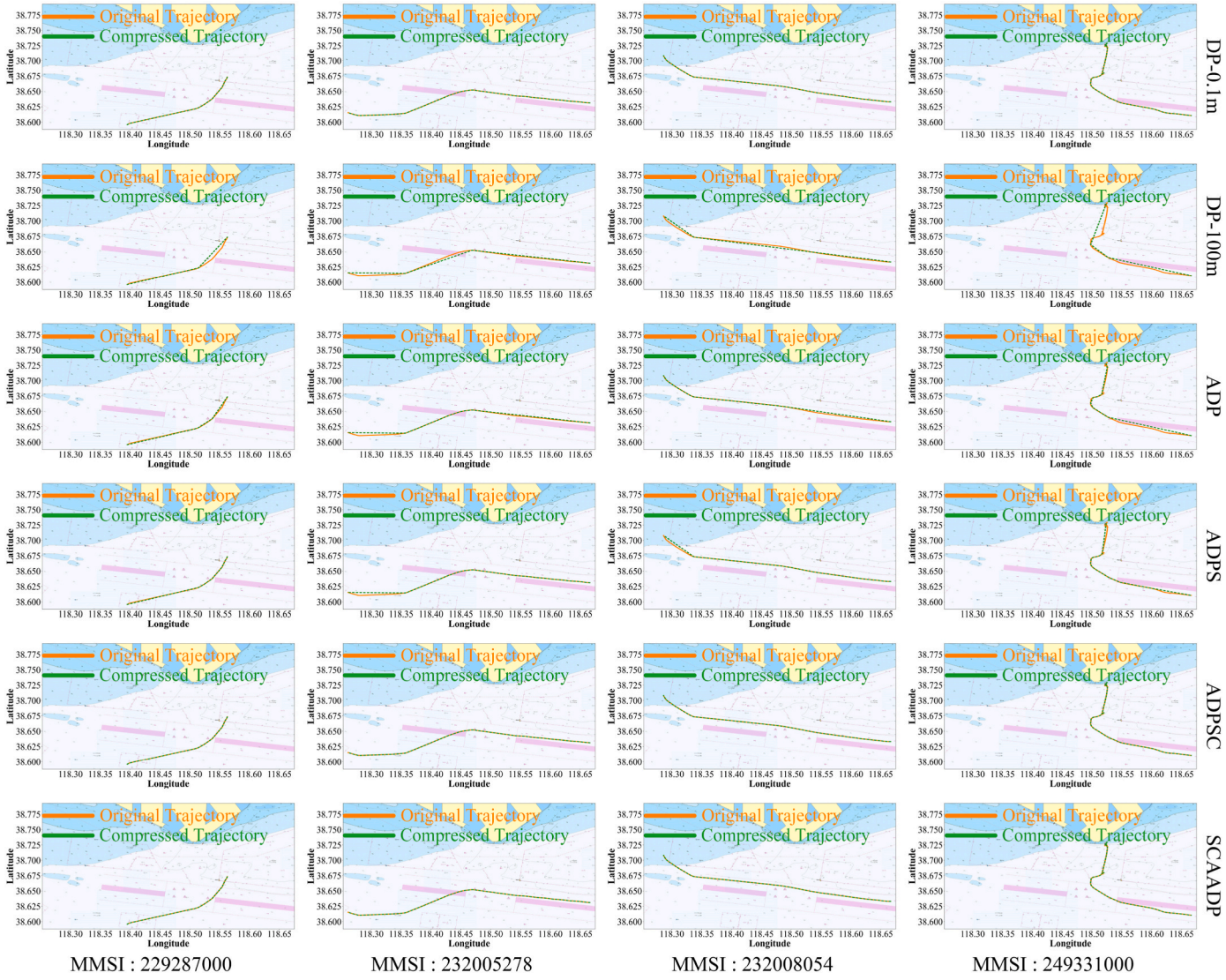

Fig. 24. Compression performance of four representative CFDP trajectories, benchmarked against original DP (0.1 m, 100 m), adaptive DP variants (ADP, ADPS, ADPSC), and the proposed SCAADP method.

DBSCAN, and the resulting cluster structures are evaluated using the SC, CHI, and DBI. The results for CFDP and CSJP are reported in Tables 12 and 13. These experiments evaluate the separability of trajectory patterns produced by the different distance measures; they should not be interpreted as direct proof that feature points are universally more important than non-feature points.

To demonstrate practical applicability, DCDTW is applied to cluster the complete trajectory datasets in the CFDP and CSJP regions using DBSCAN. The resulting clustering outcomes are visualized in Figs. 27 and 28.

In the CFDP area, trajectories are grouped into four distinct clusters corresponding to major navigation patterns. These clusters align with inbound routes to Tianjin Port, outbound routes, traffic entering or leaving the CFDP area, and trajectories travelling along straighter paths outside the main channels, as shown in Figs. 27 and 29(a).

In the CSJP area, seven trajectory clusters are identified (Fig. 28), each corresponding to a distinct shipping lane. Several cluster pairs exhibit similar spatial distributions but opposite courses over ground, reflecting bidirectional traffic along parallel routes, as illustrated in Fig. 29(b).

Overall, the clustering results are well aligned with known navigation structures, further confirming that DCDTW produces a discriminative similarity matrix suitable for large-scale vessel trajectory clustering.

The results demonstrate that the proposed DCDTW method substantially improves the reliability of trajectory clustering. By distinguishing feature and non-feature points and applying differentiated weighting, DCDTW mitigates the distortion introduced by uniform global alignment in conventional methods. This produces a more discriminative similarity matrix, which in turn yields clustering results consistent with real navigation patterns. Overall, DCDTW provides a robust similarity measurement framework for large-scale vessel trajectory analysis.

5.4.2. Efficiency acceleration with GPU parallelization

Traditional DTW and its variants incur high computational costs due to exhaustive two-dimensional euclidean distance calculations between trajectory points, which involve numerous expensive square-root operations. The proposed DCDTW method alleviates this burden by restricting such computations to a small set of feature points, while transforming similarity calculations for non-feature points into lightweight one-dimensional distance comparisons. This design substantially

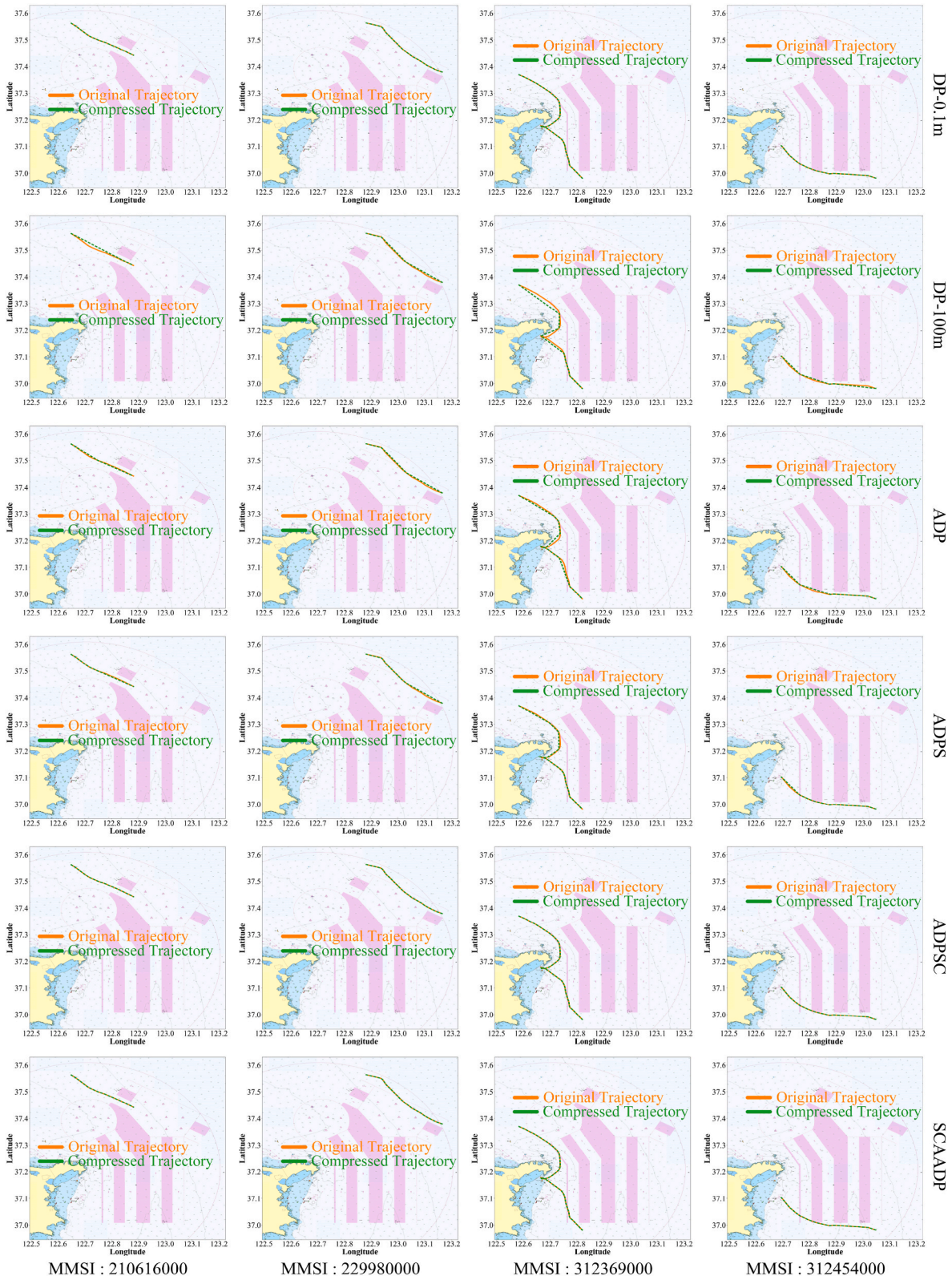

Fig. 25. Compression performance of four representative CSJP trajectories, benchmarked against original DP (0.1 m, 100 m), adaptive DP variants (ADP, ADPS, ADPSC), and the proposed SCAADP method.

Table 6
Benchmarking of trajectory compression on four representative trajectories in CFDP: proposed SCAADP versus original DP (0.1 m, 100 m) and adaptive variants (ADP, ADPS, ADPSC).

MMSI	Method	Threshold (m)	CR (%) ↑	TP-DTW ↓
229287000	DP	0.100	67.801	0.057
	ADPSC	5.909	95.026	0.918
	SCAADP	**14.097**	**96.073**	**1.046**
232005278	DP	0.100	67.010	0.064
	ADPSC	5.230	94.021	0.570
	SCAADP	**13.636**	**94.510**	**0.698**
232008054	DP	0.100	51.605	0.022
	ADPSC	4.457	95.062	0.376
	SCAADP	**10.886**	**95.276**	**0.503**
249331000	DP	0.100	24.013	0.026
	ADPSC	4.633	90.461	0.606
	SCAADP	**11.241**	**91.604**	**0.721**

Table 7
Benchmarking of trajectory compression on four representative trajectories in CSJP: proposed SCAADP versus original DP (0.1 m, 100 m) and adaptive variants (ADP, ADPS, ADPSC).

MMSI	Method	Threshold (m)	CR (%) ↑	TP-DTW ↓
210616000	DP	0.100	71.429	0.028
	ADPSC	5.194	95.819	0.224
	SCAADP	**11.187**	**95.998**	**0.407**
229980000	DP	0.100	74.034	0.075
	ADPSC	4.519	93.991	0.346
	SCAADP	**10.241**	**95.008**	**0.502**
312369000	DP	0.100	58.210	0.178
	ADPSC	4.239	94.961	2.087
	SCAADP	**10.027**	**95.002**	**2.924**
312454000	DP	0.100	59.435	0.061
	ADPSC	7.753	96.285	1.421
	SCAADP	**12.996**	**96.409**	**1.602**

Table 8
Performance evaluation of SCAADP under different dataset sizes in CFDP, showing serial CPU time, parallel GPU time, and achieved speedup.

Data Scale	Number of Trajectories	CPU Serial Computation Time (s)	GPU Parallel Computation Time (s)	Speedup Ratio
1×	630	4.126	3.881	1.063
5×	3150	20.095	4.186	4.801
10×	6300	40.973	5.093	8.045
50×	31500	220.123	6.223	35.372
100×	63000	434.851	7.219	60.237

reduces overall computational complexity.

Despite this optimization, CPU-based serial implementations of DCDTW remain time-consuming for large-scale similarity measurement. To address this limitation, DCDTW is implemented within a GPU parallel computing framework. Execution times and corresponding speedup ratios for both CPU and GPU implementations are reported in Table 14.

The results show substantial acceleration, with speedup ratios of 102.582 for CFDP and 166.019 for CSJP. Notably, although the CSJP dataset is approximately twice the size of CFDP, it achieves a 61.840% higher speedup, indicating that GPU acceleration becomes increasingly effective as the dataset size grows. These results confirm the strong scalability and computational advantage of the proposed GPU-accelerated DCDTW for large-scale trajectory similarity analysis.

5.5. Management implications and potential implementation

The proposed DCDTW framework has potential implications for maritime traffic management, navigational safety, and coastal traffic governance. By providing a feature-aware representation of vessel trajectory similarity, DCDTW can support the identification and comparison of vessel movement patterns in complex port and coastal environments. Potential users include maritime administrations, VTS operators, port authorities, and coastal traffic managers, who routinely rely on AIS-derived information to understand traffic organization and vessel behavior.

Table 9
Performance evaluation of SCAADP under different dataset sizes in CSJP, showing serial CPU time, parallel GPU time, and achieved speedup.

Data Scale	Number of Trajectories	CPU Serial Computation Time (s)	GPU Parallel Computation Time (s)	Speedup Ratio
1×	1520	9.557	7.911	1.208
5×	7600	45.605	8.698	5.243
10×	15200	93.221	9.277	10.049
50×	76000	468.301	9.902	47.294
100×	152000	926.236	10.375	89.275

From a strategic and planning perspective, DCDTW-derived trajectory distances can be combined with clustering or other pattern-analysis methods to characterise recurrent navigation routes, traffic corridors, and vessel movement patterns. Such information may help maritime and port authorities examine how vessels use available navigational space and compare observed traffic patterns with existing vessel-routing measures, traffic separation schemes, or port access arrangements. For example, persistent concentrations or changes in trajectory patterns could provide an empirical basis for further examination of traffic organisation in congested waterways. DCDTW does not determine whether a routing measure should be modified; rather, it provides trajectory-based evidence that can be considered together with safety assessments, regulatory requirements, environmental constraints, and professional judgement.

From an operational perspective, DCDTW could be incorporated into existing AIS-based monitoring or VTS platforms as a trajectory-analysis module. Incoming AIS trajectories could first undergo preprocessing and trajectory reconstruction, followed by SCAADP-based feature-point extraction and DCDTW-based similarity calculation. The resulting trajectory-distance matrix could then be used for pattern comparison, trajectory clustering, or preliminary screening of trajectories that deviate substantially from historical or representative route patterns. Relevant outputs could subsequently be presented to VTS operators or maritime managers for further assessment. In this workflow, DCDTW is intended to provide supplementary decision-support information rather than automated operational decisions.

The framework may also provide useful information for safety-oriented maritime management. Trajectories that differ substantially from established movement patterns could be flagged for further examination, which may support future applications such as abnormal-route screening, navigational risk analysis, and targeted traffic supervision. Similarly, trajectory clustering may help authorities better understand how vessels use different waterways and whether observed traffic patterns are consistent with existing routing arrangements. These potential applications are particularly relevant in congested coastal waters, port approaches, and traffic separation areas.

The GPU-based implementation provides a computational basis for applying the proposed trajectory-analysis workflow to larger AIS datasets. However, the present study evaluates trajectory comparison and clustering separability using four days of AIS data from two study areas and does not directly test management interventions or real-time operational applications. Consequently, the management uses described above should be regarded as potential implementation scenarios rather than operational capabilities demonstrated by the current experiments. Further evaluation using longer-term and geographically diverse AIS datasets, together with task-specific studies conducted in operational settings, will be required to determine the practical value of DCDTW for

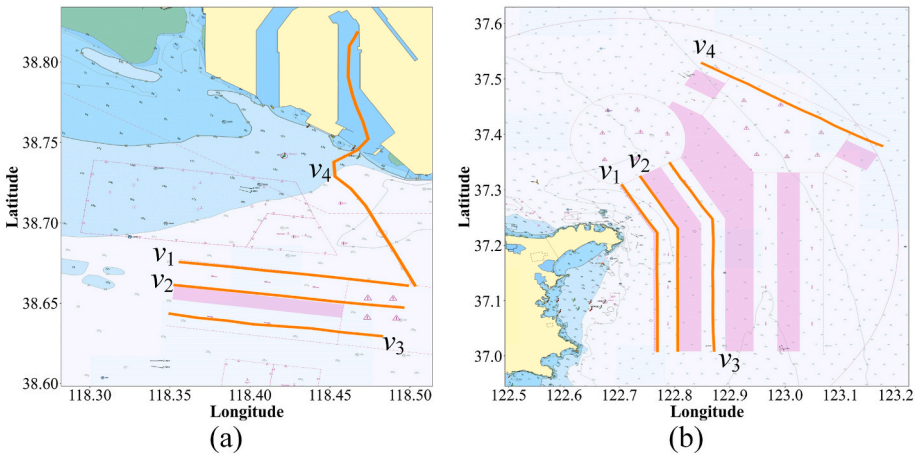

Fig. 26. Exemplary trajectories from the (a) CFDP and (b) CSJP regions, selected for similarity measure evaluation, represent the overall distribution of vessel movements in the study areas.

Table 10
Statistical Results of similarity measurements between a pair of vessel trajectories in CFDP, evaluated using various methods (i.e., DTW, DDTW, DDDTW, SW-DTW, eDTW, ACDTW, CDC-DTW, and DCDTW).

Trajectory Pair	Method	Similarity Metric	Method	Similarity Metric
v_1&v_2	DTW	7.046	eDTW	779.765
	DDTW	48.258	ACDTW	1033.791
	DDDTW	7.886	CDC-DTW	565.457
	SW-DTW	82.927	**DCDTW**	**0.227**
v_1&v_3	DTW	21.802	eDTW	1982.651
	DDTW	189.243	ACDTW	1965.008
	DDDTW	26.897	CDC-DTW	671.128
	SW-DTW	262.587	**DCDTW**	**1.829**
v_1&v_4	DTW	38.120	eDTW	4081.846
	DDTW	834.888	ACDTW	4033.375
	DDDTW	30.067	CDC-DTW	680.540
	SW-DTW	359.006	**DCDTW**	**5.029**

Table 11
Statistical Results of similarity measurements between a pair of vessel trajectories in CSJP, evaluated using various methods.

Trajectory Pair	Method	Similarity Metric	Method	Similarity Metric
v_1&v_2	DTW	43.964	eDTW	4034.296
	DDTW	12.804	ACDTW	4027.132
	DDDTW	28.470	CDC-DTW	61.633
	SW-DTW	166.154	**DCDTW**	**0.620**
v_1&v_3	DTW	186.031	eDTW	19180.536
	DDTW	3138.400	ACDTW	19110.355
	DDDTW	116.217	CDC-DTW	139.421
	SW-DTW	347.340	**DCDTW**	**4.167**
v_1&v_4	DTW	418.318	eDTW	42261.179
	DDTW	7297.490	ACDTW	42138.414
	DDDTW	231.561	CDC-DTW	1865.888
	SW-DTW	417.226	**DCDTW**	**17.944**

VTS operations, vessel-routing assessment, and coastal traffic management.

6. Conclusions

This study developed a DCDTW framework to improve vessel trajectory similarity measurement using AIS data. Unlike conventional DTW-based approaches that treat all trajectory points equally, the proposed framework explicitly differentiates between feature points and non-feature points according to their contribution to trajectory structure and navigational behavior. By adopting distinct similarity measurement strategies for these two components, DCDTW reduces the influence of densely sampled non-feature points and generates similarity measures that more accurately represent vessel movement patterns.

Table 12
Performance benchmarking of similarity measurement methods (i.e., DTW, DDTW, DDDTW, SW-DTW, eDTW, ACDTW, CDC-DTW, and DCDTW) using SC, CH, and DBI indices in CFDP waters.

Method	SC ↑	CHI ↑	DBI ↓
DTW	0.681	1077.932	0.802
DDTW	0.150	462.253	2.994
DDDTW	0.485	828.104	1.233
SW-DTW	0.307	693.521	1.541
eDTW	0.052	238.834	4.090
ACDTW	0.049	191.778	4.744
CDC-DTW	0.071	309.053	3.565
DCDTW	**0.999**	**1578.552**	**0.507**

Table 13
Performance benchmarking of similarity measurement methods using SC, CH, and DBI indices in CSJP waters.

Method	SC ↑	CHI ↑	DBI ↓
DTW	0.412	790.287	1.406
DDTW	0.052	273.773	4.028
DDDTW	0.581	996.624	1.099
SW-DTW	0.214	600.596	2.507
eDTW	0.006	100.058	6.621
ACDTW	0.020	132.749	5.928
CDC-DTW	0.101	399.747	3.497
DCDTW	**0.861**	**1309.478**	**0.617**

To support the proposed framework, an adaptive trajectory compression algorithm, SCAADP, was developed for feature-point identification. By jointly incorporating variations in SOG and COG, SCAADP automatically determines compression thresholds and extracts representative trajectory features without requiring manual parameter calibration. Both SCAADP and DCDTW were further implemented within a GPU-based parallel computing framework to improve computational efficiency and facilitate the processing of large-scale AIS datasets.

Experiments conducted using AIS data from Caofeidian Port and the Chengshan Jiao Promontory demonstrated that the proposed approach consistently outperforms conventional DTW-based methods in trajectory similarity measurement. SCAADP achieved high compression efficiency while preserving essential trajectory structures, and DCDTW produced more reliable and physically meaningful similarity

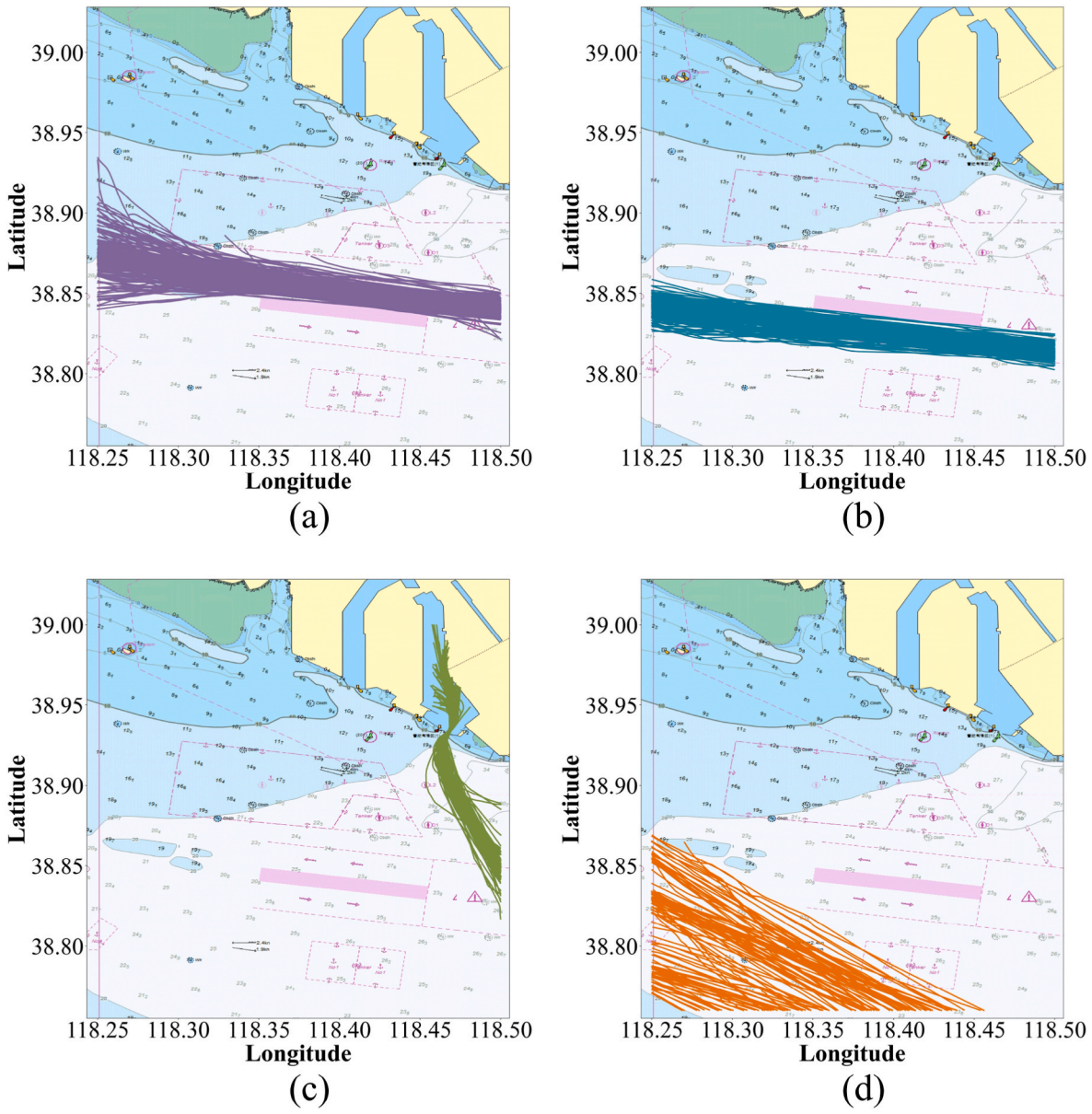

Fig. 27. Categorization of vessel trajectories in CFDP waters, with clusters derived from the DCDTW similarity measurement.

assessments across different traffic scenarios. The GPU-accelerated implementation substantially reduced computational time, confirming the scalability of the framework for large-scale maritime applications.

Beyond methodological improvements, the findings highlight the importance of trajectory similarity measurement as a foundation for maritime traffic analysis and navigational safety assessment. Accurate identification of similar vessel movements may support trajectory clustering, vessel behavior characterization, anomaly detection, and the identification of dominant traffic patterns. These capabilities are increasingly important for maritime authorities, port operators, and Vessel Traffic Service (VTS) organizations seeking to enhance situational awareness and manage growing traffic complexity in coastal waters and port approaches. By providing more reliable representations of vessel movement behavior, the proposed framework can contribute to evidence-based traffic management, operational decision support, and safety-oriented maritime governance.

The proposed framework also provides a foundation for emerging data-driven maritime applications, including intelligent traffic monitoring, risk assessment, and autonomous navigation support. Reliable trajectory similarity information can improve the understanding of vessel route preferences and behavioral patterns, thereby supporting the development of predictive and proactive maritime safety management strategies.

Several limitations should be acknowledged. First, the evaluation is based on four consecutive days of AIS data from two maritime regions in China. Although the study areas represent different navigational conditions, broader validation across seasons, vessel types, traffic environments, and geographic regions is required before the generalizability of DCDTW can be established. Second, the present experiments focus primarily on trajectory-pattern separability through clustering and do not directly validate DCDTW in downstream operational applications. Third, the framework remains dependent on AIS data quality, and the effects of missing observations, irregular sampling, and measurement errors on trajectory similarity have not yet been systematically examined.

Future work will therefore evaluate DCDTW using longer-term and geographically diverse AIS datasets and investigate its robustness to different sampling characteristics and data-quality conditions. Vessel-

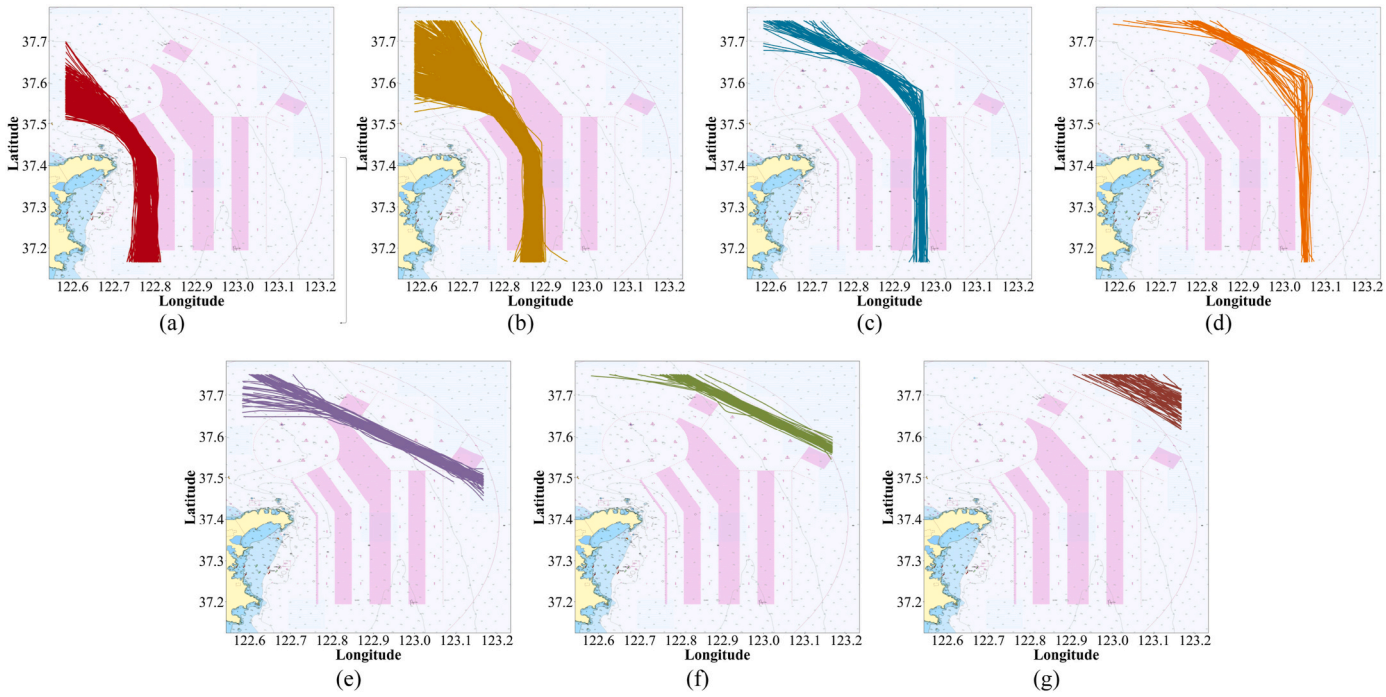

Fig. 28. Categorization of vessel trajectories in CSJP waters, with clusters derived from the DCDTW similarity measurement.

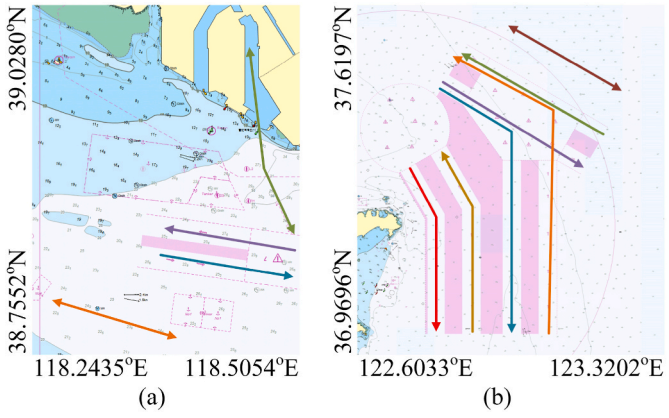

Fig. 29. Schematic illustration of channel usage, with colors representing distinct vessel movement patterns: (a) CFDP, (b) CSJP.

Table 14
Benchmarking results of the GPU-accelerated DCDTW method: serial CPU execution time, parallel GPU execution time, and achieved speedup.

Research Waters	CPU Serial Computation Time (s)	GPU Parallel Computation Time (s)	Speedup Ratio
CFDP	50941.959	496.597	102.582
CSJP	713272.483	4296.330	166.019

and task-aware extensions will also be explored to account for differences in manoeuvring behaviour and operational constraints. Further task-specific validation will examine the applicability of DCDTW to maritime anomaly detection, traffic-risk analysis, and VTS-assisted decision support.

CRediT authorship contribution statement

Yan Li: Writing – original draft, Methodology, Investigation, Data curation, Conceptualization, Resources, Software, Validation, Visualization. **Lan Song:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization, Project administration, Visualization, Validation. **Bi Yu Chen:** Visualization, Investigation, Formal analysis, Supervision, Resources. **William Luocheng Wu:** Validation, Writing – review & editing, Visualization, Investigation. **Zaili Yang:** Writing – review & editing, Conceptualization, Funding acquisition, Validation, Visualization. **Huanhuan Li:** Writing – review & editing, Supervision, Investigation, Writing – original draft, Project administration, Methodology, Validation, Conceptualization, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 864724), and the National Natural Science Foundation of China (NSFC) under Grant Nos. 42407114 and 52571407.

Data availability

Data will be made available on request.

References

Al-Naymat, G., Chawla, S., Taheri, J., 2012. SparseDTW: a novel approach to speed up dynamic time warping. https://doi.org/10.48550/arXiv.1201.2969.
Ashari, I.F., Nugroho, E.D., Baraku, R., Yanda, I.N., Liwardana, R., 2023. Analysis of elbow, silhouette, davies-bouldin, calinski-harabasz, and rand-index evaluation on K-Means algorithm for classifying flood-affected areas in jakarta. J. Appl. Inform. Comput. 7, 95–103. https://doi.org/10.30871/jaic.v7i1.4947.
Braverman, V., Cohen-Addad, V., Jiang, H.-C.S., Krauthgamer, R., Schwiegelshohn, C., Toftrup, M.B., Wu, X., 2022. The power of uniform sampling for coresets. In: 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS).

Presented at the 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 462–473. https://doi.org/10.1109/FOCS54457.2022.00051.
Chang, Y., Qi, J., Liang, Y., Tanin, E., 2023. Contrastive trajectory similarity learning with dual-feature attention. https://doi.org/10.48550/arXiv.2210.05155.
Chang, Y., Tanin, E., Cong, G., Jensen, C.S., Qi, J., 2024. Trajectory similarity measurement: an efficiency perspective. Proc. VLDB Endow. 17, 2293–2306. https://doi.org/10.14778/3665844.3665858.
Chen, W., Chen, R., Chen, Y., Kuusniemi, H., Wang, J., 2010. An effective pedestrian dead reckoning algorithm using a unified heading error model. In: IEEE/ION Position, Location and Navigation Symposium. Presented at the IEEE/ION Position, Location and Navigation Symposium, pp. 340–347. https://doi.org/10.1109/PLANS.2010.5507300.
Chowdhury, R., Navsalkar, A., Subramani, D., 2022. GPU-accelerated multi-objective optimal planning in stochastic dynamic environments. J. Mar. Sci. Eng. 10, 533. https://doi.org/10.3390/jmse10040533.
Cignoni, P., Montani, C., Scopigno, R., 1998. DeWall: a fast divide and conquer delaunay triangulation algorithm. Comput. Aided Des. 30, 333–341. https://doi.org/10.1016/S0010-4485(97)00082-1.
Dally, W.J., Keckler, S.W., Kirk, D.B., 2021. Evolution of the graphics processing unit (GPU). IEEE Micro 41, 42–51. https://doi.org/10.1109/MM.2021.3113475.
Folgado, D., Barandas, M., Matias, R., Martins, R., Carvalho, M., Gamboa, H., 2018. Time alignment measurement for time series. Pattern Recogn. 81, 268–279. https://doi.org/10.1016/j.patcog.2018.04.003.
Fu, T., Yang, Z., Ye, Z., Ma, C., Han, Y., Luo, Y., Wang, X., Wang, Z., 2025. A survey on the scheduling of DL and LLM training jobs in GPU clusters. Chin. J. Electron. 34, 881–905. https://doi.org/10.23919/cje.2024.00.070.
Fuentes, G., 2021. Generating bunkering statistics from AIS data: a machine learning approach. Transport. Res. E Logist. Transport. Rev. 155, 102495. https://doi.org/10.1016/j.tre.2021.102495.
Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., Volkov, V., 2008. Parallel computing experiences with CUDA. IEEE Micro 28, 13–27. https://doi.org/10.1109/MM.2008.57.
Gilman, G., Ogden, S.S., Guo, T., Walls, R.J., 2021. Demystifying the placement policies of the NVIDIA GPU thread block scheduler for concurrent kernels. SIGMETRICS Perform. Eval. Rev. 48, 81–88. https://doi.org/10.1145/3453953.3453972.
Górecki, T., Łuczak, M., 2013. Using derivatives in time series classification. Data Min. Knowl. Discov. 26, 310–331. https://doi.org/10.1007/s10618-012-0251-4.
Han, W., Deng, Z., Chu, J., Zhu, J., Gao, P., Shah, T., 2018. A parallel online trajectory compression approach for supporting big data workflow. Computing 100, 3–20. https://doi.org/10.1007/s00607-017-0563-8.
Hu, D., Chen, L., Fang, H., Fang, Z., Li, T., Gao, Y., 2023. Spatio-temporal trajectory similarity measures: a comprehensive survey and quantitative study [WWW Document]. arXiv.org. https://arxiv.org/abs/2303.05012v2. (Accessed 25 September 2026).
Huang, Y., Li, Y., Zhang, Z., Liu, R.W., 2020. GPU-accelerated compression and visualization of large-scale vessel trajectories in maritime IoT industries. IEEE Internet Things J. 7, 10794–10812. https://doi.org/10.1109/JIOT.2020.2989398.
Jiang, Y., Qi, Y., Wang, W.K., Bent, B., Avram, R., Olgin, J., Dunn, J., 2020. EventDTW: an improved dynamic time warping algorithm for aligning biomedical signals of nonuniform sampling frequencies. Sensors 20, 2700. https://doi.org/10.3390/s20092700.
Klein, I., Diamant, R., 2020. Dead reckoning for trajectory estimation of underwater drifters under water currents. J. Mar. Sci. Eng. 8, 205. https://doi.org/10.3390/jmse8030205.
Kumar, S., Sharma, A., Kumar, G., 2025. Data-driven predictive model for dynamic expected travel time estimation in rail freight networks: a case study. Transport. Res. E Logist. Transport. Rev. 200, 104201. https://doi.org/10.1016/j.tre.2025.104201.
Lee, C.K.H., Leung, E.K.H., 2023. Spatiotemporal analysis of bike-share demand using DTW-based clustering and predictive analytics. Transport. Res. E Logist. Transport. Rev. 180, 103361. https://doi.org/10.1016/j.tre.2023.103361.
Lee, H.-T., Lee, J.-S., Yang, H., Cho, I.-S., 2021. An AIS data-driven approach to analyze the pattern of ship trajectories in ports using the DBSCAN algorithm. Appl. Sci. 11, 799. https://doi.org/10.3390/app11020799.
Li, H., Jiao, H., Yang, Z., 2023. AIS data-driven ship trajectory prediction modelling and analysis based on machine learning and deep learning methods. Transport. Res. E Logist. Transport. Rev. 175, 103152. https://doi.org/10.1016/j.tre.2023.103152.
Li, H., Lam, J.S.L., Yang, Z., Liu, J., Liu, R.W., Liang, M., Li, Y., 2022. Unsupervised hierarchical methodology of maritime traffic pattern extraction for knowledge discovery. Transport. Res. C Emerg. Technol. 143, 103856. https://doi.org/10.1016/j.trc.2022.103856.
Li, H., Liu, J., Yang, Z., Liu, R.W., Wu, K., Wan, Y., 2020. Adaptively constrained dynamic time warping for time series classification and clustering. Inf. Sci. 534, 97–116. https://doi.org/10.1016/j.ins.2020.04.009.
Li, S., Chen, W., Yan, B., Li, Z., Zhu, S., Yu, Y., 2023. Self-supervised contrastive representation learning for large-scale trajectories. Future Gener. Comput. Syst. 148, 357–366. https://doi.org/10.1016/j.future.2023.05.033.
Li, W., He, R., Liang, B., Yang, F., Han, S., 2023. Similarity measure of time series with different sampling frequencies based on context density consistency and dynamic time warping. IEEE Signal Process. Lett. 30, 1417–1421. https://doi.org/10.1109/LSP.2023.3316010.
Li, Y., Li, H., Zhang, C., Zhao, Y., Yang, Z., 2024. Incorporation of adaptive compression into a GPU parallel computing framework for analyzing large-scale vessel trajectories. Transport. Res. C Emerg. Technol. 163, 104648. https://doi.org/10.1016/j.trc.2024.104648.
Liu, D., Li, L., Li, J., 2024. A trajectory similarity computation method based on GAT-based transformer and CNN model. Sci. Rep. 14, 16173. https://doi.org/10.1038/s41598-024-67256-7.
Liu, J., Li, H., Yang, Z., Wu, K., Liu, Y., Liu, R.W., 2019. Adaptive douglas-peucker algorithm with automatic thresholding for AIS-based vessel trajectory compression. IEEE Access 7, 150677–150692. https://doi.org/10.1109/ACCESS.2019.2947111.
Lyu, C., Wu, X., Liu, Y., Liu, Z., 2022. A partial-fréchet-distance-based framework for bus route identification. IEEE Trans. Intell. Transport. Syst. 23, 9275–9280. https://doi.org/10.1109/TITS.2021.3069630.
Maceina, T.J., Manduchi, G., 2017. Assessment of general purpose GPU systems in real-time control. IEEE Trans. Nucl. Sci. 64, 1455–1460. https://doi.org/10.1109/TNS.2017.2691061.
Meng, Q., Yu, X., Yao, C., Li, X., Li, P., Zhao, X., 2017. Improvement of OPW-TR algorithm for compressing GPS trajectory data. J. Inf. Process. Syst. 13, 533. https://doi.org/10.3745/JIPS.03.0073.
Meratnia, N., de By, R.A., 2004. Spatiotemporal compression techniques for moving point objects. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., Ferrari, E. (Eds.), Advances in Database Technology - EDBT 2004. Springer, Berlin, Heidelberg, pp. 765–782. https://doi.org/10.1007/978-3-540-24741-8_44.
Muckell, J., Hwang, J.-H., Patil, V., Lawson, C.T., Ping, F., Ravi, S.S., 2011a. SQUISH: an online approach for GPS trajectory compression. In: Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications, Com.Geo '11. Association for Computing Machinery, New York, NY, USA, pp. 1–8. https://doi.org/10.1145/1999320.1999333.
Muckell, J., Hwang, J.-H., Patil, V., Lawson, C.T., Ping, F., Ravi, S.S., 2011b. SQUISH: an online approach for GPS trajectory compression. In: Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications, COM.Geo '11. Association for Computing Machinery, New York, NY, USA, pp. 1–8. https://doi.org/10.1145/1999320.1999333.
Muckell, J., Olsen, P.W., Hwang, J.-H., Lawson, C.T., Ravi, S.S., 2014. Compression of trajectory data: a comprehensive evaluation and new approach. GeoInformatica 18, 435–460. https://doi.org/10.1007/s10707-013-0184-0.
Muscillo, R., Conforto, S., Schmid, M., Caselli, P., D'Alessio, T., 2007. Classification of motor activities through derivative dynamic time warping applied on accelerometer data. In: 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Presented at the 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 4930–4933. https://doi.org/10.1109/IEMBS.2007.4353446.
Pallotta, G., Vespe, M., Bryan, K., 2013. Vessel pattern knowledge discovery from AIS data: a framework for anomaly detection and route prediction. Entropy 15, 2218–2245. https://doi.org/10.3390/e15062218.
Peng, P., Xie, X., Claramunt, C., Lu, F., Gong, F., Yan, R., 2025. Bibliometric analysis of maritime cybersecurity: research status, focus, and perspectives. Transport. Res. E Logist. Transport. Rev. 195, 103971. https://doi.org/10.1016/j.tre.2025.103971.
Potamias, M., Patroumpas, K., Sellis, T., 2006. Sampling trajectory streams with spatiotemporal criteria. In: 18th International Conference on Scientific and Statistical Database Management (SSDBM'06). Presented at the 18th International Conference on Scientific and Statistical Database Management (SSDBM'06), pp. 275–284. https://doi.org/10.1109/SSDBM.2006.45.
Ribeiro, C.V., Paes, A., Oliveira, D. de, 2023. AIS-Based maritime anomaly traffic detection: a review. Expert Syst. Appl. 231, 120561. https://doi.org/10.1016/j.eswa.2023.120561.
Rong, H., Teixeira, A.P., Guedes Soares, C., 2020. Data mining approach to shipping route characterization and anomaly detection based on AIS data. Ocean Eng. 198, 106936. https://doi.org/10.1016/j.oceaneng.2020.106936.
Rosen, P., 2013. A visual approach to investigating shared and global memory behavior of CUDA kernels. Comput. Graph. Forum 32, 161–170. https://doi.org/10.1111/cgf.12103.
Salvador, S., Chan, P., 2007. Toward accurate dynamic time warping in linear time and space. Intell. Data Anal.
Shi, H., Fan, Z., Wang, X., Tian, C., Zhen, L., 2026. CPU-GPU solution for bus-based freight scheduling under uncertainty. Transport. Res. E Logist. Transport. Rev. 205, 104508. https://doi.org/10.1016/j.tre.2025.104508.
Sousa, R.S.D., Boukerche, A., Loureiro, A.A.F., 2020. Vehicle trajectory similarity: models, methods, and applications. ACM Comput. Surv. 53, 94:1–94:32. https://doi.org/10.1145/3406096.
Taha, A.A., Hanbury, A., 2015. An efficient algorithm for calculating the exact hausdorff distance. IEEE Trans. Pattern Anal. Mach. Intell. 37, 2153–2163. https://doi.org/10.1109/TPAMI.2015.2408351.
Tao, Z., Xu, Q., Liu, X., Liu, J., 2023. An integrated approach implementing sliding window and DTW distance for time series forecasting tasks. Appl. Intell. 53, 20614–20625. https://doi.org/10.1007/s10489-023-04590-9.
Tu, E., Zhang, G., Rachmawati, L., Rajabally, E., Huang, G.-B., 2018. Exploiting AIS data for intelligent maritime navigation: a comprehensive survey from data to methodology. IEEE Trans. Intell. Transport. Syst. 19, 1559–1582. https://doi.org/10.1109/TITS.2017.2724551.
Visvalingam, M., Whyatt, J.D., 1990. The douglas-peucker algorithm for line simplification: re-evaluation through visualization. Comput. Graph. Forum 9, 213–225. https://doi.org/10.1111/j.1467-8659.1990.tb00398.x.
Wu, R., Keogh, E.J., 2022. FastDTW is approximate and generally slower than the algorithm it approximates. IEEE Trans. Knowl. Data Eng. 34, 3779–3785. https://doi.org/10.1109/TKDE.2020.3033752.
Xing, W., Wang, J., Zhou, K., Li, H., Li, Y., Yang, Z., 2023. A hierarchical methodology for vessel traffic flow prediction using Bayesian tensor decomposition and similarity

grouping. Ocean Eng. 286, 115687. https://doi.org/10.1016/j.oceaneng.2023.115687.
Xue, H., Wang, S., Xia, M., Guo, S., 2024. G-Trans: a hierarchical approach to vessel trajectory prediction with GRU-based transformer. Ocean Eng. 300, 117431. https://doi.org/10.1016/j.oceaneng.2024.117431.
Yang, X., Tsoulakos, N., Xiao, Z., Wei, X., Fu, X., Yan, R., 2025. Estimation of shipping emissions from maritime big data: a comprehensive review and prospective. Transport. Res. E Logist. Transport. Rev. 202, 104313. https://doi.org/10.1016/j.tre.2025.104313.
Zhang, M., Hibi, K., Inoue, J., 2023. GPU-accelerated artificial neural network potential for molecular dynamics simulation. Comput. Phys. Commun. 285, 108655. https://doi.org/10.1016/j.cpc.2022.108655.
Zhang, T., Wang, Z., Wang, P., 2024. A method for compressing AIS trajectory based on the adaptive core threshold difference douglas–peucker algorithm. Sci. Rep. 14, 21408. https://doi.org/10.1038/s41598-024-71779-4.
Zhang, X., Zhou, S., Jiang, Z., Chu, X., Yu, C., 2025. A novel ship trajectory clustering model integrated with spatiotemporal attributes. Ocean Eng. 336, 121774. https://doi.org/10.1016/j.oceaneng.2025.121774.
Zhao, L., Shi, G., 2018. A method for simplifying ship trajectory based on improved douglas–peucker algorithm. Ocean Eng. 166, 37–46. https://doi.org/10.1016/j.oceaneng.2018.08.005.
Zhou, K., Adhianto, L., Anderson, J., Cherian, A., Grubisic, D., Krentel, M., Liu, Y., Meng, X., Mellor-Crummey, J., 2021. Measurement and analysis of GPU-Accelerated applications with HPCToolkit. Parallel Comput. 108, 102837. https://doi.org/10.1016/j.parco.2021.102837.
Zhou, S., Li, J., Wang, H., Shang, S., Han, P., 2023. GRLSTM: trajectory similarity computation with graph-based residual LSTM. Proc. AAAI Conf. Artif. Intell. 37, 4972–4980. https://doi.org/10.1609/aaai.v37i4.25624.
Zhou, Z., Zhang, Y., Yuan, X., Wang, H., 2023. Compressing AIS trajectory data based on the multi-objective peak douglas–peucker algorithm. IEEE Access 11, 6802–6821. https://doi.org/10.1109/ACCESS.2023.3234121.
Zhu, F., Ma, Z., 2021. Ship trajectory online compression algorithm considering handling patterns. IEEE Access 9, 70182–70191. https://doi.org/10.1109/ACCESS.2021.3078642.
Zhu, J., Niu, X., She, K., Fournier-Viger, P., Li, J., 2026. GraphTraj: Structure-aware representation learning for trajectory similarity calculation. Knowl. Base Syst. 349, 116395. https://doi.org/10.1016/j.knosys.2026.116395.