algorithms

MDPI

Article

Data-Driven Graph-Based Methods for In-Port AIS Vessel Trajectory Reconstruction

Evangelia Zaou [1,†], Neofytos Dimitriou [1,*,†] and Ognjen Arandjelović [2]

[1] Cyprus Marine and Maritime Institute, Vasileos Pavlou Square 13, Larnaca 6023, Cyprus
[2] School of Computer Science, University of St Andrews, St Andrews KY16 9SX, UK
* Correspondence: neofytosd@gmail.com
† These authors contributed equally to this work.

Abstract

AIS trajectories in and around ports are often incomplete because of transmission interruptions, reception failures, and infrastructure outages. In this paper, we investigate whether the Data-driven AIS Trajectory Interpolation method (DAISTIN), which reconstructs missing trajectory segments using a graph derived from historical AIS observations, can be improved by retaining local directional information that is otherwise discarded when observations are sampled to form the graph. To this end, we introduce two extensions, xDAISTIN and xDAISTOUT. The originality of the proposed approach lies in extending DAISTIN by enriching each sampled graph node with neighbourhood-level directional information from nearby historical AIS observations. Specifically, local vessel headings are encoded cyclically and modelled using kernel density estimators, allowing candidate graph transitions to be evaluated using local movement probabilities rather than only the heading of the sampled point. The methods are evaluated on real-world AIS data from the Port of Antwerp by introducing synthetic gaps into held-out trajectories and measuring how well they are reconstructed by the methods. Using 36,748 moving sub-trajectories from 5488 vessel trips, we compare the proposed methods against DAISTIN, linear interpolation, and several directed, undirected, probabilistic, and shortest-path graph variants. Across graph sizes from 100 k to 180 k nodes, DAISTIN and its undirected variant reconstructed 48.48–84.61% and 72.35–91.23% of missing segments, respectively, compared to 88.02–96.42% and 91.36–99.13% for xDAISTIN and its variants, and 65.75–85.17% and 72.12–97.80% for xDAISTOUT and its variants. Compared with undirected DAISTIN at 180 k nodes, undirected xDAISTIN at 180 k nodes achieved significantly lower mean SPD, DFD, and HD (all Holm-adjusted, $p < 0.001$), by approximately 14%, 43%, and 36%, respectively. These findings demonstrate that preserving neighbourhood-level directional behaviour during graph construction can substantially improve the interpolation completion rate and reconstruction accuracy in complex port environments.

Keywords: automatic identification system; interpolation; geometric sampling; kernel density estimator; Antwerp; DAISTIN

Check for updates

Academic Editor: Mauro Mezzini

Received: 19 May 2026
Revised: 4 August 2026
Accepted: 6 August 2026
Published: 18 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

According to the automatic identification system (AIS) standards [1], AIS transponders autonomously broadcast messages at intervals ranging from 2 s to 3 min, with the update rate determined by the AIS class type as well as the vessel's activity and speed. In practice, AIS time-series data often contain gaps, either because vessels stop transmitting or because

transmitted messages are not successfully received. Reception can be degraded by very high-frequency propagation conditions and infrastructure limitations; for example, during bad weather, near tall structures, when antennas are mounted low, or in offshore areas with limited coverage [2]. Separately, receiver or network downtime can interrupt data collection and introduce additional gaps [3]. Finally, vessels may intentionally deactivate their AIS, creating discontinuities that are unrelated to reception conditions [3]. These data gaps make maritime traffic modelling more difficult and can bias statistical analyses. Their impact is particularly pronounced in port areas, where complex geography and navigational constraints shape vessel motion and complicate trajectory reconstruction, while high traffic density can also make gaps frequent due to increased communication load. Compared to open-sea movements, port trajectories are often more variable and frequently non-linear, with more frequent changes in speed and heading. Figure 1 illustrates the corresponding ambiguity in reconstructing a trajectory gap. For each example of a real AIS gap from the Port of Antwerp we show several trajectories with similar start and end points and comparable speeds; the variety and complexity of their geometries make denser representation a necessity.

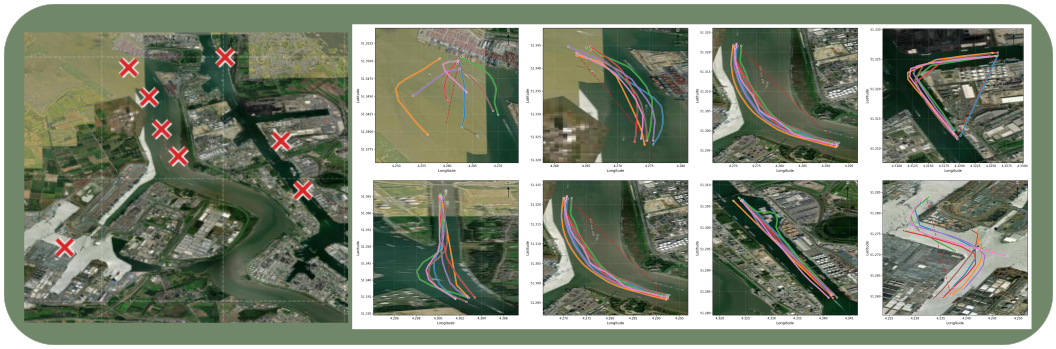


Figure 1. Examples of trajectories for selected real-world gaps in the Port of Antwerp. The left panel marks the locations of eight selected real AIS gaps (red *x* markers). The right panels visualize, for each gap case, trajectories with similar start and end locations and comparable speed in different colours; the dashed red segments connect the two points of a gap.

Many existing trajectory reconstruction methods are evaluated in open-sea settings, often on port-to-port or port-approaching segments. However, their effectiveness for in-port trajectory reconstruction remains less explored. In this work, we empirically examine this setting using linear interpolation, DAISTIN [4], and an undirected variant, and show that performance can degrade substantially in ports. Motivated by these observations, we introduce xDAISTIN and xDAISTOUT, two graph-based extensions of DAISTIN, together with multiple variants. Our experiments on real-world AIS data from the Port of Antwerp show that enriching graph nodes with local directional information substantially improves the interpolation completion rate relative to the original DAISTIN method, while the strongest xDAISTIN and xDAISTOUT configurations reduce reconstruction error relative to both DAISTIN and linear interpolation. Comparisons of directed, undirected, probabilistic, and shortest-path variants provide further insight into the contributions of different graph construction and traversal approaches.

Both DAISTIN and our proposed methods, xDAISTIN and xDAISTOUT, represent vessel motion using a graph in which nodes correspond to historical AIS positions and edges encode feasible transitions consistent with the historical direction of travel. Since constructing such a graph over all historical AIS data is computationally impractical, the node set is reduced via stratified geometric sampling. A limitation of DAISTIN is that sampling retains information only from the selected AIS data points, so directional information contained in nearby non-sampled data points is not incorporated into the

resulting nodes. In dense and navigationally constrained traffic settings—where different vessels can follow distinct manoeuvres within a small area—we hypothesize that this loss of local directional variability can negatively affect interpolation. To address this, during sampling, our proposed methods enrich each retained node with heading information from its local neighbourhood. Concretely, for each sampled node we fit a kernel density estimator (KDE) over the headings of nearby AIS data points, yielding a local probabilistic description of directional behaviour at node level. As we show empirically, using this neighbourhood-level directional information during edge construction improves reconstruction accuracy in the port setting. At a high level, xDAISTIN and xDAISTOUT share the same node-enrichment principle, but they differ in how the local heading distributions are used when evaluating candidate transitions during interpolation.

Our contributions can be summarized as follows:

- We introduce xDAISTIN and xDAISTOUT, two graph-based extensions of DAISTIN for in-port AIS trajectory reconstruction that enrich sampled graph nodes with neighbourhood-level directional information derived from nearby AIS data points.
- We systematically evaluate multiple configurations of DAISTIN, xDAISTIN, and xDAISTOUT, including probabilistic versus shortest-path inference and directed versus undirected traversal, to isolate the impact of these design choices on the interpolation completion rate and reconstruction performance.
- Using real-world AIS data from the Port of Antwerp, we conduct a comparative analysis between linear interpolation, the proposed methods xDAISTIN and xDAISTOUT, and the original DAISTIN method, including multiple variants of each, and show clear improvements in interpolation completion rate and reconstruction performance from the proposed methods across the evaluated settings.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the materials and methods, including data set information, preprocessing steps, DAISTIN, the proposed xDAISTIN and xDAISTOUT methods, and their variants. Section 4 reports the experimental setup and results. Section 5 discusses the main findings and limitations, and Section 6 concludes the paper.

2. Related Work

Over recent years, various numerical and data-driven methods have been proposed for reconstructing incomplete vessel trajectories from AIS time-series data. Examples of numerical methods include linear interpolation, splines, or combinations of the two [5]. Linear interpolation assumes that movement between two known points is linear, thereby generating intermediate points along a straight path; it is typically most suitable when gaps are short and vessel motion between observations is relatively steady, but it can fall short when trajectories are complex or gaps are prolonged. In contrast, spline interpolation relaxes the linearity assumption by using polynomials of a specified order, allowing the interpolated path to follow a curved trajectory. Nguyen et al. [5] introduced a method that combines both linear and spline interpolation by first assessing whether the vessel's motion is closer to linear or turning based on curvature estimates of trajectory segments adjacent to a signal gap. Depending on the outcome, the method applies linear interpolation for steady courses and cubic Hermite interpolation for turning manoeuvres, and can interpolate multiple trajectory parameters (e.g., position, course over ground, speed over ground, and heading) in settings where such local assumptions hold [5]. However, these assumptions may be less reliable in port environments, where vessel motion can change rapidly while following strong navigational constraints. Sang et al. [6] proposed a method aimed at complex environments such as inland ports, assuming (i) straight-line motion on steady course segments and (ii) circular-arc motion during turns. While effective when these

assumptions are met, performance may degrade when gaps increase in duration and when manoeuvres deviate from a single arc.

Deep learning methods have also been explored for reconstructing incomplete AIS trajectories. For example, Yuan et al. [7] employed Long Short-Term Memory (LSTM) networks to infer missing trajectory segments from historical sequences. However, as noted by Magnussen et al. [4], sequence models such as LSTMs can in some cases yield implausible reconstructions—e.g., producing paths that drift away from realistic vessel motion—particularly when training data are limited or when the motion patterns within a gap are not well constrained by surrounding observations. More recent works have extended this line of research through bidirectional recurrent, sequence-to-sequence, and attention-based architectures. Bai et al. [8] proposed a reconstruction model combining Dempster–Shafer evidence theory with a bidirectional Long Short-Term Memory (BiLSTM) network for missing AIS observations in geometrically constrained waterways. Ye et al. [9] combined an initial cubic-spline restoration with feature-correlation analysis and a Seq2Seq neural network, using the latter to refine the provisionally restored trajectory. More recently, Galdelli et al. [10] introduced a BackForward BiLSTM architecture that processes observations before and after a signal gap through separate forward and backward models. Their evaluation included conventional LSTM, CNN–LSTM, BiLSTM, GRU, BRITS, NRTSI, and Seq2Seq baselines. Gao [11] similarly combined temporal convolutions, self-attention, and BiLSTM layers for the restoration of missing and abnormal AIS trajectory observations. Transformer architectures have also more recently been considered; for example, Han et al. [12] proposed STAR, which employs a Transformer encoder and an integrated representation of temporal, spatial, directional, and velocity information to recover sparsely sampled marine trajectories. The ancestry of our ideas stretches yet further back. Related techniques have also been used with success in, among other areas, computer vision work on scene-specific motion modelling. Arandjelović, for example, represented people's motion trajectories in crowded public spaces using local tracklet primitives, defined by position and direction, and learned normal motion patterns from the resulting sequences of primitives [13]. Although that work focused on novelty detection rather than the reconstruction of broken trajectories, it is relevant here because it illustrates the general usefulness of decomposing trajectories into local, direction-aware motion elements and using their learned contextual regularities as a prior over possible movement.

Graph-based methods have emerged as a competing alternative, as their topology is typically the result of modelling maritime traffic dynamics and structure based on historical data, which is a strong prior that avoids the construction of implausible trajectories [14]. Filipiak et al. [15] proposed a graph-based algorithm for route identification in the Baltic Sea, illustrating how AIS-derived traffic networks can be represented and traversed. The resulting paths may sometimes exhibit artefacts such as zig-zagging; prior work has addressed this using post-processing steps (e.g., smoothing) to improve geometric realism. Onyango et al. [16] similarly proposed a framework for extracting traffic networks from AIS data, further supporting the use of graph representations for modelling vessel movement patterns.

A graph-based method of particular relevance to our work is the Data-driven AIS Trajectory Interpolation (DAISTIN) method [4]. DAISTIN constructs a directed graph from historical AIS data after applying stratified geometric sampling to reduce data volume. The sampled AIS observations become nodes in the graph, characterized by spatial information (longitude, latitude), directional information (heading), and additional information from the AIS messages. Edges are defined using two criteria: a neighbourhood constraint (proximity) and a direction constraint (heading alignment), as further discussed in Section 3.5. To interpolate a signal gap, the AIS messages before and after the gap are temporarily added

to the graph as auxiliary nodes. The algorithm then identifies the shortest path between them using the directed graph, which is returned once a smoothing post-processing step is applied. As with any sampled graph construction, the graph resolution (and therefore interpolation fidelity) depends on the number of sampled nodes, reflecting a trade-off between computational cost and granularity as AIS data volume increases. A limitation of DAISTIN is that directional information from non-sampled AIS messages is discarded, an issue that we hypothesize becomes more detrimental in dense and navigationally constrained settings (e.g., in ports). In response to this limitation, herein we propose two new methods, namely, xDAISTIN and xDAISTOUT.

3. Materials and Methods

3.1. AIS Data Collection

We collected AIS data using AISStream.io's API (https://aisstream.io/, accessed on 15 June 2026). This source was selected for its open-access availability, as commercial AIS data sets often involve high costs that limit accessibility for research [17]. Messages were collected continuously between 31 December 2023 and 24 January 2024 except on 4, 6, 7, 8, and 11 January 2024 when no or fewer messages were recorded, yielding an initial data set of 5,478,552 dynamic AIS messages from 2654 unique vessels. The data set was spatially restricted to the area of Port of Antwerp. As an initial cleaning step, messages with invalid identifiers (MMSI = 0) were removed. Table 1 summarizes the data set following key processing steps which we describe in the subsequent sections.

Table 1. Summary of the data set across processing steps. Numbers of AIS messages, unique vessels, extracted trips, and moving sub-trajectories retained after spatial restriction and cleaning, preprocessing, trip-level training/test partitioning, and the introduction of gaps in the test set. Data were collected via AISStream.io between 31 December 2023 and 24 January 2024. "–" denotes not applicable.

Data Set Stage	Messages	Vessels	Trips	Sub-Trajectories
Collected (raw)	5,478,552	2654	–	–
Spatially restricted and cleaned	5,462,834	2553	–	–
Preprocessed (moving sub-trajectories)	956,995	2377	5488	36,748
Training set (90% of trips)	884,044	† 2254	4939	34,124
Test set (10% of trips)	72,951	† 478	549	2624
Sparse test set (evaluation input)	15,189	478	549	2624

† The training/test split is performed at trip level; a vessel may therefore contribute trips to both sets (355 vessels appear in both).

3.2. Trajectory Segmentation

We retained only AIS messages from moving vessels, thereby excluding stationary activities (e.g., fishing, mooring, and anchoring). For each vessel, AIS messages were first sorted chronologically and then trips were identified by introducing a new trip whenever the time gap between consecutive messages exceeded 6 h. AIS points with speed over ground (SOG) below 1 knot were classified as stopped points. Each trip was subsequently segmented into consecutive moving and stopped sub-trajectories (see Algorithm 1). Since the objective of our work was to capture vessel movement patterns within the port, the data set was filtered to retain only moving segments containing at least three AIS data points. This preprocessing step yielded 36,748 moving sub-trajectories, extracted from 5488 vessel trips, comprising 956,995 AIS messages from 2377 unique vessels.

Algorithm 1 Moving sub-trajectory extraction

Input: Data of dynamic AIS messages $\mathcal{D}$.
Output: Moving sub-trajectories $\mathcal{S}$.
$T_{\text{max\_gap}} \leftarrow 6$ h
$v_{\text{min\_speed}} \leftarrow 1$ knot
$L_{\text{min\_points}} \leftarrow 3$
$\mathcal{S} \leftarrow [\,]$
for all $m \in \text{unique}(\mathcal{D}.\text{MMSI})$ **do**
  $\mathcal{D}_m \leftarrow \text{sortByTime}(\mathcal{D}[m])$
  $\mathcal{T} \leftarrow \text{splitByTimeGap}(\mathcal{D}_m, T_{\text{max\_gap}})$
  for all $T \in \mathcal{T}$ **do**
    $\mathcal{M} \leftarrow \text{movingSegments}(T, v_{\text{min\_speed}})$
    for all $M \in \mathcal{M}$ **do**
      if $|M| \geq L_{\text{min\_points}}$ **then**
        $\mathcal{S}.\text{append}(M)$
      end if
    end for
  end for
end for
return $\mathcal{S}$

3.3. Experimental Design and Evaluation Scope

The primary aim of our evaluation was to assess the effects of incorporating neighbourhood-level directional information within the DAISTIN framework specifically. We adopt DAISTIN as the principal comparative reference because of its methodological similarity to the proposed methods, facilitating a controlled analysis of node enrichment, graph construction, and traversal approaches. Because the methods discussed in Section 2 rely on different modelling assumptions, they were not included in the comparison; linear interpolation is provided as a simple non-graph baseline.

To assess the effectiveness in reconstructing incomplete trajectories, we used a hold-out evaluation approach. We randomly selected 10% of vessel trips to serve as the test set, while the remaining 90% were used for graph construction. The resulting training set contains 884,044 AIS messages across 4939 trips (34,124 moving sub-trajectories), and the test set contains 72,951 AIS messages across 549 trips (2624 moving sub-trajectories). Since the training/test split is performed at trip level, 355 vessels contribute trips to both sets.

To simulate incomplete AIS trajectories, synthetic gaps were introduced into the test trajectories as follows. For each moving sub-trajectory, starting from a retained observation, subsequent observations were removed until the temporal difference between two consecutive retained observations exceeded a randomly sampled threshold. This gap-duration threshold was modelled as a normally distributed random variable with mean $\mu = 6$ min and standard deviation $\sigma = 1.5$ min. The resulting sparse trajectories were used as test input to the interpolation method, while the original complete trajectories served as the ground truth for evaluation. Following the introduction of gaps in the test set trajectories, the test set retains 15,189 observed AIS messages over the 2624 evaluation sub-trajectories. The trip-level hold-out evaluation workflow and the procedure used to generate the sparse test trajectories are visualized in Figure 2.

3.4. Feature Engineering

Each AIS message provides information on the vessels' location (latitude and longitude), speed (SOG), and direction (course over ground (COG) and true heading). COG describes the vessel's direction of motion over the Earth's surface, whereas true heading reflects the direction the vessel's bow is pointing. In practice, however, COG can be noisy—particularly at low speeds and during close-quarter manoeuvres—and true heading

is frequently missing or unreliable. To obtain a single, consistently available directional feature for our methods, we therefore derive a calculated heading (CH) from consecutive observed positions (see Figure 3); where no subsequent position is available, CH falls back to the reported true heading and is left undefined when that is also unavailable. At the endpoints of a test-set gap, CH is computed using only retained observations, taking the arrival bearing at the observation preceding the gap and the departure bearing at the observation following it, so that no directional information is drawn from removed observations. Where the required neighbour is unavailable because the endpoint is the first or last retained observation of its sub-trajectory, the bearing between the two gap endpoints is used instead (20.4% and 13.0% of gaps, respectively).

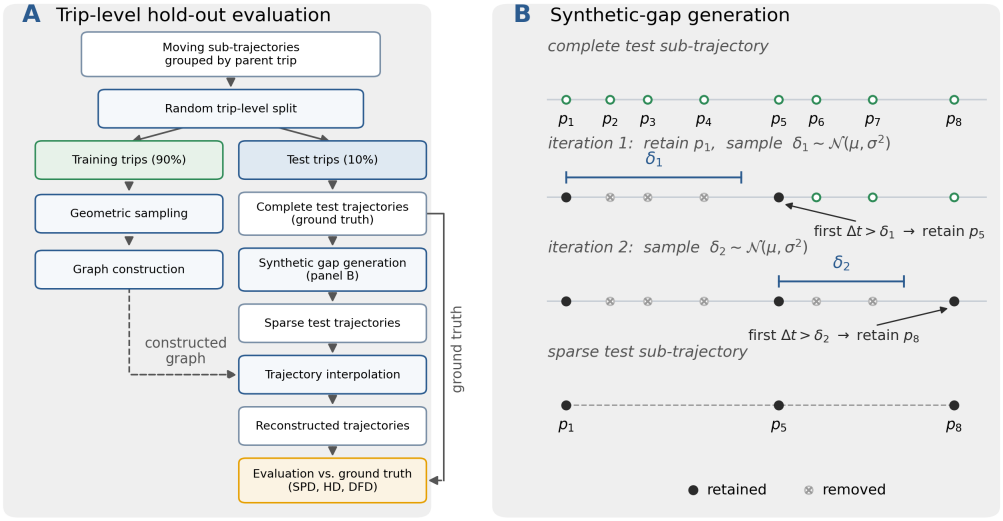


Figure 2. Trip-level hold-out evaluation and synthetic-gap generation workflow. (**A**) Vessel trips were randomly assigned into training and test partitions. The training trajectories, following geometric sampling, were used for graph construction, whereas the complete test trajectories served as the ground truth for evaluation. The sparse test trajectories (see panel (**B**)) were used as input to the interpolation methods. (**B**) Given a trajectory (green dots), starting from a retained observation (black), a gap-duration threshold $\delta \sim \mathcal{N}(\mu, \sigma^2)$ was sampled. Intermediate observations were removed (grey) until the first observation whose temporal difference from the retained observation exceeds δ was reached. That observation was retained (black), and the procedure was repeated until the end of the sub-trajectory. The last point of a given trajectory was always retained.

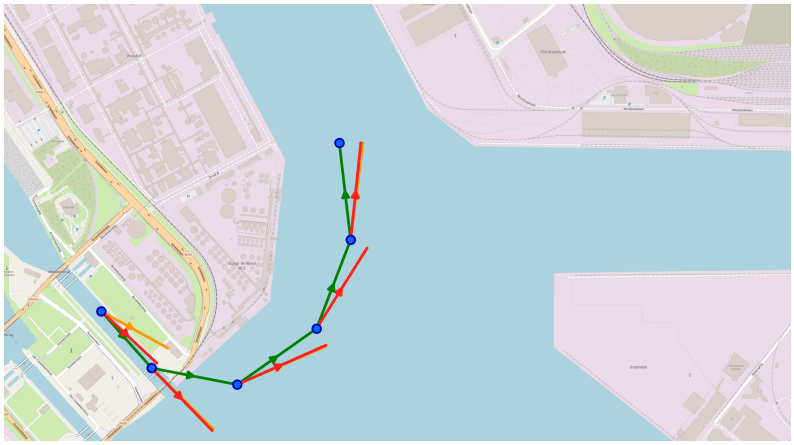


Figure 3. Difference between true heading, course over ground, and calculated heading. A sub-trajectory illustrating the directions of true heading (red line), course over ground (orange line), and calculated heading (green line).

Given two sequential points $A(\phi_1, \lambda_1)$ and $B(\phi_2, \lambda_2)$, the calculated heading (CH) is computed as

$$\mathrm{CH} = \left(\arctan 2(x, y) \cdot \frac{180^\circ}{\pi} \right) \bmod 360^\circ, \tag{1}$$

where $x = \sin(\Delta\lambda)\cos(\phi_2)$ and $y = \cos(\phi_1)\sin(\phi_2) - \sin(\phi_1)\cos(\phi_2)\cos(\Delta\lambda)$, ϕ_1, ϕ_2 are the latitudes of points A and B (in radians), λ_1, λ_2 are the corresponding longitudes (in radians), and $\Delta\lambda = \lambda_2 - \lambda_1$.

3.5. DAISTIN

DAISTIN [4] is an interpolation method that reconstructs missing trajectory segments by leveraging a pre-constructed directed graph derived from historical AIS data. Rather than directly estimating missing points, it searches this graph to identify the most plausible intermediate locations, thereby ensuring that interpolated paths conform to real-world movement patterns.

However, as the volume of historical AIS data increases, constructing a graph directly over all AIS observations can become computationally expensive. If all N historical AIS observations are used as graph nodes, radius-based edge construction can require $O(N^2)$ time in the worst case. This can occur when the spatial distribution of observations limits effective pruning of the search tree, for example, when observations are densely clustered, so that each node may have to consider up to $N - 1$ other nodes as candidate neighbours. DAISTIN instead constructs the graph over n sampled nodes, reducing this cost to $O(n^2)$, with $n \ll N$.

The sampling process begins by estimating the approximate neighbourhood density (AND) for each AIS point, defined as the number of neighbouring points within a specified radius (R_{AND}), efficiently computed using a ball-tree spatial index. The resulting AND distribution approximates a geometric distribution, with low-density points being more frequent. Based on their AND values, points are divided into quantiles and assigned density scores. Lower quantiles contain sparser regions and are preserved more fully, whereas higher-density regions are sampled more aggressively. This results in a spatially representative yet reduced data set; see Figure 4.

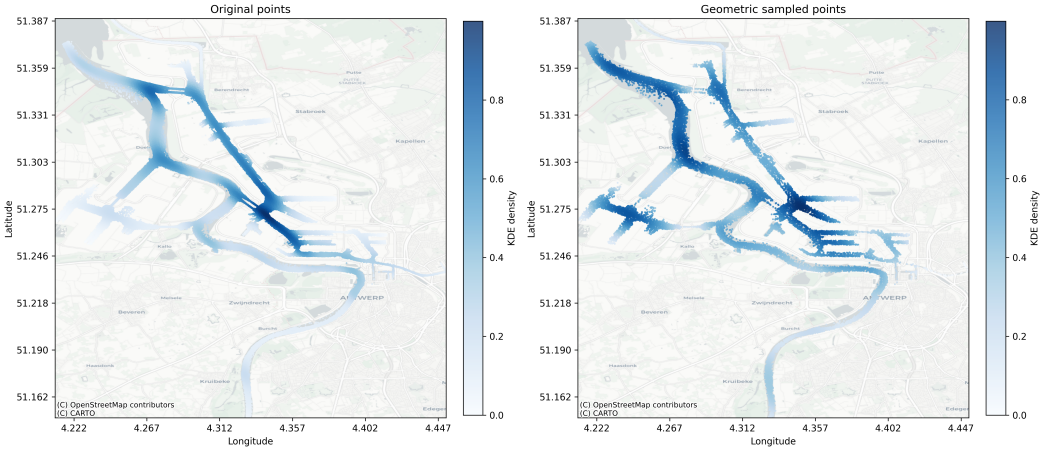


Figure 4. Geometric sampling adjusts spatial density distribution in AIS data. This figure illustrates the spatial density difference following the application of geometric sampling to AIS data. In both plots, darker shades of blue indicate higher point densities, while lighter shades indicate lower densities. The left plot illustrates the spatial density of the original data from the training set. The right plot demonstrates the impact of geometric sampling.

Once the sampled nodes are selected, DAISTIN constructs a directed graph in which each node represents an AIS data point, defined by its spatial (latitude, longitude) and

directional (heading) attributes. Although the original method employs true heading, initial experiments indicated that CH yielded better graph construction results in port environments (see Figure 3), and it was therefore adopted in our implementation. Edges are constructed based on two criteria:

1. Neighbourhood criterion: The geodesic distance between nodes does not exceed a specified threshold R_e.
2. Direction criterion: The angular difference between the vector from the source to the candidate node and the candidate node's CH must fall within a specified angular threshold $\Delta\theta$.

To efficiently identify node pairs that satisfy both criteria, a two-step process is followed (visualized in Figure 5):

1. Neighbourhood criterion evaluation: Nodes are first indexed using a ball-tree data structure to enable efficient spatial queries. For each node, a radius-based query is executed to retrieve all neighbouring nodes within the predefined distance threshold R_e.
2. Direction criterion evaluation: The candidate nodes identified in the previous step are then filtered based on the direction criterion. Only those whose angular alignment with the source node meets the specified threshold $\Delta\theta$ are retained. Directed edges are established between source and candidate nodes that satisfy both criteria.

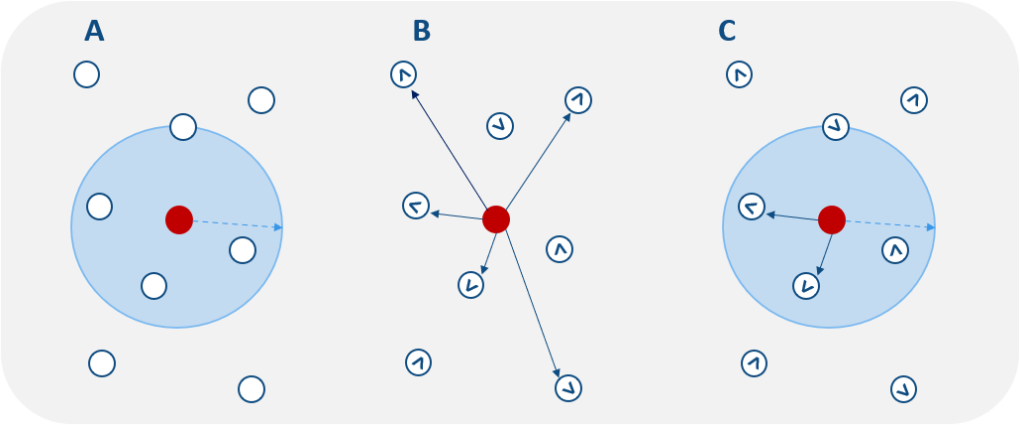


Figure 5. Visualization of DAISTIN's edge construction criteria. (**A**) Neighbourhood criterion: only nodes within a predefined spatial radius R_e from the source node (red dot) are considered candidates. (**B**) Direction criterion: a candidate node is considered only if the angular difference between the vector from the source to the candidate (selected vectors visualized in (**B**,**C**)) and the candidate's calculated heading falls within a specified angular threshold. (**C**) Combined criteria: only nodes that satisfy both the neighbourhood and direction criteria are connected by an edge.

The sampling stage operates on all N observations. A ball tree over the original two-dimensional data is constructed in $O(N \log N)$ time. AND estimation then requires one fixed-radius counting query per observation. Because only counts are required, any subtree lying entirely inside or outside the query radius is resolved in constant time from its stored index range, whereas only subtrees intersecting the query boundary are searched recursively. Here, we assume effective pruning which requires the query radius to be small relative to the spatial extent of the data; otherwise, $O(N^2)$ time may be required. Finally, quantile assignment needs $O(N \log N)$ time for sorting.

Interpolation is triggered when the time difference between two consecutive AIS points (denoted as A and B) exceeds a threshold. Since these points are unlikely to match any existing node in the graph exactly (in both location and heading), they are temporarily inserted as auxiliary nodes. Edges are then formed between these nodes and the rest of the graph based on the same neighbourhood and direction criteria.

The A^* pathfinding algorithm is consequently used to find the shortest path, based on the Haversine distance, between A and B. If the A^* pathfinding algorithm fails to identify a valid path, no interpolation is performed. If a path is found, a post-processing step is applied to eliminate any zig-zag patterns and evenly space the AIS data points along the path. To achieve this, the Douglas–Peucker algorithm [18] is used to identify the most significant data points in the trajectory, which are typically manoeuvring points, thus reducing zig-zag patterns. The simplified path is then re-evaluated, and any sub-segment exceeding the predefined distance threshold is densified via geodesic interpolation.

3.6. xDAISTIN

3.6.1. Stratified Sampling

As illustrated in the first plot of Figure 4, some areas exhibit higher density than others, typically corresponding to berths, common routes, or port entrances and exits. Including all data points from these dense areas would lead to an overly complex graph. Therefore, we implement stratified sampling based on the density of each point. Following the DAISTIN method [4], we measure the AND of each data point and partition the points into Q quantiles. This stratification allows us to group neighbourhoods based on their density, aiming to sample more points from less dense regions and fewer points from more dense regions, establishing an inversely proportional relationship between the number of points and spatial density.

To achieve this inversely proportional relationship, sampling rates across quantiles are defined using geometric series. This requires specifying the ratio parameter ρ (where $0 < \rho < 1$), the number of quantiles Q, and the desired number of sample points $n^\star$. Given that $\rho \in (0, 1)$, we know the sum of the geometric series is finite, allowing us to determine our sampling rates for each quantile individually. A scaling factor S_1 is computed such that the sum of the contributions across all quantiles approximates $n^\star$, and is given by

$$S_1 = \frac{n^\star}{\frac{1-\rho^Q}{1-\rho} \cdot P_q} \tag{2}$$

where $P_q = \frac{N}{Q}$ represents the expected number of data points per quantile under an equal partitioning assumption. In this way, we can define the individual sampling rates and sample sizes corresponding to each quantile. The sample size extracted from each quantile (n_i) is given by

$$n_i = \lfloor S_i \cdot P_q \rfloor, \qquad S_i = S_1 \cdot \rho^i, \qquad i = 0, 1, \ldots, Q-1. \tag{3}$$

Since the sampling stage uses the same AND-based, quantile-stratified procedure as DAISTIN, its time complexity remains the same. Furthermore, since the sampling strategy may interact with the subsequent method for edge construction, the experiments we conducted are controlled insofar as the sampling method is kept constant; but this does not ensure that the ranking of methods would be preserved if a different sampling strategy were pursued.

3.6.2. Heading-Preserving Geometric Sampling

The previously described stratified sampling procedure effectively obtains a representative spatial sample by accounting for the AND of each point. However, spatial proximity between AIS messages does not necessarily imply similarity in other message attributes, such as vessel heading. Consequently, although the stratified sampling algorithm minimizes the loss of spatial information, it does not ensure that directional information is adequately preserved. To address this limitation, we incorporate the headings of neighbouring points within a predefined radius R_h into each data point representing a node

(process visualized in Figure 6). By leveraging these headings, we fit a KDE to capture the heading distribution across each node's neighbourhood. A KDE is fitted only when at least two neighbouring headings are available.

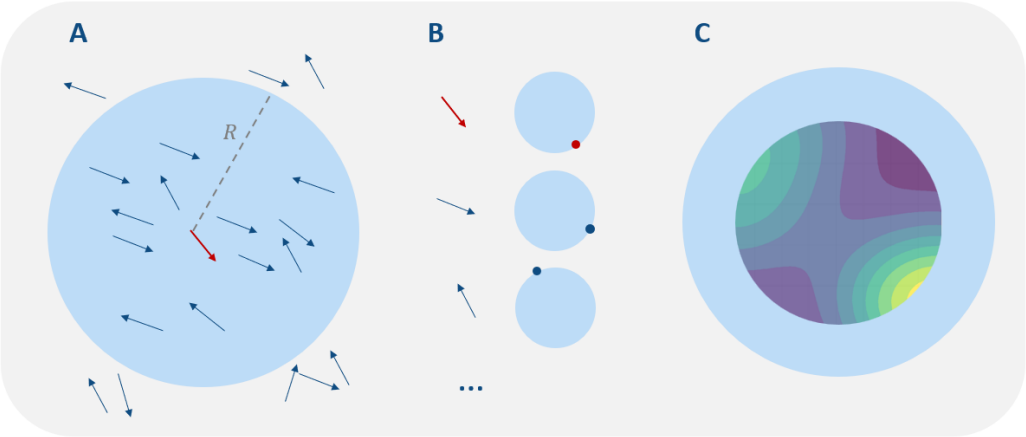


Figure 6. Spatial neighbourhood and directional encoding for KDE. (**A**) For a given radius R_h, each sampled point (red arrow) defines a spatial neighbourhood (light-blue disc); neighbouring headings are shown as blue arrows. (**B**) Each heading is mapped to the unit circle by its sine and cosine, producing a point on the circle. (**C**) These encoded heading points are then used to fit the kernel density estimator.

To define the probabilistic heading estimator, the circular nature of headings needs to be considered. Each heading, h, can be represented as a point on the unit circle, denoted as (x_h, y_h), by employing cyclical encoding. This is implemented as follows:

$$x_h = \cos\left(\frac{2 \cdot \pi \cdot h}{360}\right), \quad y_h = \sin\left(\frac{2 \cdot \pi \cdot h}{360}\right) \tag{4}$$

Let $k_{h,i}$ denote the number of original AIS observations falling within R_h of observation i, and let $\bar{k}_h$ denote the mean of $k_{h,i}$ over all N observations. Heading neighbourhoods are computed for every observation prior to sampling, so a ball tree over the N observations is constructed in $O(N \log N)$ time and a fixed-radius query is performed for each observation. Unlike the counting queries of Section 3.5, these queries return the neighbouring headings themselves, so each contributes an additive term proportional to the number of points retrieved. Assuming $O(\log N)$ traversal per radius query, as in Section 3.5, retrieving the heading neighbourhoods of all N observations therefore requires

$$O\left(N \log N + \sum_{i=1}^{N} k_{h,i}\right) = O\left(N \log N + N\bar{k}_h\right),$$

with a worst-case cost of $O(N^2)$.

Fitting the estimator for a sampled node i stores its $k_{h,i}$ encoded headings, obtained from Equation (4), in a spatial index, at a cost of $O(k_{h,i} \log k_{h,i})$. Since an estimator is fitted only for nodes with at least two neighbouring headings, the total fitting cost over the sampled nodes is

$$O\left(\sum_{i=1}^{n} k_{h,i} \log k_{h,i}\right)$$

3.6.3. Defining Edges

The definition of an edge follows DAISTIN and combines a neighbourhood and a direction criterion. The neighbourhood criterion is unchanged: only nodes located within a radius R_e around the node under investigation are considered as candidates.

The direction criterion exploits the node-specific heading estimator (KDE) which, as described in Section 3.6.2, is fitted in the cyclically encoded heading space rather than directly on the scalar heading. Recall Equation (4); a heading h is encoded as the point on the unit circle, with a two-dimensional KDE fitted on these encoded points. For a destination node B, let $\hat{f}_B^{(2D)}(x, y)$ denote this fitted two-dimensional density.

The scalar directional score used by the edge test is obtained by restricting $\hat{f}_B^{(2D)}$ to the unit circle and renormalizing it over the full angular domain, yielding the one-dimensional circular density:

$$\hat{f}_B(\phi) = \frac{\hat{f}_B^{(2D)}\left(\cos\frac{2\pi\phi}{360},\ \sin\frac{2\pi\phi}{360}\right)}{Z_B}, \qquad Z_B = \int_0^{360} \hat{f}_B^{(2D)}\left(\cos\frac{2\pi\psi}{360},\ \sin\frac{2\pi\psi}{360}\right) d\psi, \tag{5}$$

so that $\int_0^{360} \hat{f}_B(\phi)\, d\phi = 1$.

Consider a starting node A and a candidate destination node B, and let $\theta_{A\to B}$ be the bearing of $\vec{AB}$ and $\Delta\theta$ the angular half-window threshold. We define the directional compatibility score as

$$p_{A\to B}^{\text{xDAISTIN}} = \int_{\theta_{A\to B}-\Delta\theta}^{\theta_{A\to B}+\Delta\theta} \hat{f}_B(\phi)\, d\phi, \tag{6}$$

The edge $A \to B$ is accepted if and only if

$$d(A, B) \le R_e \qquad \text{and} \qquad p_{A\to B}^{\text{xDAISTIN}} > \tau, \tag{7}$$

where τ is a tunable probability threshold. When a node has fewer than two neighbouring headings, the KDE is not fitted, and the implementation falls back to a deterministic angular threshold test, accepting the edge when the absolute angular difference between $\theta_{A\to B}$ and the node heading is at most $\Delta\theta$.

A directed edge in the graph is added only when both criteria are satisfied, as illustrated in Figure 7. Since independently assessing each criterion for every node and edge is computationally expensive, we adopt the ball-tree approach from the original DAISTIN method, modifying it to incorporate our specific criteria. We provide the pseudocode of xDAISTIN graph construction in Algorithm 2.

Let $k_{h,e}$ denote the number of local heading observations associated with the node whose KDE is evaluated for candidate edge e. In xDAISTIN this is the destination and in xDAISTOUT the source node of e. Because the numbers of angular samples used for local-window integration and full-circle normalization are fixed, a conservative evaluation cost for edge e is $O(k_{h,e})$. Over all E_c candidate edges, the KDE-based evaluation cost is therefore

$$O\left(\sum_{e=1}^{E_c} k_{h,e}\right) = O\left(E_c \bar{k}_{h,E}\right),$$

where

$$\bar{k}_{h,E} = \frac{1}{E_c} \sum_{e=1}^{E_c} k_{h,e}$$

is the average heading-neighbourhood size over the KDE evaluations performed for candidate edges.

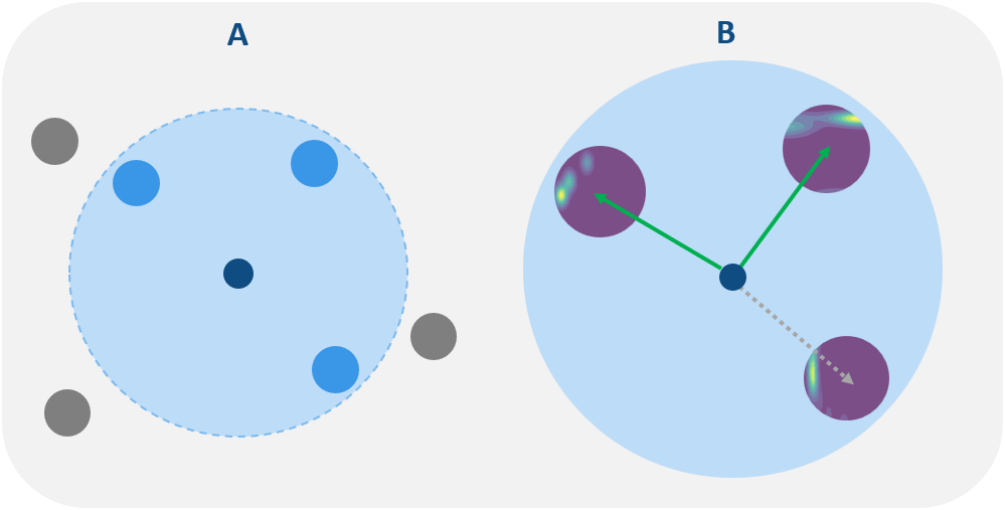


Figure 7. Edge criteria of xDAISTIN. (**A**) Neighbourhood criterion: only nodes within a radius (light blue area) R_e are treated as candidate neighbours (light blue nodes) of a sampled node (dark blue); nodes outside (grey) are ignored. (**B**) Direction criterion: for each candidate neighbour, the potential edge direction is evaluated using the heading-based KDE. Edges are created only when this direction has probability above a threshold (solid green arrows); candidates that do not satisfy this threshold are discarded (dashed grey arrow).

Algorithm 2 Construct xDAISTIN graph

Input: Training AIS messages M; number of graph nodes $n^\star$, AND neighbourhood radius R_{AND}, number of density quantiles Q; heading-neighbourhood radius R_h; maximum edge distance R_e; probability threshold τ; angular tolerance $\Delta\theta$; geometric-sampling ratio ρ;
Output: Directed interpolation graph $G = (V, E)$.
for all AIS messages $m \in M$ **do**
  $c_m \leftarrow \text{CountNeighbours}(m, R_{\text{AND}})$.
end for
Divide M into density quantiles $\{M_q\}_{q=1}^{Q}$ according to c_m.
$V \leftarrow \text{GeometricStratifiedSampling}\big(\{M_q\}_{q=1}^{Q}, n^\star, \rho\big)$
for all nodes $v \in V$ **do**
  $H_v \leftarrow \{\text{heading}(m) : m \in M,\ d(m, v) \le R_h\}$.
  if $|H_v| \ge 2$ **then**
    $\hat{f}_v \leftarrow \text{FitCircularDensity}(H_v)$.
  else
    $\hat{f}_v \leftarrow \varnothing$
  end if
end for
$\mathcal{P}_{R_e} \leftarrow \text{FindDirectedPairsWithinRadius}(V, R_e)$.
$E \leftarrow \emptyset$.
for all directed node pairs $(A, B) \in \mathcal{P}_{R_e}$ **do**
  $\theta_{A\to B} \leftarrow \text{Bearing}(A, B)$.
  $$p \leftarrow \begin{cases} \hat{f}_B([\theta_{A\to B} - \Delta\theta,\ \theta_{A\to B} + \Delta\theta]), & \text{if } \hat{f}_B \neq \varnothing, \\ 1, & \text{else if } \text{AngularDiff}(\theta_{A\to B}, h_B) \le \Delta\theta, \\ 0, & \text{otherwise.} \end{cases}$$
  if $p > \tau$ **then**
    $w \leftarrow -\log(p)$.
    $E \leftarrow E \cup \{(A, B, w)\}$.
  end if
end for
return $G = (V, E)$.

3.6.4. Interpolation

A gap is defined as two consecutive AIS points, points A and B, of a trajectory that are separated by a time difference that exceeds a predefined threshold (herein set to 3 min). We interpolate the gaps of trajectories using the constructed graph. Points A and B are temporarily added as nodes to the graph, and edges to neighbouring nodes are created according to the previously defined neighbourhood and direction criteria.

To bound the interpolation, we estimate the maximum distance that the vessel could have travelled between A and B. We use the speed at A and the time difference $t_\star$:

$$d_{\max} = \max(\text{speed}_A, \text{speed}_B, 1\,\text{knot}) \cdot t_\star$$

assuming consistent units (e.g., knots). The lower bound of 1 knot prevents points classified as stops (speed < 1) from implying a short distance travelled.

We then search for the most probable path from A to B using Dijkstra's algorithm with a cumulative distance constraint. Each directed edge in the graph is assigned a weight equal to the negative logarithm of its transition probability. Dijkstra's algorithm therefore seeks the path that minimizes the sum of these weights, i.e., the most probable path. During the search, we also track the cumulative travelled distance along each candidate path. If this distance exceeds $d_{\max}$ before reaching B, the corresponding path is discarded and the algorithm continues with the remaining candidates. In contrast to the original DAISTIN method, which finds the shortest-distance path, our approach derives a maximum travel distance from the time gap and speed and then maximizes path probability—based on KDE-estimated edge transition probabilities—subject to this distance constraint.

3.7. Variants of the Methods

In addition to the original DAISTIN and the proposed xDAISTIN, we also implement xDAISTOUT, as well as several variants of all methods, and assess their performance. The undirected DAISTIN variant retains the asymptotic time complexity of the original DAISTIN, while all xDAISTIN and xDAISTOUT variants retain that of xDAISTIN.

3.7.1. xDAISTOUT

xDAISTOUT is a variant of xDAISTIN that modifies the direction criterion used to construct edges (difference visualized in Figure 8). For a candidate edge from node A to node B that already satisfies the distance criterion, we compute the bearing of the vector $\vec{AB}$, apply cyclical encoding, and evaluate its likelihood under the KDE fitted at the starting node A rather than the destination node B. An edge between A and B is created only if this bearing falls within an angular window around a high-probability region of A's KDE.

Formally, the only change relative to xDAISTIN is that the candidate bearing is scored against the source node's circular density $\hat{f}_A$ (obtained from $\hat{f}_A^{(2D)}$ exactly as in Section 3.6.2) rather than the destination node's,

$$p_{A\to B}^{\text{xDAISTOUT}} = \int_{\theta_{A\to B}-\Delta\theta}^{\theta_{A\to B}+\Delta\theta} \hat{f}_A(\phi)\, d\phi, \tag{8}$$

and the edge $A \to B$ is accepted if and only if $d(A,B) \le R_e$ and $p_{A\to B}^{\text{xDAISTOUT}} > \tau$.

3.7.2. Undirected Version

Graph $X_{\text{undirected}}$ removes edge directionality from Graph X, turning all admissible edges into undirected connections, thereby allowing bidirectional traversal. Undirected variants are constructed for DAISTIN, xDAISTIN, and xDAISTOUT.

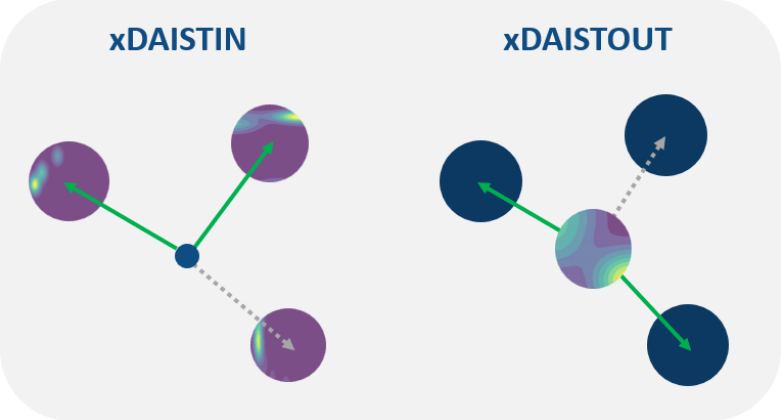


Figure 8. Comparison of the direction criterion used by xDAISTIN and xDAISTOUT. For a candidate directed edge from node A to node B, xDAISTIN evaluates the bearing $\theta_{A\to B}$ using the circular heading density fitted at the destination node B, whereas xDAISTOUT evaluates the same bearing using the circular heading density fitted at the source node A. Solid green arrows represent candidate edges whose directional compatibility is within the probability thresholds and are therefore accepted; dotted grey arrows represent candidate edges that did not meet the criterion.

3.7.3. Shortest Version

Graph X_{shortest} retains the graph topology of Graph X but changes the path selection rule: instead of maximizing path probability, it selects the path with the minimum accumulated Haversine distance. This is the original mechanism of path finding for DAISTIN [4]. We also implement it for the proposed xDAISTIN and xDAISTOUT methods.

3.7.4. Undirected-and-Shortest Version

Graph $X_{\text{undirected \& shortest}}$ combines the two previous modifications: edges are undirected, and path finding is based on the shortest Haversine distance. This variant is implemented for both xDAISTIN and xDAISTOUT.

3.8. Post-Processing

Our approach to post-processing is the same as the original DAISTIN's approach, which incorporates the Douglas–Peucker algorithm in conjunction with a linear interpolation algorithm [4].

Regarding the Douglas–Peucker algorithm, it begins by selecting the endpoints of the corresponding interpolated path. The algorithm then finds the point that is farthest from the line segment connecting these endpoints. If the farthest point is within a specified tolerance, it is removed, as the line segment sufficiently represents the original path within the given tolerance. If the point is outside the tolerance, it is retained, and the process is recursively applied to the segments formed by the current point and the endpoints. The result is a simplified path that approximates the original path with fewer points, effectively reducing complexity while maintaining its overall initial form.

By definition, this algorithm eliminates points that exhibit linear or near-linear behaviour to simplify the path. However, if a substantial portion of the trajectory follows a linear pattern, this can create large gaps between retained points. To address this, and in line with the original DAISTIN post-processing step, we apply linear interpolation to reconnect segments where the distance between consecutive points exceeds a predefined threshold.

Let m denote the number of points in the reconstructed path before simplification. For the standard Douglas–Peucker implementation, the running time depends on the recursive split pattern. It is $\Theta(m)$ when the path is accepted as a single segment, $\Theta(m \log m)$ under balanced recursive splitting, and $\Theta(m^2)$ in the worst case. If densification produces d

points, the subsequent densification step requires $O(d)$ time. Because reconstructed paths typically contain substantially fewer points than the underlying graph contains nodes, post-processing is not expected to dominate the overall runtime.

3.9. Implementation Details

For the implementation of our graph-based interpolation methods, we used Python (version 3.13.5) with several packages: networkX [19], scikit-learn [20], NumPy [21], shapely [22], and rdp [23]. For data cleaning, analysis, and visualization, in addition to the previously mentioned libraries, we also utilized the following Python packages: Pandas [24], Matplotlib (version 3.10.0) [25], contextily [26], folium [27], SciPy [28], and Seaborn (version 0.13.2) [29]. For reproducibility, Table 2 summarizes the main hyperparameter values and implementation and evaluation settings used in the reported experiments.

Table 2. Hyperparameter and evaluation-setting values. Final hyperparameter, implementation, and evaluation-setting values used in the reported experiments. The table uses "–" when a symbol is not specified for a parameter in the manuscript, and the superscript asterisk in n^* denotes the target, rather than the actual, number of sampled graph nodes.

Parameter	Symbol	Value Used
Trip-segmentation time gap	T_{max_gap}	6 h
Stopped-point speed threshold	v_{min_speed}	1 knot
Minimum moving-sub-trajectory length	L_{min_points}	3 AIS data points
Training/test split ratio	–	0.9/0.1
Hold-out split random seed	–	50
Gap-duration threshold distribution	μ, σ	$\mu = 6$ min, $\sigma = 1.5$ min
Target number of sampled graph nodes	n^*	$\{100, 120, 140, 160, 180\} \times 10^3$
Number of AND quantile groups	Q	20
Geometric-sampling ratio	ρ	0.8
AND neighbourhood radius	R_{AND}	25 m
Heading-neighbourhood radius	R_h	25 m
KDE kernel	–	Gaussian
KDE bandwidth	–	0.05
Minimum neighbouring headings for KDE fitting	–	2
Angular-window integration samples	–	100
Full-circle normalization samples	–	720
Candidate-edge radius	R_e	50 m
Angular half-window	$\Delta\theta$	30°
Directional-probability threshold	τ	0.01
Minimum speed used in travel-distance bound	–	1 knot
Douglas–Peucker tolerance	–	0.0005
Interpolation trigger	–	3 min
Post-processing densification threshold	–	100 m

Lastly, we note that the proposed methods were developed with port-specific use in mind, rather than for a global maritime traffic model. Relatedly, we stress that for a specific area of interest, graph construction is performed offline, and the resulting precomputed graph then used for online interpolation queries. Extending our approach to more extensive maritime domains would involve the partitioning of the domain into separate, possibly overlapping regions, with a separate graph constructed for each region. These graphs could be periodically rebuilt as new AIS data become available or as traffic patterns change.

The development of efficient and accurate incremental update methods is a viable topic for future work.

4. Results

The reconstructed trajectories in the test set were evaluated quantitatively against the complete ground truth trajectories and qualitatively through visual inspection of selected cases with respect to the port's geometry and layout and navigational rules.

4.1. Evaluation Metrics

For quantitatively evaluating reconstructed trajectories, we investigate the use of three widely used metrics in the literature that capture complementary aspects of trajectory similarity: segmented path distance (SPD), Hausdorff distance (HD), and discrete Fréchet distance (DFD) [30]. As illustrated in Figure 9, each metric responds differently to the selected trajectory reconstructions. Herein, the SPD and HD are reported in kilometres, whereas the DFD is reported in decimal degrees.

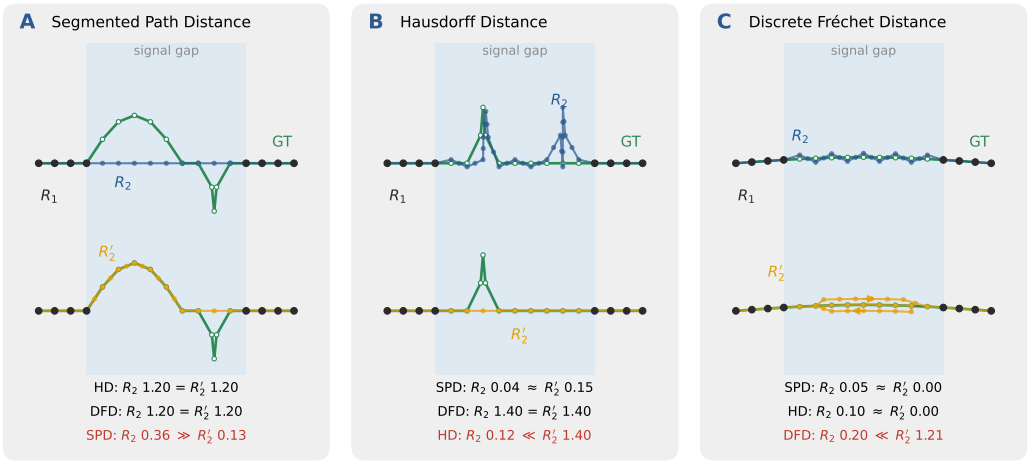


Figure 9. Evaluation metrics for trajectory reconstruction. Each panel shows the complete ground truth trajectory (GT, green), the sparse trajectory R_1 obtained by removing the observations inside the shaded signal gap, and two candidate interpolations of the gap, R_2 (blue, top row) and R_2' (orange, bottom row). Dots denote trajectory points, and red text highlights the metric illustrated in the corresponding panel. (**A**) Segmented Path Distance (SPD): an example in which SPD distinguishes between the two reconstructions, while HD and DFD assign them equal values. (**B**) Hausdorff Distance (HD): an example illustrating the sensitivity of HD to a large local deviation (green peak). (**C**) Discrete Fréchet Distance (DFD): an example in which a longer reconstruction is only penalized by DFD.

4.1.1. Segmented Path Distance

The SPD is a point-to-segment metric that captures both local deviations and structural differences between two trajectories. It is defined as

$$SPD(T^1, T^2) = \frac{1}{n_1} \sum_{i_1=1}^{n_1} \min_{i_2 \in \{0,\ldots,n_2-1\}} D_{ps}\left(p_{i_1}^1, s_{i_2}^2\right), \tag{9}$$

where T^1 and T^2 are the trajectories being compared, n_1 and n_2 are their numbers of points, $p_{i_1}^1$ is the i_1-th point of T^1, and $s_{i_2}^2$ is the i_2-th segment of T^2.

The point-to-segment distance $D_{ps}\left(p^1_{i_1}, s^2_{i_2}\right)$ is defined as

$$D_{ps}\left(p^1_{i_1}, s^2_{i_2}\right) = \begin{cases} \left\| p^1_{i_1} p^{1_{\text{proj}}}_{i_1} \right\|_2 & \text{if } p^{1_{\text{proj}}}_{i_1} \in s^2_{i_2} \\ \min\left(\left\| p^1_{i_1} p^2_{i_2} \right\|_2, \left\| p^1_{i_1} p^2_{i_2+1} \right\|_2 \right) & \text{otherwise} \end{cases} \tag{10}$$

where $p^{1_{\text{proj}}}_{i_1}$ is the orthogonal projection of $p^1_{i_1}$ onto the segment $s^2_{i_2}$, and distances are computed using the Haversine formula.

Because the SPD aggregates deviations across points (rather than focusing on a single worst deviation), we treat the SPD as the primary evaluation metric for performance assessment.

4.1.2. Discrete Hausdorff Distance

The discrete Hausdorff distance (hereafter referred to as the Hausdorff distance (HD)) quantifies the maximum of the minimum point-to-point distances between two trajectories, with each trajectory represented as a set of points. For trajectories T^1 and T^2, using the Haversine point-to-point distance D_h,

$$HD(T^1, T^2) = \max_{i_1 \in [1 \ldots n_1]} \min_{i_2 \in [1 \ldots n_2]} D_h(p^1_{i_1}, p^2_{i_2}). \tag{11}$$

In this work, we use the directed formulation of the Hausdorff distance, which measures the largest distance from a point in T^1 to its closest point in T^2. Since the HD is based on a maximum over all points, it is sensitive to outliers.

4.1.3. Discrete Fréchet Distance

The Fréchet distance is a measure of similarity between two curves that takes into account the location and ordering of the points along the curves. It is typically explained as the minimum "leash" length required to connect a point moving along one trajectory to a point moving along the other.

The DFD [31] quantifies the similarity between two polygonal curves, represented here as trajectories. Consider two trajectories T^1 and T^2, where

$$T^1 = ({t^1}_1, {t^1}_2, \ldots, {t^1}_{n_1}) \quad \text{and} \quad T^2 = ({t^2}_1, {t^2}_2, \ldots, {t^2}_{n_2})$$

represent the sequences of points of each trajectory. A coupling L between T^1 and T^2 is defined as a sequence of pairs

$$({t^1}_{a_1}, {t^2}_{b_1}), ({t^1}_{a_2}, {t^2}_{b_2}), \ldots, ({t^1}_{a_m}, {t^2}_{b_m})$$

such that:

- The sequence starts and ends with the endpoints of both trajectories: $a_1 = 1$, $b_1 = 1$, $a_m = n_1$, and $b_m = n_2$;
- For each $i = 1, \ldots, m-1$, the indices satisfy:

$$a_{i+1} \in \{a_i, a_i + 1\}, \quad b_{i+1} \in \{b_i, b_i + 1\},$$

ensuring that the traversal respects the order of points in both trajectories.

The length of the coupling L, denoted $\|L\|$, is defined as the maximum distance between any pair of matched points:

$$\|L\| = \max_{i=1,\ldots,m} d({t^1}_{a_i}, {t^2}_{b_i}).$$

Given this, the DFD between T^1 and T^2 is defined as

$$\mathrm{DFD}(T^1, T^2) = \min\left\{ \|L\| \,\middle|\, L \text{ is a valid coupling between } T^1 \text{ and } T^2 \right\}.$$

Among the three metrics, only the DFD exhibits symmetry. This symmetry can negatively affect trajectory-level evaluation because it may under-penalize interpolation failures: when gaps remain unfilled, the predicted trajectory can become a subsequence of the ground truth, resulting in an artificially small (sometimes zero) DFD. Accordingly, while we report the DFD in selected experiments, it should be interpreted with caution, especially when the interpolation completion rate is low.

4.2. Interpolation Completion Rate

Given points A and B that define a gap, we define the interpolation completion rate of a graph method as the percentage of gaps for which the graph found a path connecting A to B. Table 3 reports the interpolation completion rate for each method.

Table 3. Percentage of interpolated gaps across methods and number of nodes. Comparison of interpolation completion rate across all DAISTIN, xDAISTIN, and xDAISTOUT variants with varying number of graph nodes. Bold indicates the highest interpolation completion rate for a given number of nodes.

	Number of Nodes				
Method	**100 k**	**120 k**	**140 k**	**160 k**	**180 k**
DAISTIN	48.48%	63.45%	75.98%	80.62%	84.61%
DAISTIN undirected	72.35%	78.94%	85.46%	89.36%	91.23%
xDAISTIN	88.02%	90.83%	92.81%	94.91%	96.42%
xDAISTIN undirected	91.56%	93.35%	95.79%	97.60%	99.12%
xDAISTIN shortest	91.36%	94.03%	95.39%	95.72%	97.46%
xDAISTIN undirected and shortest	**92.92%**	**95.75%**	**97.55%**	**97.64%**	**99.13%**
xDAISTOUT	65.75%	72.52%	78.44%	82.40%	85.17%
xDAISTOUT undirected	91.56%	93.20%	95.55%	97.43%	97.80%
xDAISTOUT shortest	72.12%	76.45%	81.12%	85.87%	89.08%
xDAISTOUT undirected and shortest	92.89%	95.48%	97.33%	97.43%	97.80%

There are marked differences between the reported percentages of DAISTIN compared to its undirected variant as well as to all the proposed methods. Removing directionality substantially improves DAISTIN's interpolation ability (from 48% to 72% at 100 k; and from 85% to 91% at 180 k). For both xDAISTIN and xDAISTOUT, certain variants exhibit higher interpolation completion rates. For xDAISTIN, the interpolation percentages range from 88 to 96% (directed), 92 to 99% (undirected), 91 to 97% (shortest), and 93 to 99% (undirected and shortest). For xDAISTOUT, across 100 k–180 k sampled nodes, the interpolation percentages range from 66 to 85% (directed), 92 to 98% (undirected), 72 to 89% (shortest), and 93 to 98% (undirected and shortest). Similar to DAISTIN, the undirected variants of both xDAISTIN and xDAISTOUT exhibit higher interpolation completion rates. Both the shortest and the undirected-and-shortest variants also exhibit higher interpolation completion rates than the original methods. The highest interpolation completion rates are achieved by the undirected-and-shortest variant, with the undirected variant close behind, for both xDAISTIN and xDAISTOUT. Notably, across all methods (rows), interpolation completion rates increase monotonically with the increase in graph nodes. Interpolation completion rates should be interpreted alongside trajectory reconstruction metrics (see Section 4.3) since a high interpolation completion rate does not guarantee accurate trajectory reconstruction.

4.3. Quantitative Evaluation

All the quantitative results, i.e., SPD, DFD, and HD, across the evaluated methods and graph sizes, are presented in Tables 4–6. In addition, we tested differences between family variants at a fixed graph size through column-wise comparisons, and differences between graph sizes for a fixed method through row-wise comparisons. Matching coloured asterisks

identify pairs of variants for which no statistically significant column-wise difference was detected after Holm–Bonferroni correction ($p_{adj} \geq 0.05$). Matching coloured crosses have the same interpretation for row-wise graph-size comparisons. Pairs that are not linked by a matching symbol and colour differed significantly ($p_{adj} < 0.05$).

Pairwise comparisons were based on two-sided paired-sample *t*-tests applied to the 2624 test set error differences, with Holm–Bonferroni correction applied. For column-wise comparisons involving more than two variants, namely, xDAISTIN and xDAISTOUT, a Friedman repeated-measures omnibus test was conducted before the pairwise tests. For row-wise comparisons, a Friedman repeated-measures omnibus test across the five graph sizes was conducted for every variant before pairwise follow-up testing. All Friedman tests were significant after correction ($p_{adj} < 0.001$). The linear baseline was included in the tables for ease of comparative analysis but was excluded from the tests.

Table 4. Summary of evaluation of DAISTIN variations. Comparison of the original DAISTIN and the undirected variant across different graph sizes using SPD, HD, and DFD means. Matching coloured asterisks indicate non-significant differences between variants at the same graph size, whereas matching coloured crosses indicate non-significant differences between graph sizes for the same variant ($p_{adj} \geq 0.05$). Bold indicates the best performance under a metric for a given number of nodes (column-wise).

Metric	Variation	100 k	120 k	140 k	160 k	180 k
SPD	Original	0.0364	0.0344	0.0320	0.0311	**0.0296** *
	Undirected	**0.0339**	**0.0321**	+**0.0307**	+**0.0303**	+0.0302 *
	Baseline (linear)	0.0350	0.0350	0.0350	0.0350	0.0350
DFD	Original	+0.0056	+0.0056	0.0054	0.0052 *	**0.0047**
	Undirected	+**0.0049**	+**0.0048**	**0.0044**	+**0.0049** *	0.0055
	Baseline (linear)	0.0491	0.0491	0.0491	0.0491	0.0491
HD	Original	0.4348	0.3647	0.3030	0.2767	0.2402
	Undirected	0.3222	**0.2776**	**0.2332**	**0.2020**	**0.1919**
	Baseline (linear)	**0.3070**	0.3070	0.3070	0.3070	0.3070

Table 5. Summary of evaluation of xDAISTIN variations. Comparison of the xDAISTIN graph variations across different graph sizes using SPD, HD, and DFD means. Matching coloured asterisks indicate non-significant differences between variants at the same graph size, whereas matching coloured crosses indicate non-significant differences between graph sizes for the same variant ($p_{adj} \geq 0.05$). Bold indicates the best performance under a metric for a given number of nodes (column-wise).

Metric	Variation	100 k	120 k	140 k	160 k	180 k
SPD	Original	0.0325 *	0.0310	0.0298	0.0291 **	0.0286 **
	Undirected	**0.0304**	**0.0292**	**0.0274**	**0.0264**	**0.0258**
	Shortest	0.0345	0.0327 *	0.0319	0.0290 *	0.0288 *
	Undirected and shortest	+0.0326 *	+0.0325 *	0.0312	0.0285 *	0.0282 *
	Baseline (linear)	0.0350	0.0350	0.0350	0.0350	0.0350
DFD	Original	0.0041	0.0039	0.0038	0.0037	0.0035
	Undirected	**0.0038**	**0.0037**	**0.0036**	0.0034	**0.0031** *
	Shortest	+0.0075	0.0080	+0.0078	**0.0033**	**0.0031** **
	Undirected and shortest	0.0068	0.0081	0.0079	0.0034	**0.0031** *
	Baseline (linear)	0.0491	0.0491	0.0491	0.0491	0.0491
HD	Original	0.2418	0.2194	0.2095	0.1864	0.1729
	Undirected	0.1921 *	0.1811	**0.1684**	**0.1476**	**0.1228**
	Shortest	0.2220	0.1992	0.1934	0.1737	0.1585
	Undirected and shortest	**0.1911** *	**0.1774**	0.1708	0.1537	0.1303
	Baseline (linear)	0.3070	0.3070	0.3070	0.3070	0.3070

Table 6. Summary of evaluation of xDAISTOUT. Comparison of the xDAISTOUT graph variations across different graph sizes using SPD, HD, and DFD means. Matching coloured asterisks indicate non-significant differences between variants at the same graph size, whereas matching coloured crosses indicate non-significant differences between graph sizes for the same variant ($p_{adj} \geq 0.05$). Bold indicates the best performance under a metric for a given number of nodes (column-wise).

Metric	Variation	100 k	120 k	140 k	160 k	180 k
SPD	Original	0.0385	0.0370	0.0356	0.0347	0.0336
	Undirected	**0.0318**	**0.0308**	**0.0292**	**0.0284** *	**0.0278**
	Shortest	0.0368	0.0352	0.0336	+0.0326	+0.0327
	Undirected and shortest	+0.0326	+0.0325	0.0313	0.0286 *	0.0285
	Baseline (linear)	0.0350	0.0350	0.0350	0.0350	0.0350
DFD	Original	0.0050	0.0048	0.0046	0.0044	0.0042
	Undirected	**0.0038**	**0.0038**	**0.0036**	**0.0034**	**0.0034**
	Shortest	0.0046	+0.0044	+0.0043	+0.0040	+0.0040
	Undirected and shortest	0.0068	0.0081	0.0079	**0.0034**	**0.0034**
	Baseline (linear)	0.0491	0.0491	0.0491	0.0491	0.0491
HD	Original	0.3924	0.3647	0.3313	0.2899	0.2630
	Undirected	0.1937	0.1843	0.1739 *	**0.1533**	**0.1498**
	Shortest	0.3531	0.3326	0.3036	0.2576	0.2264
	Undirected and shortest	**0.1913**	**0.1786**	**0.1726** *	0.1555	0.1538
	Baseline (linear)	0.3070	0.3070	0.3070	0.3070	0.3070

4.3.1. Effect of Sampling Level on the Main Graph-Based Methods

The row-wise comparisons in Tables 4–6 present the effect of increasing the number of nodes for DAISTIN, xDAISTIN, and xDAISTOUT graphs. Consistent with the interpolation completion rate results, increasing the number of graph nodes generally reduced the reconstruction errors under the three metrics. The most consistent results were observed when comparing HDs; every increase between adjacent graph sizes produced a statistically significant reduction in mean HD for every evaluated method and variant. The original DAISTIN also exhibited this consistent reduction in SPD between every pair of adjacent graph sizes. For undirected DAISTIN, however, no significant SPD differences were detected between the 140 k, 160 k, and 180 k node graphs. Based on DFD, the original DAISTIN configurations with 100 k and 120 k nodes did not differ significantly, and no significant differences were detected among the 100 k, 120 k, and 160 k undirected configurations. In relation to DFD, the results from the undirected DAISTIN were notably non-monotonic: the lowest mean DFD was obtained at 140 k nodes, whereas increasing the graph size from 140 k to 160 k and from 160 k to 180 k resulted in statistically significant increases in DFD. This occurred despite the corresponding increase in interpolation completion rate (see Table 3; for ease of analysis, these are 85.46%, 89.36% and 91.23%, respectively, for the 140 k, 160 k, and 180 k node graphs).

For xDAISTIN, the SPD decreased significantly with every adjacent increase in graph size for all four variants, with the sole non-significant graph-size comparison occurring between 100 k and 120 k for the undirected-and-shortest variant. Across all pairwise graph-size comparisons, the only additional non-significant result was for DFD between the 100 k and 140 k shortest variants. In addition, the DFD was non-monotonic for the shortest-path variants: increasing the graph size from 100 k to 120 k significantly increased the DFD for both the shortest and undirected-and-shortest variants, after which, for larger graphs (140 k, 160 k, 180 k), the DFD decreased significantly. For xDAISTOUT, under SPD, no significant differences were detected between the 160 k and 180 k shortest variant, and between the 100 k and 120 k undirected-and-shortest variant. Under DFD, the xDAISTOUT shortest variants did not differ significantly between 120 k and 140 k and between 160 k and 180 k nodes. Consequently, denser graphs consistently improved the HD and generally improved the SPD, whereas their effect on the DFD was not as clear and it was method and variant dependent.

4.3.2. Effect of Graph-Based Method

The column-wise comparisons in Tables 4–6 present the effect of the graph variants at each fixed graph size, with asterisks denoting the pairs with no significant difference. These comparisons were conducted separately within the DAISTIN, xDAISTIN, and xDAISTOUT families; consequently, the statistical results below concern differences among variants of the same method family.

For DAISTIN, the undirected variant achieved a significantly lower mean HD than the original directed variant at every evaluated graph size. Under SPD, the undirected variant also achieved significantly lower mean errors from 100 k to 160 k nodes. At 180 k nodes, original DAISTIN had a numerically lower mean SPD, although the difference was not statistically significant ($p_{adj} = 0.067$). Under DFD, undirected DAISTIN achieved significantly lower mean errors at 100 k, 120 k, and 140 k nodes. No significant difference was detected at 160 k nodes ($p_{adj} = 0.096$), whereas at 180 k nodes the direction was reversed, with original DAISTIN achieving a significantly lower mean DFD. Thus, for DAISTIN, removing directionality consistently improved the HD, generally improved the SPD, but had a graph-size-dependent effect on the DFD.

Within the xDAISTIN family, the undirected variant achieved the lowest mean SPD at every graph size and was significantly better than each of the other three variants in all five graph sizes. For HD, the undirected-and-shortest variant achieved the lowest mean at 100 k and 120 k nodes. At 100 k nodes, however, it did not differ significantly from the undirected variant, whereas at 120 k nodes its lower mean HD was statistically significant. From 140 k to 180 k nodes, the undirected variant achieved the lowest mean HD and was significantly better than the remaining variants. The xDAISTIN DFD results showed a different pattern. The undirected variant achieved the lowest mean DFD and was significantly better than all other variants at 100 k, 120 k, and 140 k nodes. At 160 k nodes, the shortest variant instead achieved the lowest mean DFD and was significantly better than the other three variants. At 180 k nodes, the shortest variant again had the lowest numerical mean, but it did not differ significantly from either the undirected or the undirected-and-shortest variant.

Within the xDAISTOUT family, the undirected variant consistently achieved the lowest mean SPD. It was significantly better than all other variants at 100 k, 120 k, 140 k, and 180 k nodes. At 160 k nodes, no significant difference was detected between the undirected and undirected-and-shortest variants, although both significantly outperformed the original and shortest variants. For xDAISTOUT under DFD, the undirected variant achieved the lowest mean error and was significantly better than the other variants from 100 k to 140 k nodes. At 160 k and 180 k nodes, the undirected-and-shortest variant instead achieved the lowest mean DFD and was significantly better than all remaining variants. Under HD, the undirected-and-shortest variant achieved the lowest mean at 100 k and 120 k nodes. At 140 k nodes, it did not differ significantly from the undirected variant. At 160 k and 180 k nodes, the undirected variant achieved the lowest mean HD and was significantly better than all other variants.

Overall, removing graph directionality provided the most consistent improvements, particularly for SPD and HD. The shortest-path variants produced more graph-size- and metric-dependent effects.

4.3.3. Comparison of Selected DAISTIN, xDAISTIN, and xDAISTOUT Configurations

Based on the observations from the preceding analyses, four graph configurations were selected for a focused comparison: directed DAISTIN with 160 k nodes, undirected DAISTIN with 180 k nodes, undirected xDAISTIN with 180 k nodes, and undirected xDAISTOUT with 180 k nodes. Their mean SPD, DFD, and HD values are presented in Table 7. The Friedman repeated-measures tests identified significant overall differences among the four selected

configurations for SPD, DFD, and HD ($p < 0.001$). Consequently, pairwise comparisons were conducted using two-sided paired-sample t-tests with Holm–Bonferroni correction. Matching red crosses identify configuration pairs for which no statistically significant difference was detected using the Holm-adjusted paired-sample t-test.

Table 7. Summary of selected graph configurations. Comparison of selected DAISTIN, xDAISTIN, and xDAISTOUT configurations using SPD, DFD, and HD means. Matching coloured crosses indicate pairs of configurations with no statistically significant difference for the corresponding metric ($p_{\text{adj}} \geq 0.05$). Bold values indicate the lowest mean error for each metric.

Metric	DAISTIN Directed, 160 k	DAISTIN Undirected, 180 k	xDAISTIN Undirected, 180 k	xDAISTOUT Undirected, 180 k
SPD	+0.0311	+0.0302	**0.0258**	0.0278
DFD	+0.0052	+0.0055	**0.0031**	0.0034
HD	0.2767	0.1919	**0.1228**	0.1498

Overall, undirected xDAISTIN with 180 k nodes provided the strongest reconstruction performance among the four selected configurations, achieving significantly lower mean errors under all three metrics. Undirected xDAISTOUT with 180 k nodes consistently ranked second, while the relative ordering of the two DAISTIN configurations was metric dependent.

4.4. Empirical Runtime and Graph Size

We measured the empirical runtimes of graph construction and graph-based interpolation. In Table 8, we report and compare the graph construction runtimes between 100 k and 180 k node graphs of DAISTIN, xDAISTIN, and xDAISTOUT. We also report the time needed to convert the graphs into their undirected variants. Table 9 reports online interpolation runtime statistics of the main methods and their undirected variants at the 100 k and the 180 k node graph size (mean, median, std, max) over a random sample of 1000 gaps.

Table 8. Offline graph construction runtime. Directed build time denotes full graph construction from the loaded sampled nodes for DAISTIN, xDAISTIN, and xDAISTOUT. Undirected build time denotes the additional time required to derive the undirected graph from the already constructed directed graph.

Graph Family	Nodes	Directed Build Time (s)	Undirected Build Time (s)
DAISTIN	100 k	18.02	2.07
DAISTIN	180 k	47.65	6.28
xDAISTIN	100 k	575.51	10.90
xDAISTIN	180 k	2477.60	42.78
xDAISTOUT	100 k	578.24	8.76
xDAISTOUT	180 k	2540.86	38.69

Table 9. Online interpolation runtime for the main graph-based methods. Mean, median, standard deviation (Std.), and max times are computed at the gap-level over a random sample of 1000 gaps.

Method	Nodes	Mean Time (s)	Median Time (s)	Std. Time (s)	Max Time (s)
DAISTIN	100 k	0.168	0.156	0.034	0.353
DAISTIN	180 k	0.342	0.316	0.101	1.736
DAISTIN undirected	100 k	0.187	0.171	0.051	0.492
DAISTIN undirected	180 k	0.392	0.343	0.196	2.880
xDAISTIN	100 k	0.540	0.424	0.361	4.413
xDAISTIN	180 k	1.333	0.968	1.070	13.341
xDAISTIN undirected	100 k	0.660	0.532	0.437	4.713
xDAISTIN undirected	180 k	1.923	1.476	1.463	13.196
xDAISTOUT	100 k	0.399	0.351	0.178	1.505
xDAISTOUT	180 k	1.041	0.820	0.669	5.289
xDAISTOUT undirected	100 k	0.662	0.535	0.448	4.844
xDAISTOUT undirected	180 k	1.814	1.431	1.310	11.186

4.5. Qualitative Evaluation

To perform qualitative evaluation of reconstructions we visualized the distribution of the test set trajectory gaps in terms of displacement between the two points defining the gap, and created 20 bins with similar density. We then sampled 12 random gaps from each bin, and saved interpolations by the four selected graphs (DAISTIN directed with 160 k nodes, DAISTIN undirected with 180 k, xDAISTIN undirected 180 k, and xDAISTOUT undirected 180 k) and baseline (linear interpolation) on a total of 240 cases.

Based on visual examination, the first observation that can be made concerns the nature of the reconstructed trajectories as produced by the DAISTIN graphs (both directed and undirected). More often than not, cases in which the reconstructions of the DAISTIN and DAISTIN undirected graphs are superior to the xDAISTIN graphs are ones whose ground truth favours simpler trajectories (e.g., a linear reconstruction or one with fewer intermediate points). Such examples are shown in rows A for DAISTIN undirected and B for DAISTIN directed of Figure 10. We note that the majority of cases where DAISTIN graphs surpass our proposed methods fall under this category. However, in Figure 10 we chose to illustrate more interesting examples, without claiming that the examples shown are representative of the data set. Rows C and D demonstrate cases where directed edges enabled DAISTIN (160 k) to reconstruct a fairly accurate trajectory following the right side of the lane. Rows E and F present cases for which DAISTIN undirected and directed, respectively, provided reconstructions more aligned to the ground truth trajectories.

The second observation concerns our proposed methods, xDAISTIN undirected (180 k) and xDAISTOUT undirected (180 k): we were able to identify different port areas in which each method performs better. This can be observed by comparing Figures 11 and 12. Specifically, the two methods appear to perform best in different areas of the port: one in narrow navigational corridors (xDAISTOUT), and the other in confined inner-port waters, such as terminal or berthing areas (xDAISTIN). This observation is exploratory and should not be interpreted as evidence of method specialization.

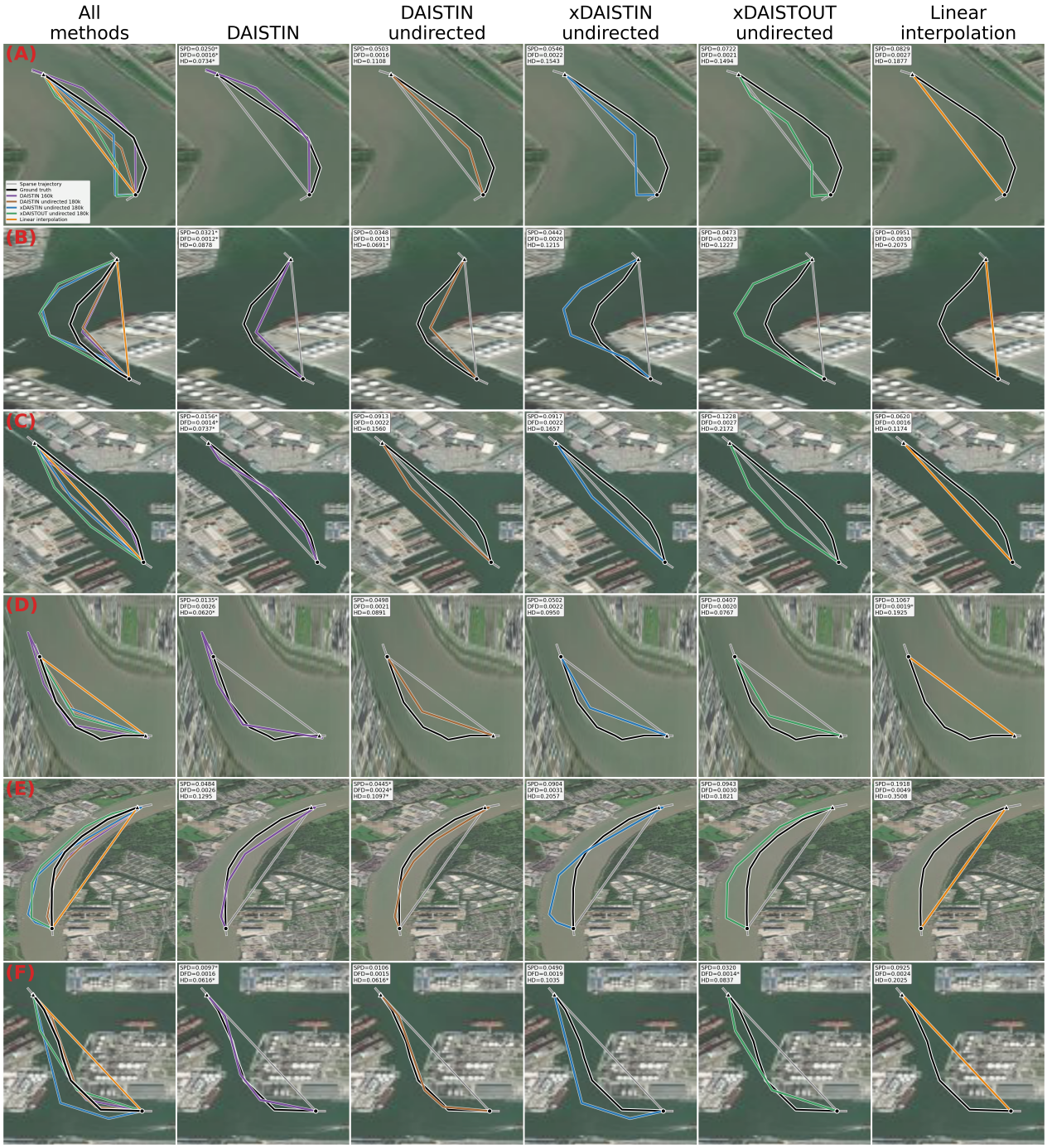


Figure 10. Qualitative examination of gap interpolation by DAISTIN variants. Selected cases (**A**–**F**) where gap interpolations are performed best by the DAISTIN variants according to SPD. Columns, in order, show the gap interpolation by all methods: DAISTIN (160 k), undirected DAISTIN (180 k), undirected xDAISTIN (180 k), undirected xDAISTOUT (180 k), and linear interpolation. Black trajectories denote ground truth and grey denotes the sparse observed trajectory. Asterisks denote the lowest reconstruction error for each metric within each case (row-wise).

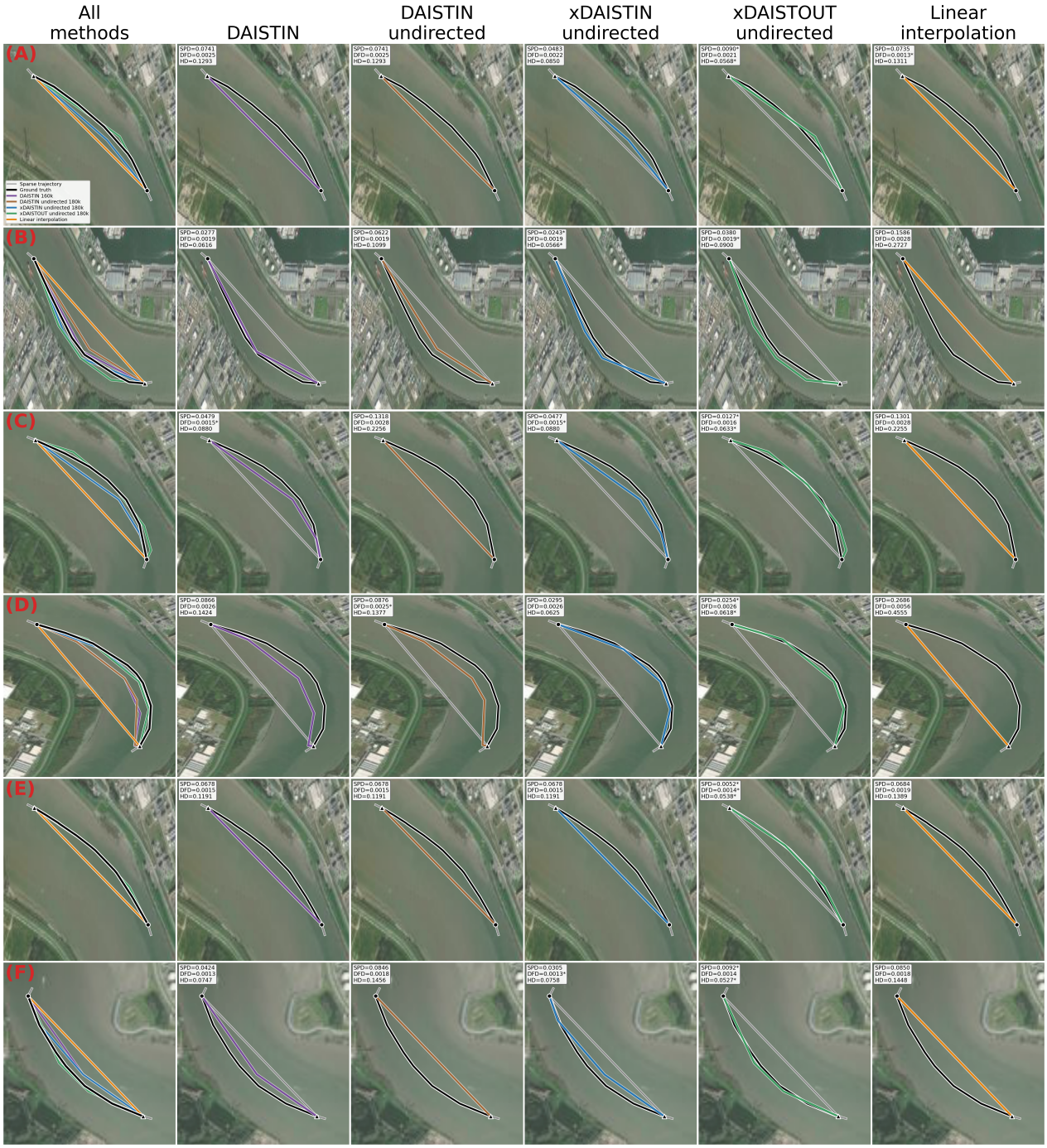


Figure 11. Qualitative examination of gap interpolation in corridors. Selected cases (**A**–**F**) where the gap is located in a corridor, with the xDAISTOUT undirected graph performing the best in most of them according to SPD. Columns, in order, show the gap interpolation by all methods: DAISTIN (160 k), undirected DAISTIN (180 k), undirected xDAISTIN (180 k), undirected xDAISTOUT (180 k), and linear interpolation. Black trajectories denote ground truth and grey denotes the sparse observed trajectory. Asterisks denote the lowest reconstruction error for each metric within each case (row-wise).

Finally, our qualitative analysis also exposed limitations of both the methods and of the ground truth in a few instances. In Figure 13, we present selected cases of DAISTIN artefacts (A–C), an xDAISTIN and xDAISTOUT artefact (D), a gap where all methods struggle (E), a gap favouring linear interpolation (F), and cases where the ground truth is insufficiently detailed (G,H).

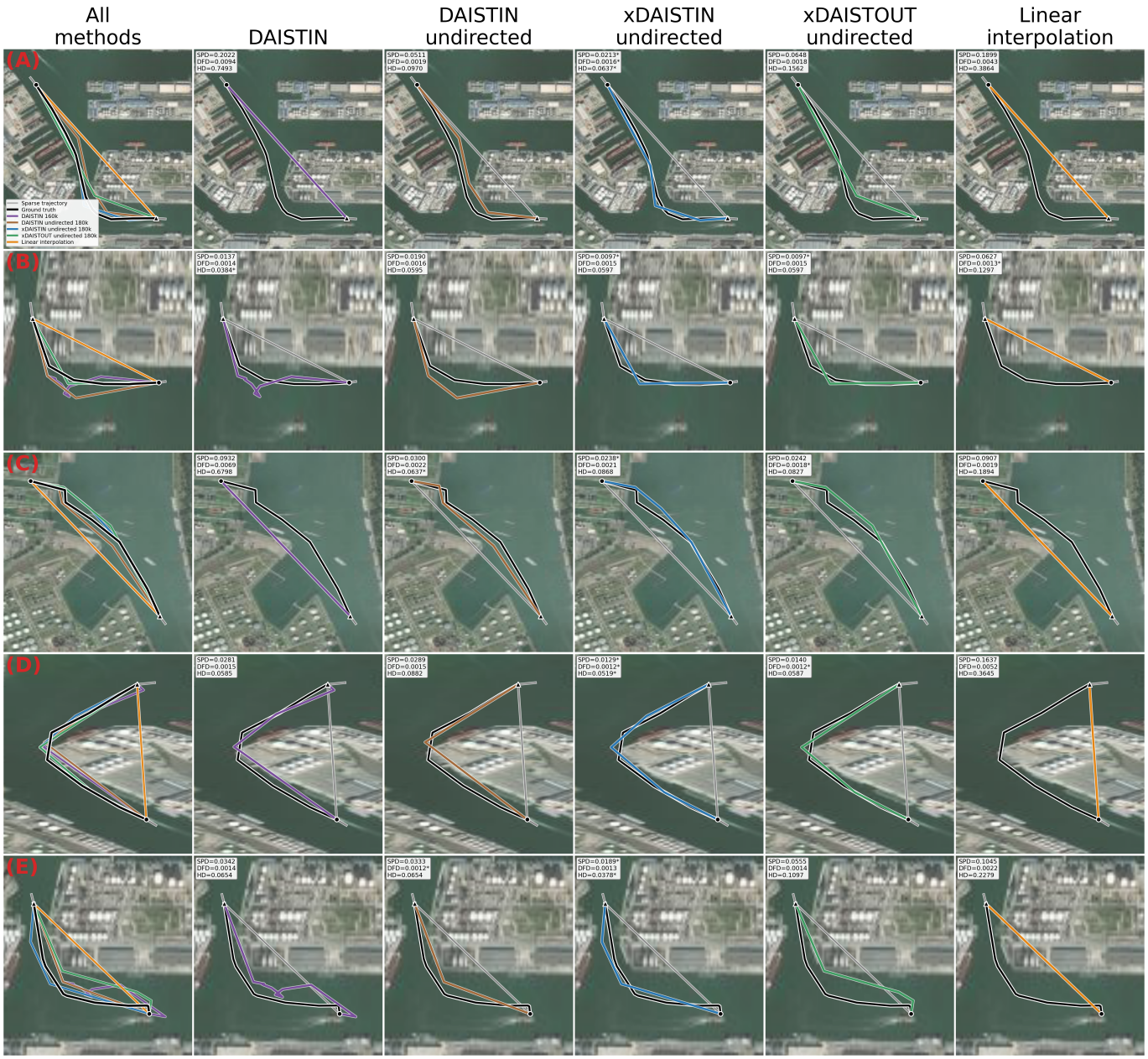


Figure 12. Qualitative examination of gap interpolation in confined inner-port waters. Selected cases (**A**–**E**) where gap interpolations are located in confined inner-port waters such as berths and terminals, with the xDAISTIN undirected graph performing the best in most of them according to SPD. Columns, in order, show the gap interpolation by all methods: DAISTIN (160 k), undirected DAISTIN (180 k), undirected xDAISTIN (180 k), undirected xDAISTOUT (180 k), and linear interpolation. Black trajectories denote ground truth and grey denotes the sparse observed trajectory. Asterisks denote the lowest reconstruction error for each metric within each case (row-wise).

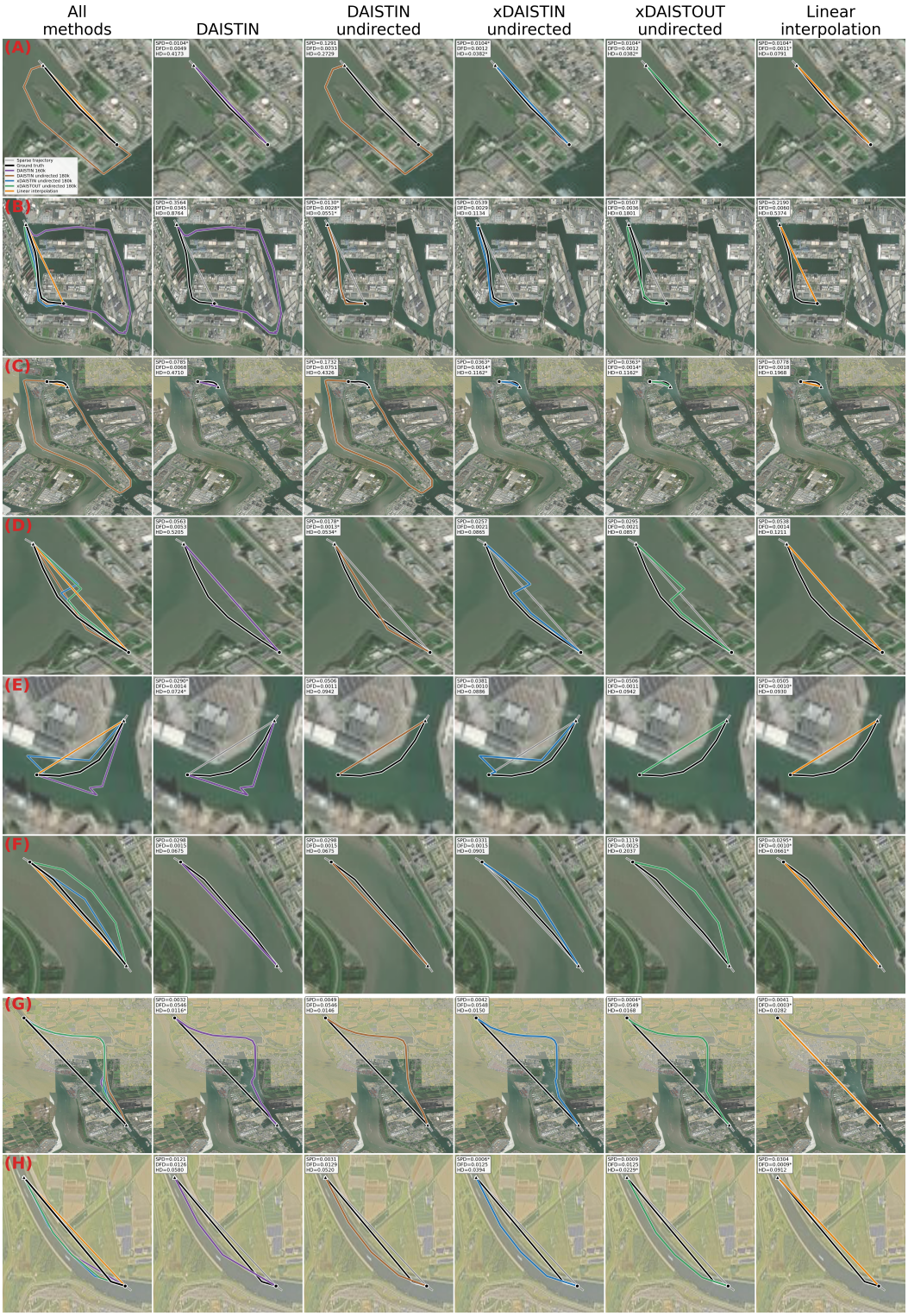


Figure 13. Reconstruction failures, artefacts, and incomplete ground truth cases. Selected cases illustrating directed and undirected DAISTIN artefacts (**A**–**C**), xDAISTIN and xDAISTOUT movement artefact (**D**), a gap where all methods struggle (**E**), a case favouring linear interpolation (**F**), and two cases with insufficiently detailed ground truth (**G**,**H**). Columns, in order, show the gap interpolation by all methods: DAISTIN (160 k), undirected DAISTIN (180 k), undirected xDAISTIN (180 k), undirected xDAISTOUT (180 k), and linear interpolation. Black trajectories denote ground truth and grey trajectories denote the sparse observed trajectory. Asterisks denote the lowest reconstruction error for each metric within each case (row-wise).

5. Discussion

The results support the central hypothesis of our work: directional information discarded during geometric sampling is consequential for graph-based trajectory reconstruction in ports. Enriching sampled nodes with local circular heading distributions improved the interpolation completion rate at every evaluated graph size and reduced reconstruction errors. Most notably, directed xDAISTIN with 100 k nodes reconstructed 88.02% of the introduced gaps, exceeding the 84.61% achieved by directed DAISTIN with 180 k nodes. In the focused comparison, undirected xDAISTIN with 180 k nodes achieved the strongest performance, reducing the mean SPD by approximately 14%, mean DFD by approximately 43%, and mean HD by approximately 36% relative to undirected DAISTIN at the same graph size. These findings suggest that preserving neighbourhood-level directional information produces graphs that are both better connected and represent real-world movement patterns more effectively.

Removing graph directionality generally improved both the interpolation completion rate and reconstruction accuracy. Importantly, the undirected variants do not discard directional information during graph construction: edges are first selected using the direction criterion of the corresponding method and are subsequently made traversable in both directions. This suggests that directional information is useful for identifying plausible connections, whereas enforcing a single traversal direction can make the graph overly restrictive during interpolation.

The core difference between xDAISTIN and xDAISTOUT is that xDAISTIN evaluates a candidate transition using the destination node's heading distribution, whereas xDAISTOUT evaluates it using the source node's distribution. In the focused comparison, undirected xDAISTIN achieved significantly lower errors than undirected xDAISTOUT under all three metrics. This suggests that assessing whether a candidate location is normally approached along the proposed bearing may be more informative than assessing whether the bearing is common when leaving the current location. Nevertheless, the qualitative results suggest that the methods may have complementary strengths in different port areas, an observation that requires further empirical evaluation.

The qualitative analysis broadly supports the quantitative findings. Linear interpolation remained effective for simple and approximately straight gaps, but fails when there is any deviation from a straight movement. DAISTIN and its undirected variant reconstructed some nontrivial movements more successfully than the proposed methods. However, most cases in which DAISTIN performed better involved simple, approximately linear trajectories or trajectories with few intermediate observations. Both DAISTIN variants also occasionally produced substantial detours. Overall, the proposed methods, xDAISTIN and xDAISTOUT, produced trajectories that were more closely aligned with the ground truth, including in corridors and inner-port areas. Nevertheless, artefacts remained, and in some cases the ground truth was too sparse to reliably determine which graph reconstruction was most accurate.

Several limitations should be acknowledged. First, the evaluation was conducted using AIS data from a single port and over a relatively short collection period; additional ports are needed to assess generalization across different layouts and traffic patterns. Second, the evaluation relied on synthetically introduced gaps. Although this provides ground truth from real-world data, real AIS losses may follow different temporal and spatial patterns and would require an independent reference source, such as radar or satellite remote sensing data [32], for quantitative evaluation. Finally, the methods were assessed using a fixed hyperparameter configuration and only on moving sub-trajectories.

6. Conclusions

This work introduced xDAISTIN and xDAISTOUT, two extensions of DAISTIN for reconstructing incomplete AIS trajectories in complex port environments. Both methods preserve directional information that would otherwise be lost during geometric sampling by enriching sampled graph nodes with local circular heading distributions estimated from nearby historical AIS observations. xDAISTIN evaluates candidate transitions using the destination node's distribution, whereas xDAISTOUT uses the source node's distribution.

Experiments on real-world AIS data from the Port of Antwerp, using synthetic gaps introduced into held-out trajectories, showed that this enrichment substantially improves the interpolation completion rate. Directed xDAISTIN with 100 k nodes reconstructed 88.02% of the introduced gaps, exceeding the 84.61% achieved by directed DAISTIN with 180 k nodes. At 180 k nodes, the undirected-and-shortest path xDAISTIN variant achieved the highest interpolation completion rate, at 99.13%, whereas undirected probabilistic xDAISTIN provided the strongest reconstruction accuracy in the focused comparison. It achieved mean SPD, DFD, and HD values of 0.0258, 0.0031, and 0.1228, respectively, and significantly outperformed the selected DAISTIN and xDAISTOUT configurations under all three metrics. Relative to undirected DAISTIN at the same graph size, it reduced mean SPD by approximately 14%, mean DFD by approximately 43%, and mean HD by approximately 36%.

In our experiments, shortest-path graph traversal slightly improved the interpolation completion rate and sometimes reduced worst-case deviation for smaller graphs, whereas the probabilistic approach generally produced lower average reconstruction error and performed better as graph density increased. Future work should examine computational optimizations, multi-port validation, real AIS losses supported by independent ground truth, low-speed manoeuvres (<1 knot), hyperparameter sensitivity, and hybrid criteria combining source- and destination-side directional information, i.e., combining the edge construction criteria of xDAISTIN and xDAISTOUT. Overall, our findings demonstrate that preserving neighbourhood-level directional behaviour during graph construction enables more accurate reconstruction of in-port AIS trajectories. Furthermore, our cautious hypothesis is that shortest-path traversal may be preferable when it is desirable to avoid large deviations and maximize the interpolation completion rate, especially for relatively sparse graphs, whereas the probabilistic approach appears preferable when greater importance is placed on the average accuracy of the reconstructed trajectory and the graph is sufficiently dense.

Author Contributions: Conceptualization, E.Z. and N.D.; methodology, E.Z. and N.D.; software, E.Z. and N.D.; validation, E.Z., N.D. and O.A.; formal analysis, E.Z.; investigation, E.Z., N.D. and O.A.; data curation, E.Z.; writing—original draft preparation, E.Z., N.D. and O.A.; writing—review and editing, N.D. and O.A.; visualization, E.Z. and N.D.; supervision, N.D.; project administration, N.D.; funding acquisition, N.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was co-financed by the European Union—NextGenerationEU, through the Research and Innovation Foundation with grant numbers CODEVELOP-GT/0322/0096 (ADAPTATION) and the EU H2020 Research and Innovation Programmes under Grant Agreements No. 857586 (CMMI-MaRITeC-X).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Research data for this article are made available via Mendeley Data (DOI: 10.17632/smb56stcdx.1) at https://data.mendeley.com/datasets/smb56stcdx/1 (accessed on 5 August 2026).

Acknowledgments: Our work would not be possible without the support of our colleague Charalambos Rotsides, who was instrumental in the efficient collection, processing, and storage of the data. During the preparation of this manuscript, the authors used OpenAI's ChatGPT (model: GPT-5.6 Sol) for language editing assistance. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AIS	Automatic identification system
AND	Approximate neighbourhood density
CH	Calculated heading
COG	Course over ground
DAISTIN	Data-driven AIS Trajectory Interpolation
DFD	Discrete Fréchet distance
FD	Fréchet distance
HD	Hausdorff distance
KDE	Kernel density estimator
MMSI	Maritime Mobile Service Identity
SOG	Speed over ground
SPD	Segmented path distance
xDAISTIN	Extended Data-driven AIS Trajectory Interpolation with Destination-node Directional Evaluation
xDAISTOUT	Extended Data-driven AIS Trajectory Interpolation with Source-node Directional Evaluation

References

1. International Maritime Organization. *Resolution A.1106(29): Revised Guidelines for the Onboard Operational Use of Shipborne Automatic Identification Systems (AIS)*; International Maritime Organization: London, UK, 2015.
2. Liang, M.; Su, J.; Liu, R.W.; Lam, J.S.L. AISClean: AIS data-driven vessel trajectory reconstruction under uncertain conditions. *Ocean. Eng.* **2024**, *306*, 117987. [CrossRef]
3. MIS Marine. *The Use of AIS Data to Identify Dark Activity for Marine Auditing Purposes*; MIS Marine: Birmingham, UK, 2020.
4. Magnussen, B.B.; Bläser, N.; Lu, H. DAISTIN: A Data-Driven AIS Trajectory Interpolation Method. In *Proceedings of the 18th International Symposium on Spatial and Temporal Data, Calgary, AB, Canada, 23–25 August 2023*; Association for Computing Machinery: New York, NY, USA, 2023; pp. 75–84. [CrossRef]
5. Nguyen, V.S.; Im, N.K.; Lee, S.M. The interpolation method for the missing AIS data of ship. *J. Navig. Port Res.* **2015**, *39*, 377–384. [CrossRef]
6. Sang, L.Z.; Wall, A.; Mao, Z.; Yan, X.P.; Wang, J. A novel method for restoring the trajectory of the inland waterway ship by using AIS data. *Ocean. Eng.* **2015**, *110*, 183–194. [CrossRef]
7. Yuan, Z.; Liu, J.; Liu, Y.; Li, Z. A novel approach for vessel trajectory reconstruction using AIS data. In *Proceedings of the ISOPE International Ocean and Polar Engineering Conference, Honolulu, HI, USA, 16–21 June 2019*; ISOPE: Cupertino, CA, USA, 2019; pp. 4554–4559.
8. Bai, X.; Fang, M.; Xu, X.; Xiao, Y.; Wu, Y.; Chen, N. Research on Ship AIS Trajectory Reconstruction for Narrow-Mouth Section Waters under Complex Environment. *Navig. China* **2024**, *47*, 106–113. [CrossRef]
9. Ye, L.; Chen, X.; Liu, H.; Zhang, R.; Li, J.; Lu, C.; Zhao, Y. A Study of Multi-Step Sparse Vessel Trajectory Restoration Based on Feature Correlation. *Appl. Sci.* **2024**, *14*, 4057. [CrossRef]
10. Galdelli, A.; Narang, G.; Tomassini, S.; D'Agostino, L.; Tassetti, A.N.; Mancini, A. Data Imputation in Large and Small-Scale Spatiotemporal Time Series Gaps Using BackForward Bi-LSTM. *J. Big Data* **2025**, *12*, 115. [CrossRef]
11. Gao, Z.; Gao, J. Research on Ship Trajectory Repair Algorithm Based on AIS Data. In *Proceedings of the 4th International Conference on Computer, Artificial Intelligence and Control Engineering, Hefei, China, 10–12 January 2025*; Association for Computing Machinery: New York, NY, USA, 2025. [CrossRef]

12. Han, X.; Bai, S.; Issayeva, G.; Ma, C.; Li, F.; Shang, X. STAR: Spatio-Temporal Trajectory Recovery for Sparse and Uncertain Marine Trajectories. In *Proceedings of the Database Systems for Advanced Applications*; Lecture Notes in Computer Science; Zhu, F., Yu, P.S., Nadamoto, A., Lim, E.P., Shim, K., Ding, W., Zhang, B., Eds.; Springer: Singapore, 2026; Volume 15991, pp. 482–486. [CrossRef]
13. Arandjelović, O. Contextually learnt detection of unusual motion-based behaviour in crowded public spaces. In *Proceedings of the Computer and Information Sciences II: 26th International Symposium on Computer and Information Sciences*; Springer: London, UK, 2011; pp. 403–410.
14. Dimitriou, N. Historical Knowledge Graphs for Global Maritime Estimated Time of Arrival. *arXiv* **2026**, arXiv:2605.18408. [CrossRef]
15. Filipiak, D.; Węcel, K.; Stróżyna, M.; Michalak, M.; Abramowicz, W. Extracting maritime traffic networks from AIS data using evolutionary algorithm. *Bus. Inf. Syst. Eng.* **2020**, *62*, 435–450. [CrossRef]
16. Onyango, S.O.; Owiredu, S.A.; Kim, K.I.; Yoo, S.L. A quasi-intelligent maritime route extraction from AIS data. *Sensors* **2022**, *22*, 8639. [CrossRef] [PubMed]
17. Hadjipieris, A.; Dimitriou, N.; Arandjelović, O. Unsupervised Port Berth Localization from Automatic Identification System Data. *Sensors* **2025**, *25*, 6845. [CrossRef] [PubMed]
18. Saalfeld, A. Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm. *Cartogr. Geogr. Inf. Sci.* **1999**, *26*, 7–18. [CrossRef]
19. Hagberg, A.A.; Schult, D.A.; Swart, P.J. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*; Varoquaux, G., Vaught, T., Millman, J., Eds.; SciPy Proceedings Organization: Pasadena, CA, USA, 2008; pp. 11–15.
20. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
21. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef] [PubMed]
22. Gillies, S. Shapely: Python Package for Manipulation and Analysis of Geometric Objects. 2007. Available online: https://github.com/Toblerity/Shapely (accessed on 5 August 2026).
23. Hirschmann, F. Ramer-Douglas-Peucker Algorithm (rdp), Python Package. Available online: https://pypi.org/project/rdp/ (accessed on 5 August 2026).
24. McKinney, W. Data structures for statistical computing in Python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; Volume 445, pp. 51–56.
25. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
26. Arribas-Bel, D. Contextily: Python Package for Context Geo Tiles. 2023. Available online: https://contextily.readthedocs.io (accessed on 5 August 2026).
27. Story, R. Folium, Python Package. 2025. Available online: https://pypi.org/project/folium/ (accessed on 5 August 2026).
28. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* **2020**, *17*, 261–272. [CrossRef] [PubMed]
29. Waskom, M.L. Seaborn: Statistical data visualization. *J. Open Source Softw.* **2021**, *6*, 3021. [CrossRef]
30. Besse, P.; Guillouet, B.; Loubes, J.M.; Royer, F. Review & Perspective for Distance Based Trajectory Clustering. *arXiv* **2015**, arXiv:1508.04904.
31. Eiter, T.; Mannila, H. *Computing Discrete Fréchet Distance*; Technical Report CD-TR 94/64; Technical University of Vienna: Vienna, Austria, 1994.
32. Belmouhcine, A.; Pham, M.T.; Hadjisoteriou, A.; Dimitriou, N. Upsampling and Knowledge Distillation Improve Small-Object Detection in Satellite Remote Sensing. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2026**, *19*, 16607–16622. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.