

SUMMARY

This paper addresses the challenge of missing position reports within vessel trajectories. The position reports are continuously transmitted by the Automatic Identification System (AIS) and describe the movement of a vessel, detailing critical information such as the vessel's current position, course, speed, and estimated time of arrival. AIS serves as the foundation for a multitude of applications and research endeavors and plays a critical role in enhancing maritime safety and operational efficiency, such as collision avoidance mechanisms.

In this paper, we present DGIVT (Depth-Map Enhanced Graph Imputation for Vessel Trajectories), a framework for performing imputation on vessel trajectories, to minimize the number of missing position reports. We utilize the freely available AIS data provided by the Danish Maritime Authority to extract a year's worth of position reports, which we refer to as vessel samples in this paper. Employing a grid across the Danish maritime waters, we assign each vessel sample to a corresponding grid cell, we denote this as the vessel sample grid. Subsequently, we employ a density sampling algorithm to minimize the number of vessel samples within each cell while preserving a spatial representation of the vessel samples. For each cell, we construct a directed graph using the vessel samples as vertices and connect them to other vertices within a distance that also adheres to the certain course over ground criteria. To accommodate various vessel types, and ensure that the imputation does not render vessel samples in inaccessible regions of the vessel, we assign a depth to each vertex. This depth is determined by referencing a depth grid generated from a depth map. In areas lacking depth data, we rely on the draught of the vessel sample. Finally, we ensure connectivity by linking graphs for each cell with those in adjacent cells.

Following the graph creation, we proceed to imputation on vessel trajectories, where we begin by identifying the cells in the vessel sample grid, traversed by the vessel trajectory, and extracting the cells graphs. We then analyze consecutive vessel samples within the vessel trajectory, namely a source and a destination vessel sample. We proceed by identifying vertices within a specified distance from both the source and destination vessel samples. For each, we establish edges that adhere to criteria including distance, course over ground, and depth. After the edges are defined, we identify all vertices between the source and destination vessel samples, eliminating those that do not meet the specified thresholds. We then utilize an A* algorithm to navigate the paths between the source and destination vessel samples, and add the path to the imputed vessel trajectory, and continue to the next two consecutive vessel samples. We do this iteratively until all vessel samples are traversed, yielding the complete imputed trajectory. Due to the placement of vessel samples within the graph, imputed trajectories may exhibit a zigzag pattern. To address this, we employ a refinement algorithm that solves a linear matrix equation and calculates the least squares solution. This allows us to determine whether a vessel is traveling straight or turning.

We evaluate our framework within a defined region consisting of nine cells. Utilizing approximately, 1450 random vessel trajectories, we apply three distinct reduction methods. These include multiple gap reduction, where we minimize the trajectory to maintain a minimum distance between each vessel sample; single gap reduction, involving the introduction of a single gap within the trajectory; and realistic frequency reduction, aimed at simulating scenarios where expected vessel samples are missing.

Our results indicate that we perform well in contrast to linear imputation and the framework GTI on trajectories with multiple and single gaps. However, there remains potential for improvement, particularly in terms of the number of vessel positions and achieving a closer resemblance to the original trajectory shape.

DGIVT: Depth-Map Enhanced Graph Imputation for Vessel Trajectories

Alex Farup Christensen* and Cecilie Merete Welling Fog†

Computer Science Department, Aalborg University,
Aalborg, Denmark

Emails: {afch19*, cfog19†}@student.aau.dk

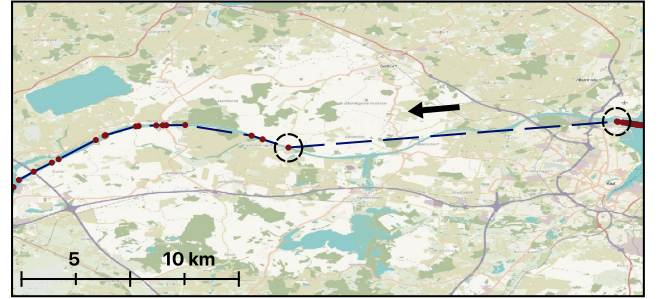
Abstract—This paper presents the Depth-Map Enhanced Graph Imputation for Vessel Trajectories (DGIVT), a novel framework designed to address the challenge of missing AIS (Automatic Identification System) position reports in maritime navigation. Utilizing historical AIS position reports and sea depth map, DGIVT constructs directed graphs that facilitate accurate imputation of vessel trajectories through an A* algorithm, ensuring realistic pathfinding that considers vessel-specific constraints such as depth. The approach distinguishes itself by not relying on pre-established maritime routes, thus providing a flexible and comprehensive solution for a variety of vessel types across different maritime contexts. Our evaluation demonstrates the framework’s effectiveness in enhancing the integrity of vessel trajectories, which is crucial for improving maritime safety and operational efficiency. Results highlight substantial improvements in trajectory estimation over traditional methods, thereby supporting the critical role of accurate data in maritime operations.

1. INTRODUCTION

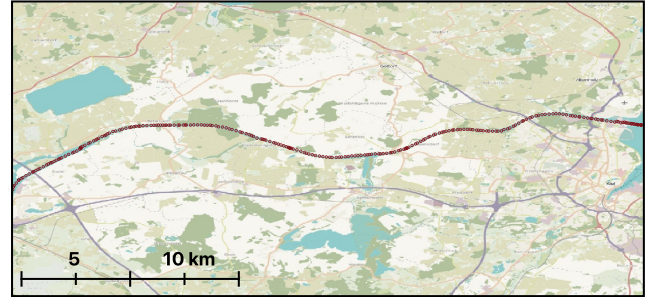
Ocean shipping is responsible for transporting approximately 80% of all traded commodities worldwide across the seas, playing a vital role in global trading [1, p. 55]. Given the vast scale of ocean shipping, a significant technological advancement in maritime navigation is the Automatic Identification System (AIS), designed primarily to enhance vessel identification and location tracking at sea [2, p. 5]. Mandated by the International Maritime Organization (IMO) in 2004 to ensure the Safety of Life at Sea, the use of AIS is compulsory for international passenger- and cargo vessels over 300 gross tonnage. AIS operates autonomously, incorporating both transceiver systems and onboard navigational aids such as Global Positioning System (GPS) receivers, gyrocompasses, and rate-of-turn indicators [3]. Position reports based on a vessel’s movements are continuously transmitted by AIS and received by maritime authorities and nearby vessels. The position reports detail critical information such as the vessel’s current position, course, speed, and estimated time of arrival at ports. The aggregation of AIS position reports provides a dynamic representation of a vessel’s movement over time, also described as the vessel’s trajectory.

The data provided by AIS, describing the movement of vessels through trajectories, serve as a foundation for a multitude of applications and research endeavors, playing a critical role in advancing maritime safety and operational efficiency. These

applications enhance maritime safety and efficiency through collision avoidance mechanisms [4], analysis of maritime traffic patterns [5], and optimization of routing for reduced travel time and improved Estimated Time of Arrival (ETA) [6].



(a) Trajectory of a cargo vessel traveling between Kiel and Brunsbüttel, missing multiple position reports. The arrow indicates the travel direction.



(b) The trajectory of the cargo vessel in Figure 1a) with realistic position reports after performing imputation.

Figure 1. Comparative trajectories of a cargo vessel traveling between Kiel and Brunsbüttel, illustrating the completeness of the imputation method.

Despite the broad utility of the AIS at enhancing maritime safety and efficiency, significant challenges impede its effectiveness [7]. In densely trafficked maritime zones, the volume of data transmitted between vessels and maritime authorities can exceed the inherent communication limits of the systems deployed, causing a loss of crucial position reports. Missing position reports can further be caused by a defective transceiver or the AIS being switched off manually. The challenge of missing position reports poses a risk to maritime safety by hindering effective collision avoidance

but also affects the accuracy of maritime traffic analysis and routing optimization.

Additionally, the integrity of AIS data is further compromised by instances of vessels transmitting under the same Maritime Mobile Service Identity (MMSI)—a unique identifier for each vessel—leading to potential identification errors, directly impacting one of AIS’s core functions of vessel identification and tracking. Manual entry errors, including outdated ETAs, incorrect destination information, and outdated navigational statuses, also undermine the integrity of AIS data. Such inaccuracies can significantly impair the effectiveness of AIS applications, from safety protocols to operational decision-making, thereby challenging the overall efficiency of maritime operations. Building on the understanding of the challenges encountered by the AIS, this paper focuses on the challenge of missing position reports.

An example of a real vessel trajectory with missing position reports is illustrated in Figure 1a. Initially, following the direction traveled by the vessel indicated by the black arrow, the trajectory features densely placed position reports. However, between the two highlighted position reports, a significant gap creates the appearance that the vessel is traversing land. Following the second highlighted position report, although the trajectory may not be missing position reports, incorporating additional ones enhances the realism of the trajectory. This improvement allows applications to utilize the trajectory more effectively, enabling, for example, more accurate calculations of the ETA.

To address the large gaps created by missing position reports, along with improving the visual path traveled by vessels, our research aims to develop a trajectory imputation framework that is applicable across a wider range of vessel types, thereby offering a comprehensive solution for the imputation of maritime trajectory data. We achieve this by constructing a regional grid in which positional reports from historical AIS data are assigned to each subregion. Subsequently, we employ a sea depth map to develop a graph for each subregion, ensuring connectivity between adjacent subregions within the grid.

To reduce the complexity of constructing the graph, we utilize a density sampling technique, such that the number of vertices in the graph is reduced, without reducing area coverage. By using a grid, we can focus on the specific subregions traversed by a vessel. Utilizing a sea depth map along with properties provided by the AIS, such as course over ground, vessel type, and draught, vertices are connected and traversed using the A* algorithm [8]. This allows for imputation to only be performed if a vessel can travel to a given vertex. To avoid unrealistic zigzag patterns in the trajectories, the quality of the trajectories is improved using a linear matrix equation, such that the trajectory is visually improved and looks realistic.

Our methodology is inspired by a method described by three papers [9, 10, 11] proposing the novel idea of constructing a graph from historical trajectory data, rather than relying on pre-established road networks. While the imputation methods

proposed by Isufaj et al. [9] and Elsharif et al. [10] demonstrate effectiveness in addressing vehicle trajectory data on land, their applicability to AIS data presents significant challenges, due to vessel movement being influenced by factors such as sea depth constraints. Furthermore, unlike land-based vehicles which are largely confined to road networks, maritime vessels navigate through a considerably more flexible spatial domain. Although vessels follow designated shipping lanes, the vastness of the maritime environment allows for a wider range of movement, significantly broadening the potential area of travel.

While the proposal by Magnussen et al. [11] does target vessel trajectories, it also presents its own set of limitations. Primarily focusing on tanker vessels, it overlooks the diversity of vessel types and their specific navigational requirements, such as sea depth. This not only narrows the applicability of their methods but also misses the nuances that other vessel types, such as passenger, fishing, and tender vessels introduce into the trajectory data. Additionally, the focus on trajectories within large open sea areas fails to account for the complexities encountered in coastal sea areas. In these areas, movement is more constrained, and navigational challenges are increased due to the proximity of land, influencing the patterns of vessel trajectories.

The following summarizes the contributions of this paper:

- Implement graphs using a regional grid and sea depth map.
- Design an imputation method that utilizes properties provided by the AIS, such as draught and course over ground, to address the issue of missing position reports.
- Supports trajectory imputation for multiple vessel types, such as fishing, cargo, and passenger vessels.
- Utilizes 397 days worth of AIS data (1,489,884,660 vessel positions) to implement, test, and validate our solution against alternatives.

The remainder of this paper is organized as follows: Section 2 discusses related work, Section 3 introduces a set of preliminaries that are used throughout the paper. While Section 4 presents the solution, Section 5 highlights the results, Section 6 concludes the paper.

2. RELATED WORK

In this section, we examine related approaches to trajectory imputation, as well as similar studies related to time series imputation.

A wide range of studies examine techniques for inserting points between two consecutive GPS points. This notion includes various names such as trajectory interpolation, -restoration, -recovery, -cleaning, and -imputation. Common for all of these approaches is to combat trajectories with large gaps, by inserting candidate GPS points in the trajectory. This task can generally be viewed from two different settings, based on whether underlying map information, such as a road network, is present or not. For example, several studies [12, 13, 14] rely on existing road networks to apply imputation methods for generating missing GPS points in trajectories

with gaps. However, for the maritime domain, such existing networks are not present, and cannot be utilized for vessel trajectory imputation. Furthermore, by using underlying networks the imputation becomes reliant and requires continuously up-to-date maps as networks may evolve.

To avoid relying on existing networks for trajectory imputation, previous studies [9, 10, 11, 15] have proposed leveraging graphs derived from GPS points of historic trajectories, in conjunction with employing a graph search algorithm to perform imputation for new trajectories.

One of these studies [10] proposes a novel method *TrImpute* that relies on 'crowd wisdom' to guide the imputation process of trajectories with gaps, meaning it prioritizes directions frequented by other trajectories during the imputation process. However, to construct its graphs, *TrImpute* is dependent on trajectories with no missing points, which makes it susceptible to a high error rate in regions, where data is sparse. It further uses an exhaustive search strategy to traverse its graph, leading to computational inefficiency during the imputation. A similar approach [9], *GTI*, mitigates the computational inefficiency of *TrImpute* by using Dijkstra's shortest path algorithm. However, both methods have only been tested on vehicle trajectories and do not account for different types of vehicles. This limitation makes them less suitable for vessel trajectories, where it is crucial to ensure that imputation does not occur in locations where vessels cannot normally travel, such as on land or in shallow waters where grounding might occur. The navigability of travel varies depending on the vessel type and its draught.

A study [11] that does consider vessel trajectories, is the novel method *DAISTIN* that has a similar approach to the aforementioned studies, *GTI* and *TrImpute*, but uses an A^* shortest path algorithm to traverse the graph. As this method only considers vessels of the type 'oil tanker', it has not been tested for other vessel types, and further, like the other related studies, does not consider locations where the vessel might ground, as this consideration is not necessary when imputing data for a single vessel type.

Unlike the aforementioned studies that are based on graphs, one study [15] proposes the novel method *KAMEL*, and is based on the Natural language processing (NLP) model *BERT*. *BERT* can perform linguistic tasks, including sentence completion, and is trained to predict missing words in a sentence, given the surrounding context of the left- and right word. *KAMEL* is equipped with a *BERT* model improved with spatial awareness, such that it can find missing GPS points in trajectories. *KAMEL* showcases a high accuracy for performing imputation on trajectories with few points, however, like the aforementioned studies, *KAMEL* also only considers vehicle trajectories, and does not extend to considering different types of vehicles.

Instead of leveraging graphs from historic trajectory data, or utilizing NLPs, another interesting approach to the imputation problem is proposed by other studies [12, 16, 17]. The propositions are inspired by the Generative Adversarial Network (GAN) methodology or framework proposed by one study [18]. The GAN framework consists of two main parts, a

generator network, and a discriminator network, that compete with each other to produce the best possible imputation positions. For instance, the study by Shi et al. [16] describes how a generator uses an encoder-decoder architecture to fill in missing trajectory data, while a discriminator evaluates the realism of the generated trajectories. This iterative process continues until the discriminator cannot differentiate between the original and the generated trajectories. Despite its ingenuity, this methodology proves unreliable for our purposes in vessel trajectory imputation. The technique relies on large amounts of trajectory data with no missing points as training material, which contrasts sharply with the sparse nature of AIS data, characterized by significant gaps between consecutive AIS points.

While *DAISTIN* like *DGIVT* considers vessel trajectories, none of the aforementioned studies [9, 10, 11, 15] consider the notion of inaccessible regions. For vessels, this encapsulates land and sea regions where grounding can occur. *DGIVT* uses a similar methodology to *TrImpute*, *GTI* and *DAISTIN*, but with the awareness of inaccessible regions by considering a vessel's draught, as well as the sea depth.

3. PRELIMINARIES

This section presents concepts used throughout this study. The first concept is a position report. A single position report contains 27 data fields [19], where not all fields are relevant to the algorithms we present in Section 4. We limit ourselves to 11 of the 27 data fields, namely longitude, latitude, timestamp, draught, course over ground, speed over ground, navigational status, vessel type, width, length, and MMSI, which is a unique identifier of a vessel. To make a clear distinction between position reports with 27 data fields, and a position report with 11 data fields, a vessel sample is introduced in Definition 1.

Definition 1 (Vessel Sample). *A vessel sample vs is an 11-tuple $(t, lng, lat, draught, cog, sog, nav_status, vessel_type, width, length, mmsi)$ where:*

- t is the timestamp, indicating when the position was transmitted.
- lng and lat represent the longitude and latitude of the vessel, respectively.
- $draught$ is the draught measurement of the vessel and indicates the depth of water required to float.
- cog and sog are abbreviations of the course over ground and speed over ground, where course over ground describes the heading of the vessel, and speed over ground describes the speed of the vessel, respectively.
- nav_status and $vessel_type$ represent the navigational status and the vessel type of the vessel, respectively.
- $width$ and $length$ are the width and length of the vessel.
- $mmsi$ is the MMSI, a unique identifier for the vessel.

With the introduction of vessel samples, we utilize the Haversine formula [20][p. 2] to calculate the great-circle distance between two vessel samples on a sphere, using the positions' longitudes and latitudes. This formula calculates the distance in meters between two vessel samples, assuming that

the radius r of the earth is 6,371,000 meters. The formula is shown in Definition 2.

Definition 2 (Haversine). Given two vessel samples vs_1 and vs_2 , the Haversine formula calculates the distance in meters between the vessel samples, given their longitudes and latitudes. It is defined as:

$$d(vs_1, vs_2) = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\phi_2 - \phi_1}{2} \right) + \cos(\phi_1) \cos(\phi_2) \sin^2 \left(\frac{\gamma_2 - \gamma_1}{2} \right)} \right),$$

where:

- $\sqrt{}$ is the square root.
- ϕ_1 and ϕ_2 are the latitudes of vs_1 and vs_2 , respectively, expressed in radians.
- γ_1 and γ_2 are the longitudes of vs_1 and vs_2 , respectively, also expressed in radians.
- r is Earth's radius, approximately, 6,371,000 meters.

We can extend the formula d to also work with other tuples containing longitude and latitude

With the definitions of a vessel sample and the Haversine formula, we define a vessel trajectory.

Definition 3 (Vessel Trajectory). Let VS be a set of vessel samples. A vessel trajectory vt is a sequence $vt = \langle vs_1, \dots, vs_n \rangle$, given the following conditions:

$$\begin{aligned} n &\geq 2 \\ vs_i &\in VS \wedge i \in \{1, \dots, n\} \\ vs_i.mmsi &= vs_{i+1}.mmsi \wedge i \in \{1, \dots, n-1\} \\ vs_1.t &< vs_2.t < \dots < vs_n.t \end{aligned}$$

The distance D traveled by a vessel trajectory is calculated using the Haversine formula d as follows:

$$D(vt) = \sum_{i=1}^{n-1} d(vs_i, vs_{i+1})$$

We proceed to define additional concepts used for vessel trajectory imputation. As described by Section 1, *DGIVT* uses a sea depth map to guide the imputation of vessel trajectories, such that vessels do not appear to either cross land or travel in locations, where they cannot travel due to their draught. To utilize the sea depth map, we first define a grid.

Definition 4 (Grid). Given a geographical region \mathcal{R} bounded by coordinates (lng_{\min}, lat_{\min}) and (lng_{\max}, lat_{\max}) , a grid G is a matrix of subregions $G_{i,j}$ with m rows and n columns, representing subdivisions into $m \times n$ subregions $G_{i,j}$, referred to as cells, with subareas:

$$\begin{aligned} G_{i,j} &= [lng_{\min} + (j-1) \cdot \Delta lng, \quad lng_{\min} + j \cdot \Delta lng) \\ &\quad \times [lat_{\min} + (i-1) \cdot \Delta lat, \quad lat_{\min} + i \cdot \Delta lat) \\ i &\in \{1, \dots, m\} \\ j &\in \{1, \dots, n\}, \end{aligned}$$

where Δlng and Δlat are the longitudinal and latitudinal dimensions of each cell, calculated as:

$$\begin{aligned} \Delta lng &= \frac{lng_{\max} - lng_{\min}}{n} \\ \Delta lat &= \frac{lat_{\max} - lat_{\min}}{m} \end{aligned}$$

The centroid of each cell $G_{i,j}$ is calculated as:

$$\text{centroid}(G_{i,j}) = (lng_{\min} + (j-0.5) \cdot \Delta lng, \quad lat_{\min} + (i-0.5) \cdot \Delta lat)$$

Next, each cell $G_{i,j}$ has a sea depth $\text{depth}_{i,j}$, which may be unknown. With the formula for finding the centroid of a cell, it can be determined given a vessel sample vs , which cell $G_{i,j}$ in grid G has the centroid closest to vs , using the method $\text{closest_cell}(vs, G)$ defined by next:

$$\text{closest_cell}(vs, G) = \left(\arg \min_{G_{i,j}} d(vs, \text{centroid}(G_{i,j})), \quad \min_{G_{i,j}} d(vs, \text{centroid}(G_{i,j})) \right)$$

where $\arg \min$ returns the cell $G_{i,j}$ with the centroid having the smallest distance to the vessel sample, and \min returns the distance to the cell. If two cells have the same distance, the first one found is returned.

Furthermore, given a vessel sample vs , the intersection method $\text{ints}(vs, G_{i,j})$ determines whether vs is within the boundaries of cell $G_{i,j}$.

$$\text{ints}(vs, G_{i,j}) = \begin{cases} 1 & ((lng_{\min} + (j-1) \cdot \Delta lng) \leq vs.lng < (lng_{\min} + j \cdot \Delta lng)) \wedge \\ & ((lat_{\min} + (i-1) \cdot \Delta lat) \leq vs.lat < (lat_{\min} + i \cdot \Delta lat)) \\ 0 & \text{otherwise} \end{cases}$$

A simplified grid is illustrated in Figure 2. It illustrates a grid of size m rows and n columns. Each cell is labeled with its column and row index, and each has a red dot, representing the cells' centroids. The blue dot illustrates a vessel sample vs that falls within the bounds of cell $G_{2,2}$.

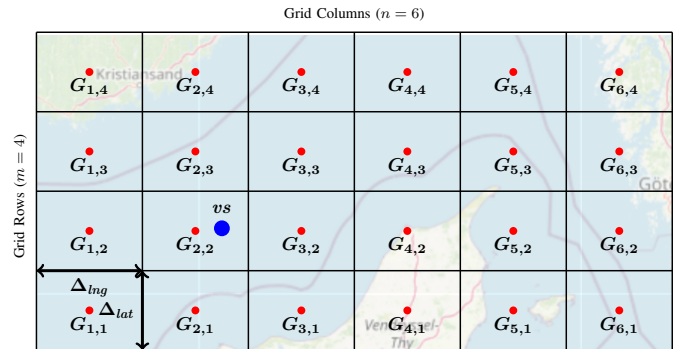


Figure 2. Illustration of the grid system as defined in Definition 4.

With the concept of grid introduced, we introduce the next concept, directed graph, which is essential for the imputation of vessel trajectories.

Definition 5 (Directed Graph). A directed graph DG is defined by as a 3-tuple $DG = (V, E, w)$, where

- V is a non-empty set of vertices
- $E \subseteq V \times V$ is a set of 2-tuples, called directed edges.

For each directed edge $(u, v) \in E$

- u is the source vertex
- v is the destination vertex
- $u \in V, v \in V, u \neq v$
- $w : E \mapsto \mathbb{R}$ is an associated weight that maps each edge (u, v) to a numerical value

In Section 4, we will showcase the *DGIVT* solution of graph creation and imputation, utilizing the definitions defined in this section.

4. DGIVT

In this section, we introduce the key components of Depth-Map enhanced Graph Imputation for Vessel Trajectories (DGIVT), a framework for performing imputation on vessel trajectories.

4.1 DGIVT overview

An overview of DGIVT is presented in Figure 3. The figure illustrates the three modules of the framework and the overall process of data flow and execution. The rectangular boxes represent input and output, whereas the hexagonal boxes denote data processing. The following sections in this chapter provide an in-depth look at these modules and their purposes.

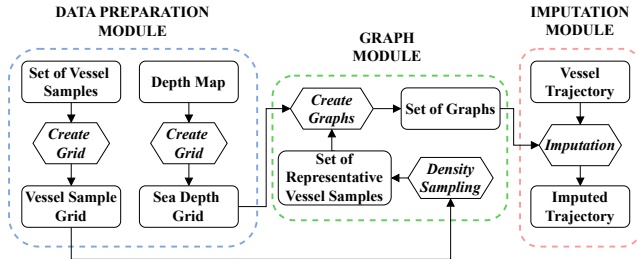


Figure 3. Overview of the DGIVT Framework.

Data Preparation Module [Section 4.2]: This module encapsulates two core functionalities. The first functionality processes as a set of vessel samples, associating each with a cell in a grid, referred to as vessel sample grid in Figure 3. This grid serves as a foundational data structure for subsequent graph construction algorithms. Concurrently, another grid is constructed using a depth map, associating each cell in this grid with a sea depth. We refer to this grid as the sea depth grid in Figure 3.

Graph Module [Section 4.3]: This module leverages both the vessel sample grid and sea depth grid generated by the Data Preparation Module. Its primary purpose is graph

construction, beginning with a density sampling technique [21] that reduces the number of vessel samples in cells with dense spatial representation. Following the density sampling technique, graphs are constructed with the remaining vessel samples as vertices for each cell in the vessel sample grid. Each vertex is associated with a depth, using the sea depth grid. Edges are furthermore established between graphs of connected cells.

Trajectory Imputation Module [Section 4.4]: This module leverages the graphs created in the Graph Module. It identifies cells in the vessel sample grid traversed by a given vessel trajectory. The imputation process involves integrating consecutive vessel samples into the graphs as vertices, specifically targeting the graphs corresponding to the cells intersected by these samples. To traverse the graph, we apply an A^* to guide the imputation.

4.2 Data Preparation Module

The data preparation module has two core functionalities. First, it processes the incoming set of vessel samples, VS , which are later used for graph construction. A naive approach is to use all the vessel samples to construct the graphs; however, the number of vessel samples may reach billions, making such processing computationally inefficient. We therefore propose an alternative approach that divides the overall spatial region into smaller subregions. Each subregion is represented as a cell, $VSG_{i,j}$, within a grid, VSG , and contains a subset of the vessel samples in VS . An illustration with real vessel samples is seen in Figure 4, highlighting the vessel samples associated with a respective cell in the vessel sample grid.

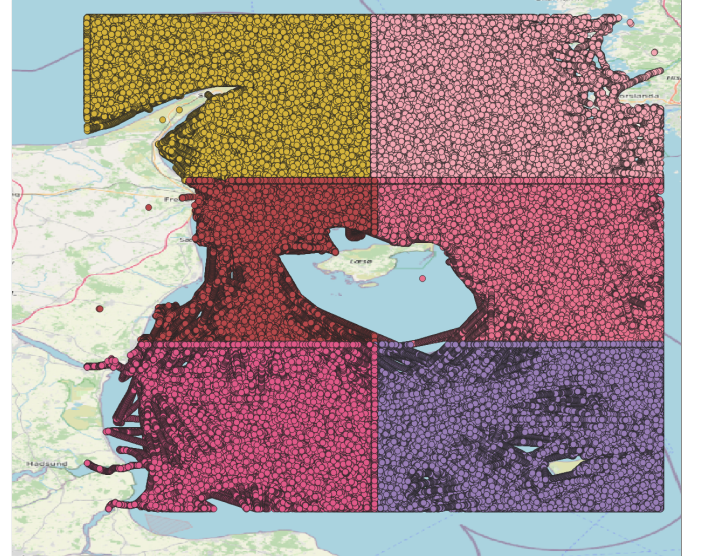


Figure 4. Illustration of the vessel sample grid VSG . Each color corresponds to a specific cell $VSG_{i,j} \in VSG$, and each cell contains a subset of vessel samples $vs \in VS$. The centrally located region, with no vessel samples, appears due to shallow depth, making it inaccessible to vessels. The presence of vessel samples on land likely occurs due to faulty AIS transmission, as vessel samples with a draught ≤ 0 have been discarded.

In addition to creating the vessel sample grid, the *DGIVT* framework utilizes a sea depth grid, *SDG*, where each cell, $SDG_{i,k}$, is associated with a depth value. The sea depth grid is used for associating each vessel sample with a depth. Thus, the module outputs two distinct grids: the vessel sample grid, *VSG*, which reduces computational complexity in graph construction and vessel trajectory imputation, and the sea depth grid, *SDG*, which enriches each vessel sample with sea depth information.

4.3 Graph Module

Given the vessel sample grid, *VSG*, and a sea depth grid, *SDG*, as arguments, the primary objective of the graph module is to create a directed graph, *DG*, for a set of vessel samples, *VS*, associated with a cell in the vessel sample grid, *VSG*. This section presents a data-driven approach for constructing such graphs. The first step in the module processes the incoming vessel samples, *VS*. If the cell contains a substantial amount of vessel samples, it can become a bottleneck when performing imputation on a vessel trajectory, due to the size of the graph created for the cell. Hence, we reduce the number of vessel samples in cells with dense spatial representation to improve computational efficiency while preserving the spatial representation of the data. In subregions with a high density of vessel samples, we minimize their number; in subregions with a low density of vessel samples, we maintain the existing number of samples. Algorithm 1, introduces a density sampling technique, implementing a vessel sample reduction function.

Algorithm 1 Density Sampling

Input: A set of vessel samples (*VS*), a vessel sample cell ($VSG_{i,j}$), and a neighbor distance threshold ($ndist_th$)

Output: Set of vessel samples *RVS*

```

1: function DENSITY_SAMPLING(VS,  $VSG_{i,j}$ ,  $ndist\_th$ )
2:   RVS  $\leftarrow \emptyset$ 
3:   EVS  $\leftarrow \emptyset$ 
4:   CVS  $\leftarrow \{vs \mid vs \in VS \wedge \text{ints}(vs, VSG_{i,j})\}$ 
5:   for all vs  $\in CVS$  do
6:     if EVS = CVS then
7:       Exit
8:     if vs  $\notin EVS$  then
9:       NB  $\leftarrow \text{range\_query}(vs, CVS, ndist\_th)$ 
10:      md  $\leftarrow \max_{nvs \in NB} nvs.draught$ 
11:      md  $\leftarrow \max(vs.draught, md)$ 
12:      Q  $\leftarrow \text{quadrants}(md, NB, EVS)$ 
13:      RVS  $\leftarrow RVS \cup Q$ 
14:      EVS  $\leftarrow EVS \cup NB \cup \{vs\}$ 
15:   return RVS

```

The algorithm takes three arguments: the set of vessel samples, *VS*, a cell in the vessel sample grid, $VSG_{i,j}$, and a neighbor distance threshold, $ndist_th$. In Lines 2 and 3, we initialize two empty sets: *RVS*, for storing spatially representative vessel samples, and *EVS*, for tracking excluded vessel samples. We then proceed in Line 4 to instantiate *CVS*

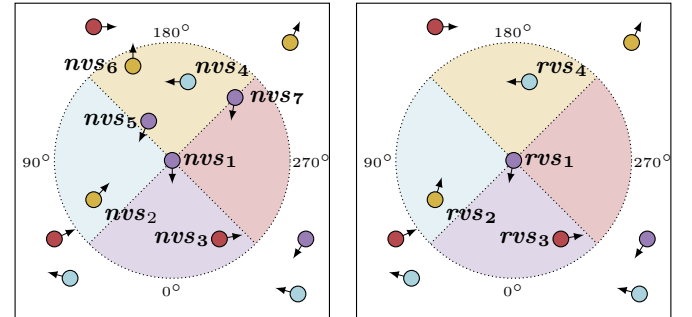
as a subset of vessel samples in *VS* that intersects with cell $VSG_{i,j}$ using the intersection function defined in Definition 4.

In Line 5, we iterate through all the vessel samples in *CVS*, ensuring in Lines 6 to 8 that neither the elements in *EVS* correspond to the elements in *CVS*, nor that we have processed a vessel sample already in *EVS*. If *EVS* = *CVS*, it means we have finished processing all the vessel samples, and we exit the *for* loop. Conversely, if we have only processed the current vessel sample, we continue to the next vessel sample.

In Line 9, we employ a range query function to locate all neighboring vessel samples to *vs* within the neighbor distance threshold, $ndist_th$. Although not illustrated in the algorithm, the set *NB* is organized as a kdtree data structure [22], which is a binary search tree specifically designed for efficiently searching multidimensional data, such as vessel samples.

In Lines 10 and 11, we define the variable *md*, representing the maximum draught among *vs* and its neighboring vessel samples, $nvs_i \in NB$. This variable is useful during graph creation when no associated depth in our sea depth grid is available for the vessel sample, allowing us to use *md* as an alternative measure.

Line 12 introduces the variable *Q*, instantiated through the *quadrants* function. This function takes three arguments: *md*, *NB*, and *EVS*. It is designed to select a maximum of four representative vessel samples, and we demonstrate the function in Figure 5, where Figure 5a shows the set of neighbors, $\{nvs_1, \dots, nvs_7\} \in NB$.



(a) Shows the set of neighbors $nvs_i \in NB$ within a given distance $ndist_th$. Each vessel sample is depicted in a different color to indicate its course above ground relative to the randomly selected center nvs_1 . (b) Results after performing the *quadrants* function. The representative vessel samples rvs_i contain the tuples elements of its associated nvs_i , but with updated draught and course over ground.

Figure 5. The *quadrants* function. It shows a circle divided into four colors, centered on nvs_1 . The arrows indicate the vessel samples' course over ground, and the colors of the vessel sample show which quadrant they are associated with in the circle, with respect to nvs_1 .

We use colors to illustrate the course over ground differences between neighboring vessel samples, categorized into four ranges: $[315^\circ, 45^\circ]$, $(45^\circ, 115^\circ]$, $(115^\circ, 225^\circ]$, and $(225^\circ, 275^\circ)$, relative to $nvs_1.cog$. For each quadrant, we select a neighbor vessel sample, calculate the average course over ground for all neighbor vessel samples within the quadrant, and generate a corresponding representative vessel sample, incorporating both the maximum draught variable, *md*, and an

average course over ground for the quadrant. This is illustrated in Figure 5b, where the arrows on the representative vessel samples, correspond to an average of the course over ground for the neighbor vessel samples associated with the quadrants. The argument *EVS* is used to ensure that a neighbor vessel sample, used as a representative vessel sample, has not been added to *EVS*.

An example of applying density sampling to real vessel samples, in a cell, is demonstrated in Figure 6. The figure shows a clear distinction between the number of vessel samples before and after the implementation of the density sampling algorithm, while keeping a spatial representation of the vessel samples.

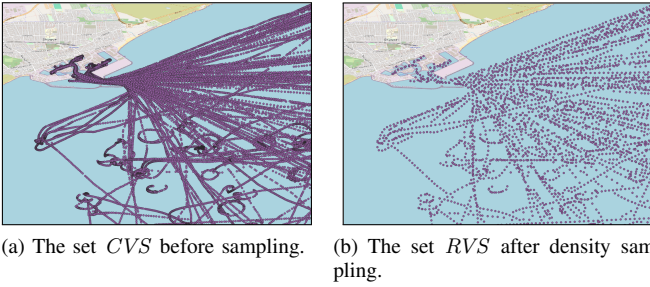


Figure 6. Illustration of Density Sampling.

The reduction of vessel samples significantly improves the computational efficiency during vessel trajectory imputation, as the numbers of vertices, V , and edges, E , to consider are significantly reduced. The next step of the module is to create a directed graph, DG , using the representative vessel samples RVS . In Algorithm 2, we present our graph creation approach.

The algorithm takes six arguments: the representative vessel samples, RVS , a sea depth grid, SDG , a cell in the vessel sample grid, $VSG_{i,j}$, two threshold values, $dist_th$ and $cdist_th$, and a distance penalty rate, $dist_pr$.

In Line 2, we define a set VVS , which we instantiate to all vessel samples vs in RVS that intersect with the cell $VSG_{i,j}$. To associate a vessel sample with a depth, we extend our definition of a directed graph from Definition 5 to include an additional tuple element, dm , such that a directed graph is now defined as a 4-tuple $DG = (V, E, w, dm)$, where dm is defined as a mapping function:

$$dm : VS \mapsto \mathbb{R},$$

We instantiate the tuple elements required for building the directed graph, DG , in Lines 3, 4, 5, and 6.

In Line 7, we proceed by iterating through all vessel samples, $vs \in VVS$. For each vs , we use the function *closest_cell* in Line 8, described in Definition 4, to retrieve the cell $SDG_{k,l} \in SDG$ with a centroid closest to vs , along with the distance to the centroid of $SDG_{k,l}$, $cdist$. Retrieving the distance to the centroid of the $SDG_{k,l}$ is necessary, as we do not necessarily have a sea depth grid that covers the entire domain. We check in Line 9, whether $cdist$ exceeds the cell distance threshold, $cdist_th$. If this is the case, we map the

Algorithm 2 Graph Creation

Input: A set of representative vessel samples (RVS), a sea depth grid (SDG), a cell in vessel sample grid ($VSG_{i,j}$), a distance threshold ($dist_th$), a cell distance threshold ($cdist_th$), a distance penalty rate ($dist_pr$)

Output: Directed graph DG

```

1: function CREATE_GRAPH( $RVS, SDG, VSG_{i,j}, dist\_th,$ 
                         $cdist\_th, dist\_pr$ )
2:    $VVS \leftarrow \{vs \mid vs \in RVS \wedge ints(vs, VSG_{i,j})\}$ 
3:    $V \leftarrow \emptyset$ 
4:    $E \leftarrow \emptyset$ 
5:    $w \leftarrow$  new mapping function for edges
6:    $dm \leftarrow$  new mapping function for vertices to depth
7:   for  $vs \in VVS$  do
8:      $(SDG_{k,l}, cdist) \leftarrow closest\_cell(vs, SDG)$ 
9:     if  $cdist > cdist\_th$  then
10:       $dm(vs) \leftarrow \neg vs.draught$ 
11:     else
12:       $dm(vs) \leftarrow depth_{k,l}$ 
13:      $V \leftarrow V \cup \{vs\}$ 
14:   for  $vs \in V$  do
15:      $NB \leftarrow range\_query(vs, V, dist\_th)$ 
16:      $tdist \leftarrow dist\_th$ 
17:     while  $NB \setminus \{vs\} = \emptyset$  do
18:        $tdist \leftarrow tdist \cdot dist\_pr$ 
19:        $NB \leftarrow range\_query(vs, tdist)$ 
20:     for  $nvs \in NB \setminus \{vs\}$  do
21:        $w(vs, nvs) \leftarrow weight(vs, nvs)$ 
22:        $E \leftarrow E \cup \{(vs, nvs)\}$ 
23:    $DG \leftarrow (V, E, w, dm)$ 
24:   return  $DG$ 

```

negated draught value of the vessel sample in Line 10, and otherwise, map the depth, $depth_{k,l}$, of the cell to the vessel sample in Line 12. We then proceed to add vs to the set of vertices, V , in Line 13.

In Line 14, we enter a second *for* loop, iterating through each $vs \in V$. We begin by instantiating the set of neighbors NB for $vs \in V$. Next, we introduce a temporary distance variable, $tdist$. We use this variable, given $NB = \emptyset$ evaluates to *true* in the *while* loop in Line 17. We multiply $tdist$ with the distance penalty rate, $dist_pr$, to keep increasing the search space for neighbors until the predicate of the *while* loop evaluates to *false*.

Once the *while* loop evaluates to *false*, we iterate through all the neighboring vessel samples, $nvs \in NB$ in Line 20. Within the *for* loop, we map the weight from vs to nvs to w using the *weight* function. The weight function takes two arguments: the current vessel sample, vs , and the current neighbor vessel sample, nvs .

The function for *weight* is defined as follows:

$$weight(vs_1, vs_2) = d(vs_1, vs_2) + cog_penalty(vs_1, vs_2),$$

where $cog_penalty$ influences the weight of the edge, if they are not traveling in a similar direction. We formally define the $cog_penalty$ as:

$$cog_penalty(vs_1, vs_2) = \left(\frac{\min(|cd|, 360 - |cd|)}{180} \right) \cdot \Theta,$$

where:

- $cd = vs_2.cog - vs_1.cog$.
- Θ is an angle threshold applied, when the course over ground $vs_2.cog \neq vs_1.cog$.

We demonstrate the $weight$ function in Figure 7.

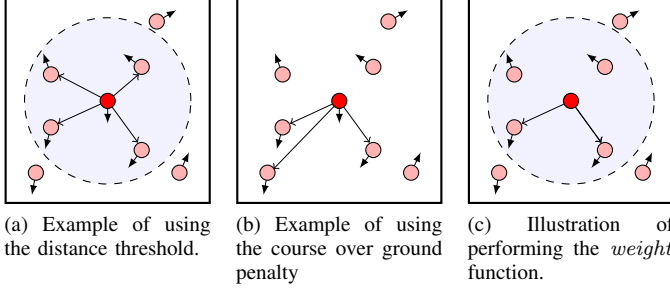


Figure 7. Illustrates the function $weight$.

Figure 7a illustrates how the edges are created, if we only consider the distance to the neighboring vessel samples. Figure 7b illustrates how the edges are created, if we only consider the course over ground for the neighboring vessel samples. Here we see, that unlike in Figure 7b, we only create edges when the course over ground does not differ greatly. Figure 7c illustrates how the edges are created when we combine the two functions. We see that we only include the neighboring vessel samples if they are both within the distance threshold and have a course over ground that does not differ greatly from vs .

After creating the mapping to w , we add the edge (vs, nvs) to the set of edges, E , in Line 22, and continue iterating. Once done, we instantiate and return the directed graph, DG , in Lines 23 and 24.

To ensure connectivity across different subregions in VSG , it is essential to establish edges between the individual graphs associated with each subregion. For instance, consider a scenario where we need to perform imputation on a vessel trajectory starting in subregion $VSG_{1,1}$, and ends in subregion $VSG_{1,2}$. Without interconnected graphs, it would be impossible to traverse between subregions. To address this, we link the graphs within VSG using the method outlined in Algorithm 2 (see Line 14). This process specifically employs an adjusted version of the $range_query$ to identify vessel samples, $vs \in VS$, for each subregion, $VSG_{1,1}$ and $VSG_{1,2}$, that are within a distance threshold of each other, ensuring spatial navigation across subregions.

To summarize, this module utilizes a density sampling technique to reduce the set of vessel samples, VS , without compromising the spatial representation, and return a representative set of vessel samples, RVS . Using the set of

representative vessel samples, the module continues creating vertices, V , edges, E , edge weights, w , and depths, dm , to create a directed graph, DG , for each grid cell $VSG_{i,j}$. Once a directed graph has been created for all cells in VSG , the vessel samples in neighboring cells, within a distance threshold of each other, are connected by edges in their respective directed graphs. The next module Section 4.4 will now leverage the graphs to perform vessel trajectory imputation.

4.4 Vessel Trajectory Imputation

The main objective of this module is to insert realistic vessel samples in a vessel trajectory. Our imputation approach can be viewed in Algorithm 3. It takes four arguments: A vessel sample grid, VSG , a vessel trajectory, vt , a distance threshold, $dist_th$, and a distance penalty rate, $dist_pr$.

Algorithm 3 Imputation

Input: A vessel sample grid (VSG), a vessel trajectory (vt), a distance threshold ($dist_th$), and a distance penalty rate ($dist_pr$)

Output: A vessel trajectory(ivt)

```

1: function IMPUTATION( $VSG, vt, dist\_th, dist\_pr$ )
2:    $ivt \leftarrow$  empty sequence of vessel samples
3:    $DG \leftarrow find\_intersecting\_graphs(VSG, vt)$ 
4:   for  $i \leftarrow 1$  to  $len(vt) - 1$  do
5:      $svs \leftarrow vs_i \in vt$ 
6:      $dvs \leftarrow vs_{i+1} \in vt$ 
7:      $DG.V \leftarrow DG.V \cup \{svs, dvs\}$ 
8:      $dist \leftarrow dist\_th$ 
9:      $SNB \leftarrow range\_query(svs, DG.V, dist\_th)$ 
10:    while  $SNB \setminus \{svs\} = \emptyset$  do
11:       $dist \leftarrow dist \cdot dist\_pr$ 
12:       $SNB \leftarrow range\_query(svs, DG.V, dist)$ 
13:    for  $nvs \in SNB \setminus \{svs\}$  do
14:      if  $reachable(svs, nvs)$  then
15:         $DG.E \leftarrow DG.E \cup \{(svs, nvs)\}$ 
16:         $DG.w(svs, nvs) \leftarrow weight(svs, nvs)$ 
17:     $DNB \leftarrow range\_query(dvs, DG.V, dist\_th)$ 
18:    Perform Lines 10 and 16 for DNB
19:     $dist \leftarrow d(svs, dvs)$ 
20:     $NB \leftarrow range\_query(svs, DG.V, dist)$ 
21:     $NB \leftarrow \{svs, dvs\} \cap (NB \cup range\_query(dvs, DG.V, dist))$ 
22:    Perform Lines 10 and 12 for NB
23:    for  $nvs \in NB \setminus \{svs, dvs\}$  do
24:      if  $\neg reachable(svs, nvs) \vee \neg reachable(nvs, dvs)$  then
25:        Remove all edges and vertices containing nvs
26:     $path \leftarrow A^*(DG, svs, dvs)$ 
27:    if  $path = \emptyset$  then
28:       $ivt \leftarrow ivt \circ \langle svs, dvs \rangle$ 
29:    else
30:       $ivt \leftarrow ivt \circ path$ 
31:  return  $ivt$ 

```

The algorithm begins by defining DG in Line 3, using the function *find_intersecting_graphs*. The function selects the graphs that the vessel trajectory, vt , travels through, and merges them. We then proceed traversing through every vessel sample $vs \in vt$, using a *len* function that returns the total number of vessel samples in the vessel trajectory. We extract one from the length of the vessel trajectory to avoid out-of-bounds issues.

We then proceed to instantiate a source vessel sample, svs , and a destination vessel sample, dvs , which are two consecutive vessel samples in vt . We add both svs and dvs to the vertices $DG.V$ in Line 7. In Lines 8 to 12, we follow a similar approach in Lines 15 to 19 in Algorithm 2, ensuring we find the neighbors SNB to svs . If no neighbors are found, we increase the search space, using a distance threshold, $dist_th$. Once the *while* loop evaluates to *true*, we begin traversing the found neighbors $nvs \in SNB$ in Line 13. For each nvs , we check whether svs is able to reach nvs , using the function *reachable* in Line 14. This function compares the draught of the source vessel sample with the depth value on the neighboring vessel sample. If it determines the depth of the neighbor vessel sample is appropriate given the draught of the source vessel sample, we add edge (svs, nvs) to the set of edges $DG.E$, as well as update the weight in mapping function in Lines 15 and 16.

This is illustrated in Figure 8, where the darker area symbolizes a depth, where the vessel can travel. Figure 8a

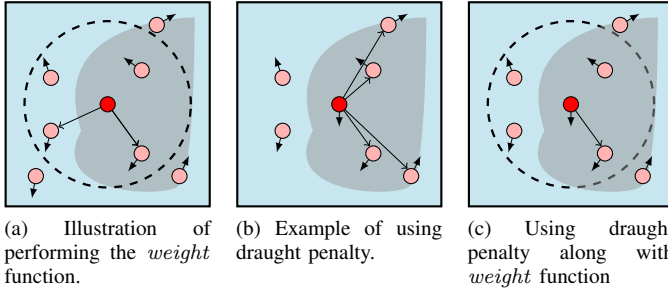


Figure 8. Illustrates the difference in edge creation when taking draught into account. The darker spots symbolize a depth, where the vessel can travel.

demonstrates the edges created for a source vessel sample, where we only apply the *weight* function. It illustrates that if we only use the *weight*, we end up creating edges to neighboring vessel samples, where the vessel cannot travel. Figure 8b demonstrates potential edges, if we only consider the draught of the vessel, and do not include the *weight* function. Here there vessel is able to travel to all the edges, but there is no filtering applied. Figure 8c demonstrates the edges created, when both using both functions, and we see that the combination of the functions returns only a single edge.

While not demonstrated in the algorithm, we follow a similar approach, as described above, to add edges between the neighboring vessels samples DNB to dvs in Lines 17 and 18

In Line 19, we initialize the variable $dist$ as the distance between svs and dvs , and in Lines 20 and 21, we define NB

as the set of common neighbors between svs and dvs . If the set of neighbors is empty, similar to earlier, we increase the search space.

Once we have a valid set of neighbors, NB , we utilize the *reachable* function again in Line 24 to remove neighboring vessel samples with an inappropriate depth relative to the draught values of svs and dvs . Consequently, neighboring vessel samples are removed in Line 25 if the *if* condition evaluates to *true*. After removing invalid neighboring vessel samples, we employ an A^* algorithm in Line 26.

Depending on whether the algorithm finds a path, we either add the found path or $\langle svs, dvs \rangle$ to the imputed vessel trajectory ivt in Lines 28 and 30. We then continue to the next two consecutive vessel samples in the *for* loop in Line 4, until the whole vessel trajectory has been processed. We return the imputed vessel trajectory in Line 31.

While not demonstrated in the algorithm, we update the timestamp, course over ground, and speed over ground of the vessel samples in the paths found by the A^* algorithm. We acquire this by performing linear interpolation [23], and we illustrate this calculation for the timestamp of the vessel sample in Equation 1. The calculation can be done similarly for speed over ground and course over ground.

$$vs_i.t = vs_1.t + \frac{vs_n.t - vs_1.t}{D(path)} \cdot \sum_{j=1}^{i-1} d(vs_j, vs_{j+1}) \quad (1)$$

where

- $1 < i < n$
- $\langle vs_1, \dots, vs_n \rangle \in path$
- $vs_1 = svs$
- $vs_n = dvs$

4.4.1 Vessel Trajectory Refinement: The arrangement of vertices in the directed graph produced in Algorithm 2, may result in Algorithm 3 producing vessel trajectories with a zigzag pattern. By solving a linear matrix equation and computing the least squares solution [24], we can determine whether a subset of the vessel positions in a vessel trajectory is considered straight. As long as the residuals obtained from solving the linear matrix equation are below or equal to the threshold ϵ , we say that the vessel travels in a straight line. When the residual exceeds ϵ , we say that the vessel is turning and look at the next subset of vessel positions. Algorithm 4 and Figure 9 provide a thorough description of this procedure.

Algorithm 4 takes two arguments: a vessel trajectory, vt , and a threshold value, ϵ , used to say when a subset of the trajectory is no longer considered to travel straight. An illustration of such a vessel trajectory can be seen in Figure 9a, where $vt = \langle vs_1, \dots, vs_7 \rangle$.

In Lines 2 and 3, we instantiate the variables *anchor* and *w*. The variable *anchor* determines the beginning of the sequence of vessel positions that we are currently looking at. As we are beginning at the start of the vessel trajectory, *anchor* is instantiated to one. The variable *w* represents the total number of vessel samples, we are considering at once. We instantiate

Algorithm 4 Vessel Trajectory Refinement**Input:** A vessel trajectory(vt) and a residual threshold ϵ **Output:** A vessel trajectory rvt

```

1: function TRAJECTORY_REFINEMENT( $vt, \epsilon$ )
2:    $anchor \leftarrow 1$ 
3:    $w \leftarrow 3$ 
4:    $turn\_detected \leftarrow false$ 
5:    $prev\_fit \leftarrow \langle vs_1, vs_2 \rangle \in vt$ 
6:    $rvt \leftarrow \langle \rangle$ 
7:   while ( $anchor + w \leq len(vt)$ ) do
8:      $sec \leftarrow \langle vs_{anchor}, \dots, vs_{anchor+w-1} \rangle \in vt$ 
9:      $(fit\_sec, res) \leftarrow best\_fit(sec)$ 
10:    if ( $res > \epsilon$ )  $\wedge$  ( $(anchor + w) < len(vt)$ ) then
11:       $seq\_ex \leftarrow \langle vs_{anchor}, \dots, vs_{anchor+w-2},$ 
12:                     $vs_{anchor+w} \rangle, vs \in vt$ 
13:       $(\_, res) \leftarrow el2(best\_fit(seq\_ex))$ 
14:      if  $res > \epsilon$  then
15:         $turn\_detected \leftarrow true$ 
16:      else
17:         $vs\_tp \leftarrow find\_center\_pos(\langle vs_{anchor+w-2},$ 
18:                                      $vs_{anchor+w-1},$ 
19:                                      $vs_{anchor+w} \rangle)$ 
20:         $sec \leftarrow \langle vs_{anchor}, \dots, vs_{anchor+w-2}, vs\_tp \rangle,$ 
21:                     $\langle vs_{anchor}, \dots, vs_{anchor+w-2} \rangle \in vt$ 
22:         $(fit\_sec, res) \leftarrow best\_fit(sec)$ 
23:      if  $turn\_detected$  then
24:         $rvt \leftarrow rvt \circ prev\_fit$ 
25:         $anchor \leftarrow anchor + w - 1$ 
26:         $w \leftarrow 3$ 
27:         $prev\_fit \leftarrow \langle vs_{anchor}, vs_{anchor+1} \rangle \in vt$ 
28:      else
29:         $prev\_fit \leftarrow fit\_sec$ 
30:         $w \leftarrow w + 1$ 
31:       $rvt \leftarrow rvt \circ prev\_fit$ 
32:    return  $rvt$ 

```

w to three, as a sequence of two vessel samples, $\langle vs_1, vs_2 \rangle$ always shapes a straight line, and, furthermore, is the minimum required number of vessel samples to define a vessel trajectory. This is also why the variable $prev_fit$ in Line 5 is instantiated as the two first vessel samples in the vessel trajectory.

In Line 6, we instantiate rvt , representing the refined version of vt , and in Line 7, we use a function len to check whether we are exceeding the length of vt . If this is not the case, we enter the *while* loop.

In Line 8, we instantiate the variable sec to a sequence of vessel samples, beginning from vs_{anchor} to $vs_{anchor+w-1}$. In Figure 9a, this is equivalent to the first three vessel samples in the vessel trajectory.

In Line 9, we use the $best_fit$ function to obtain a best-fit line [24] fit_sec along with a residual res used to say whether vt is traveling straight for the vessel samples in sec . An example

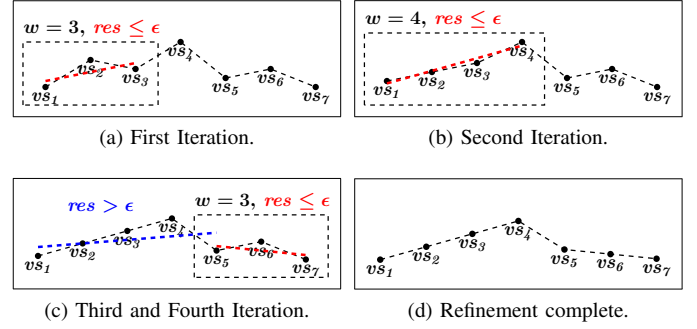


Figure 9. Illustrates Algorithm 4 on a vessel trajectory. Figure 9a, Figure 9b, Figure 9c, and Figure 9d illustrates how the vessel trajectory is refined for each iteration.

of a best-fit line is shown in Figure 9a as the red dashed line. We then check in Line 10, whether res exceeds the threshold ϵ and that $vt_{anchor+w}$ is not the last vessel sample in vt . If it evaluates to *true*, we enter the *if* case, and check if exchanging the vessel sample vs_{w-1} with the following vessel sample vs_w also result in a residual $res > \epsilon$. Here, we use a function $el2$ to retrieve the second tuple element returned by the function $best_fit$ in Line 12, as we only need the residual value. If the residual $res > \epsilon$, we enter the *if* case in Line 13 and instantiate the variable $turn_detected$ to *true* in Line 14. Otherwise, we enter the *else* in Line 15, and in Lines 16 and 18, use the function $find_center_pos$ to center $vs_{anchor+w-1}$ between the vessel samples $vs_{anchor+w-2}$ and $vs_{anchor+w}$ and again calculate the best-fit line.

In Line 19, given we determined the vessel to be turning, we enter the *if* and in Lines 20 and 23 we add $prev_fit$ to rvt , as it can be refined no further. We update the variables $anchor$ and w such that the variable $prev_fit$ looks at the next three vessel samples in the next iteration. If it is not determined that the vessel is turning in Line 19, we instead update $prev_fit$ to the variable fit_sec in Line 24, and expand w to look at the next vessel sample in vt .

Once the *while* loop evaluates to *false*, we expand the refined vessel trajectory rvt with the remaining vessel samples in $prev_fit$ in Line 27, and return the refined vessel trajectory rvt in line Line 28.

In Figure 9a, the residual value is not greater than or equal to ϵ , and we, therefore, update the vessel samples to the best-fit line and expand w . The same happens in the second iteration in Figure 9b, where the residual value is also not greater than or equal to ϵ . However, in the third iteration, the residual value is greater than ϵ , and we determine that the vessel is turning, resulting in a best-fit line with a residual less than ϵ . The completed trajectory is shown in Figure 9d.

5. EXPERIMENTS & EVALUATION

This section illustrates the effectiveness of *DGIVT* in imputing vessel trajectories. We introduce the data utilized to demonstrate *DGIVT* in Section 5.1. Section 5.3 highlights the evaluation metrics we use for comparing the original vessel

trajectories with the imputed trajectories. Section 5.4 presents the final results of *DGIVT*.

5.1 Data

We utilize real AIS data obtained from the Danish Maritime Authority to extract vessel trajectories. This involves both cleansing and filtration of the data, described in further detail in Appendix A. Additionally, we impose the constraint that all vessel samples must fall within the longitude and latitude of the Danish maritime domain. The vessel trajectories obtained from the cleansing and filtration process, contain vessel samples spanning the entire day, covering both moving and stationary periods. To determine whether a vessel trajectory should be divided, we consider factors such as speed over ground, navigational status of the vessel samples, and harbor data. This is also described in more detail in Appendix A. In total, we have 815,806 vessel trajectories, which in total include 1,489,884,660 vessel samples. The total distance covered by these trajectories is approximately 80,985,367.19 kilometers. The vessel sample grid, *VSG*, is restricted to the area of the Danish maritime domain, and each grid cell, $VSG_{i,j}$, covers an area of approximately $50km \times 50km$, resulting in 169 cells. The sea depth grid, *SDG*, is obtained from a .tiff file containing approximately 38,000,000 cells of size $50m \times 50m$, ranging with depth values from approximately $-480m$ to $0m$.

5.2 Experiments Setup

To perform comparable results with existing solutions, we limit our subregion coverage to nine cells, covering an area of $22.5000km$, as seen in Figure 10.

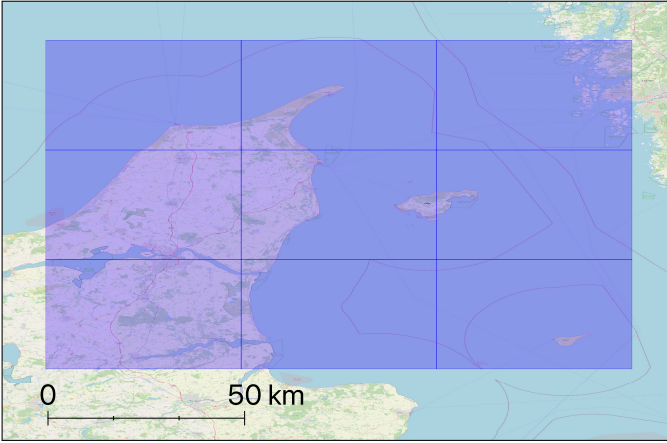


Figure 10. Nine cells in the vessel sample grid, covering an area of $22.5000km$.

These cells are associated with a total of 182,466,434 vessel samples, approximately 12 percent of our total vessel samples.

Both the implementation of *DGIVT* and our data is deployed on a server, equipped with the following specifications:

- AMD EPYC Processor with 16 vCPUs
- Operating System: Ubuntu 22.04 LTS (Jammy Jellyfish)

- 64 Gigabyte (GB) of RAM
- 1000 GB of SSD storage space for the Operating System, *DGIVT* components, and data

Imputation tests are conducted using three types of trajectories to assess performance under different conditions:

- Trajectories with **multiple gaps** of 8000, 4000, 2000, or 1000 meters, with a minimum distance of the respective intervals between consecutive vessel samples.
- Trajectories with a **single gap** of 8000, 4000, 2000, or 1000, respectively.
- Trajectories reflecting a **realistic** frequency of missing vessel samples, such that each expected vessel sample, based on the transmission rates in Table 1, is consistently absent.

Moving Status	Transmission Rate Frequency
Anchored or Moored	3 minutes
Speed between 0 and 14 knots	10 seconds
Speed between 0 and 14 knots and changing course	3.33 seconds
Speed between 14 and 23 knots	6 seconds
Speed between 14 and 23 knots and changing course	2 seconds
Speed above 23 knots	2 seconds
Speed above 23 knots and changing course	2 seconds

Table 1
Transmission rate of Class A vessels [25].

Additionally, we evaluate the effectiveness of *DGIVT* for vessels traversing all nine cells, as well as for those navigating near the coast. For vessel trajectories traversing the nine cells, we have selected 50 random trajectories across all vessel types and reduced the number of vessel samples for each trajectory type, as described previously. This results in an unequal amount of trajectories per run test, as not all the trajectories necessarily have traveled, for example, 8000 meters. To test the effectiveness of vessel trajectories near the coast, we select a 1000 random vessel trajectory in the area from Aalborg harbor to Kattegat. We select 1000 trajectories to make more meaningful comparisons with *GTI*, as it requires the same data for both graph creation and imputation.

5.3 Evaluation Metrics

For evaluating the correctness of our imputation, we use Dynamic Time Warping (DTW) [26] and Fréchet Distance [27] as metrics for comparing how close the imputed vessel trajectories are to the original vessel trajectories.

Given two vessel trajectories $vt_1 = \langle vs_1, \dots, vs_n \rangle$ and $vt_2 = \langle vs_1, \dots, vs_m \rangle$, the DTW algorithm finds the cumulative cost of the optimal alignment path between vt_1 and vt_2 . This is achieved using dynamic programming, starting with the computation of the haversine distance matrix, *HD*:

$$HD_{i,j} = d(vs_i, vs_j)$$

where:

- $vs_i \in vt_1 \wedge i \in \{1, \dots, n\}$.
- $vs_j \in vt_2 \wedge j \in \{1, \dots, m\}$.

The cumulative cost matrix, C , is then computed using the recursive relation:

$$C_{i,j} = HD_{i,j} + \min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1})$$

given the conditions:

- $C_{0,0} = HD_{0,0}$,
- $C_{i,0} = HD_{i,0} + C_{i-1,0} \wedge i \in \{1, \dots, n\}$,
- $C_{0,j} = HD_{0,j} + C_{0,j-1} \wedge j \in \{1, \dots, m\}$.

Finally, the $C_{n,m}$ represents the cumulative alignment cost, quantifying the similarity or dissimilarity between the two vessel trajectories.

The Fréchet Distance between vt_1 and vt_2 is the minimum, possible maximum distance between any two corresponding vessel samples in the two trajectories, as we traverse through them:

$$F(vt_1, vt_2) = \inf_{\alpha, \beta} \max_{t \in [0,1]} d(vt_1(\alpha(t)), vt_2(\beta(t))),$$

where α and β are continuous non-decreasing functions mapping the interval $[0, 1]$ to the parameterized indices of the two trajectories.

5.4 Results

The algorithms in Section 4 use three different threshold values: a distance threshold in density sampling, a distance threshold in graph creation, and a distance penalty rate. We aim to find the optimal threshold values for the graphs in the nine cells, through non-exhaustive experimentation. Table 2 shows the result of the optimal threshold values tested throughout the experimentation.

Table 2

List of threshold experimentation values, with the optimal column being the values resulting in the best imputation results.

Threshold values	Optimal	Start	End	Step
Distance Threshold in Density Sampling	80m	50m	150m	10m
Distance Threshold in Graph Creation	160m	100m	300m	10m
Distance Penalty in Imputation	50m	20m	100m	10m
Number of Vertices	4.17M	5.35M	1.11M	
Number of Edges	43.3M	58.8M	2.30M	

Table 2 reveals that the optimal value for the distance threshold in density sampling, yielding the most favorable imputation results, is approximately 80 meters, resulting in a total of 4,170,938 vertices.

In graph creation, we find that we get the best imputation results when the distance threshold is approximately twice the density sampling distance threshold, and therefore select a distance threshold of 160 meters, resulting in the creation of 43,349,897 edges. Choosing a lower threshold value,

combined with the course over ground penalty, complicates the creation of edges between vertices, due to how the density sampling distributes vertices across the graph to ensure even coverage. Increasing the distance threshold above 160 meters, resulted in fewer vessel samples in the imputation output, and yielded no substantial benefits.

For the distance penalty threshold in the imputation algorithm, we find that the optimal value is 50 meters. Increasing this threshold leads to a reduction in vessel samples, as it removes samples close to the distance threshold, despite their courses over ground being close. Conversely, a smaller value resulted in more vessel samples but included many with near opposite courses over ground.

Moving forward, the remaining part of this section aims to test and evaluate *DGIVT* using the optimal threshold values and compare with *Linear Interpolation* and *GTI*. We use the two evaluation metrics described in Section 5.3 to compare the imputed vessel trajectories with their original vessel trajectories. We test two subregions with both *DGIVT*, *Linear Interpolation*, and *GTI*. Each approach uses the three types of trajectories described in Section 5.2 with different gaps of sizes: 8000, 4000, 2000, and 1000 meters. The first subregion consists of the earlier mentioned nine cells, with ≈ 450 vessel trajectories for each gap size and type. The second subregion is from Aalborg harbor to Kattegat with ≈ 1000 vessel trajectories for each gap size and type and includes four subregions of the 9 cells. Our goal is to evaluate how effectively the three approaches, *DGIVT*, *GTI*, and *Linear Interpolation* can reconstruct the three types of vessel trajectories and compare to the original vessel trajectory’s shape and number of vessel samples.

The imputation results of *DGIVT* and *GTI* for the nine cells are detailed in Tables 3 and 4.

Table 3

DGIVT imputation results for the nine cells.

Vessel Trajectory gap	No. of Original vs avg.	No. of Reduced vs avg.	No. of Imputed vs avg.	Execution Time (s) avg.
Mult-8000	1723	11	734	29.50
Mult-4000	1667	19	730	31.35
Mult-2000	1618	35	721	41.45
Mult-1000	1604	66	727	55.87
Single-8000	1723	1444	1532	13.65
Single-4000	1667	1499	1546	13.50
Single-2000	1618	1507	1529	14.85
Single-1000	1604	1533	1535	15.21
Realistic	1556	1234	1289	19.68

The first column in both tables describes the three vessel trajectory types: multiple, single, and realistic, while the second column presents the average number of vessel samples found in the original trajectories. The third column presents the number of vessel samples remaining in the trajectory after reducing the vessel samples in each vessel trajectory, using the methods described in Section 5.2. The fourth column presents

Table 4
GTI imputation results for the nine cells.

Vessel Trajectory gap	No. of Original <i>vs</i> avg.	No. of Reduced <i>vs</i> avg.	No. of Imputed <i>vs</i> avg.	Execution Time (s) avg.
Mult-8000	1723	11	11	0.07
Mult-4000	1667	19	20	0.11
Mult-2000	1618	35	36	0.23
Mult-1000	1604	66	69	0.46
Single-8000	1723	1444	2186	13.19
Single-4000	1667	1499	2343	13.10
Single-2000	1618	1507	2407	16.78
Single-1000	1604	1533	2431	26.38
Realistic	1556	1234	1220	60.83

the average number of vessel samples after imputation.

Our findings highlight that for trajectories containing multiple gaps of 8000, 4000, 2000, and 1000 meters, *DGIVT* notably enriches the reduced vessel trajectories, reclaiming up to 45% of the original number of vessel samples. In contrast, *GTI* struggles to enrich the vessel trajectories, likely because *GTI* requires the vessel trajectories to travel closely to perform meaningful imputations. This is also reflected in Figures 12a and 12d, where both the DTW and Fréchet scores for the trajectories with multiple gaps are significantly lower for *GTI* and *Linear Interpolation* compared to *DGIVT*.

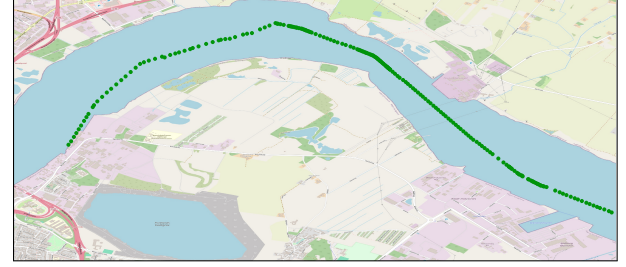
For trajectories featuring a single gap of 8000, 4000, 2000, and 1000 meters, respectively, *DGIVT* exhibits less substantial performance, contributing on average with 88 vessel samples to the reduced vessel trajectory. Meanwhile, *GTI* achieves a high number of vessel samples, however, upon closer examination of the imputed trajectories produced for both *DGIVT* and *GTI* in Figure 11, a nuanced difference emerges. While *GTI* does add a substantial amount of vessel samples, it does not necessarily contribute to closing the gaps as depicted in 11c, whereas *DGIVT* introduces significantly fewer samples but adeptly fills the gap in the vessel trajectory in 11b. This is likewise also depicted in Figures 12b and 12e where the two metric scores are lower in comparison to *GTI*, despite their overall amount of imputed vessel samples is significantly higher than *DGIVT*.

For the subregion from Aalborg harbor to Kattegat, the tests are conducted with a sample of 1000 random vessel trajectories, each reduced with the aforementioned methods. Results of the tests are seen in Tables 5 and 6 and figs. 13a to 13f.

Similar to the earlier results of the nine cells, the vessel trajectories with multiple gaps, *DGIVT* can reclaim approximately 40 percentage of the number of vessel samples in the original vessel trajectory. *GTI* also performs similarly for vessel trajectories with 8000 and 4000 meters vessel trajectories, however, it outperforms *DGIVT* for the vessel trajectories with multiple gaps of 1000 and 2000 meters, in both the number of vessel samples produced and execution time. However, for the vessel trajectories with a single gap,



(a) The Original Vessel Trajectory.



(b) The Imputation by *DGIVT*.



(c) The Imputation by *GTI*.

Figure 11. Single gap of 4000 meters.

Table 5
DGIVT imputation results for the area of Aalborg harbor to Kattegat.

Vessel Trajectory gap	No. of Original <i>vs</i> avg.	No. of Reduced <i>vs</i> avg.	No. of Imputed <i>vs</i> avg.	Execution Time (s) avg.
Mult-8000	801	5	341	6.48
Mult-4000	801	8	344	6.48
Mult-2000	801	15	348	6.86
Mult-1000	801	28	351	7.09
Single-8000	801	519	619	7.80
Single-4000	801	631	669	7.90
Single-2000	801	703	708	8.22
Single-1000	801	734	721	7.52
Realistic	801	628	641	6.74

DGIVT is consistent in its imputation, while *GTI* is unable to create any results, as seen in both Table 6 and Figure 13. This is due to *GTI* using every vessel sample as a vertex in its graph creation, and as such, is unable to produce any results due to its computational complexity.

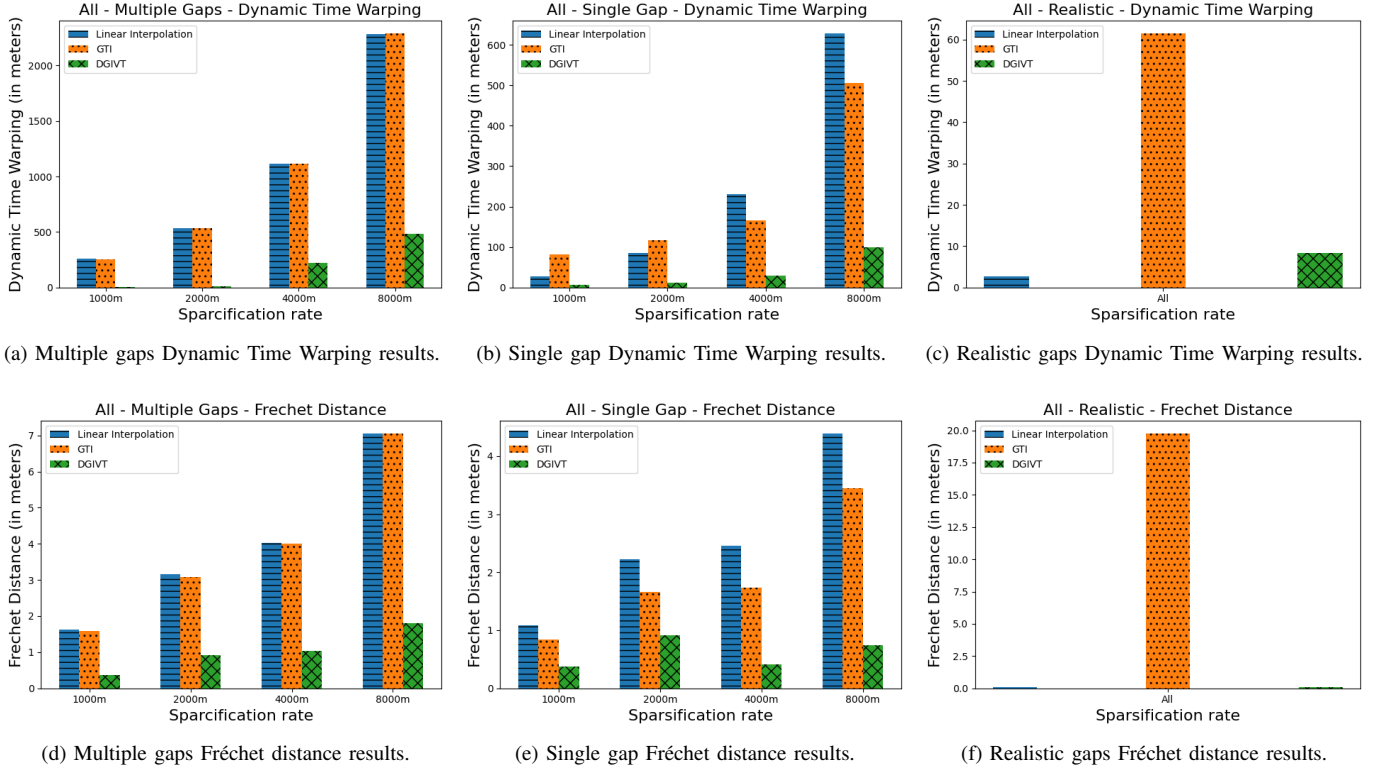


Figure 12. Dynamic Time Warping and Fréchet data for the 9 cells region. The y-axis is given in meters and the x-axis is the distance gap.

Table 6
GTI imputation results for the area of Aalborg harbor to Kattegat.

Vessel Trajectory gap	No. of Original	No. of Reduced	No. of Imputed	Execution Time (s)
Metric	vs avg.	vs avg.	vs avg.	avg.
Mult-8000	801	5	6	0.009
Mult-4000	801	8	9	0.017
Mult-2000	801	15	467	0.077
Mult-1000	801	28	934	0.220
Single-8000	801	519	N/A	N/A
Single-4000	801	631	N/A	N/A
Single-2000	801	703	N/A	N/A
Single-1000	801	734	N/A	N/A
Realistic	N/A	N/A	N/A	N/A

6. CONCLUSION

In this paper, we introduced *DGIVT*, a trajectory imputation framework to insert realistic vessel samples in vessel trajectories. The framework utilizes a data-driven approach that uses a regional grid, associating vessel samples from historic AIS data to each subregion in the grid. Within each subregion, we utilize a density sampling algorithm to maintain a spatial representative set of vessel samples. We then construct a directed graph for each subregion and proceed to impute vessel trajectories, by inserting realistic vessel samples between gaps, simulating as though no vessel samples are missing. The results show that *DGIVT* effectively accomplishes this task

by achieving DTW and Fréchet scores lower than both *Linear Interpolation* and *GTI* in trajectories with multiple and single gaps. However, in realistic trajectories where no substantial gaps exist, *Linear Interpolation* seems to fare better than *DGIVT*. We suspect this is an effect of the realistic trajectories structure, where no large gaps occur, and *DGIVT* inability to fill gaps for trajectories, where two consecutive vessel samples are near each other. For future development of *DGIVT*, we propose experimenting with individual threshold values for each cell in the vessel sample grid, to accommodate different regional differences. We furthermore suggest exploring the creation of graphs tailored to specific vessel types and applying imputation techniques to these graphs. This approach operates under the assumption that vessels belonging to the same vessel type, follow similar trajectories.

ACKNOWLEDGMENTS

We would like to thank Christian Søndergaard Jensen and Kristian Torp for their invaluable guidance and counseling throughout the project. We are also grateful for the AIS data and sea depth map provided by the Danish Maritime Authority, as well as the harbor data supplied by DIPAAL.

REFERENCES

- [1] U. N. C. O. T. A. DEVELOPMENT., *Review of Maritime Transport 2023*. UNITED NATIONS, 2023.
- [2] A. Artikis and D. Zissis, *Guide to maritime informatics*, 1st ed. Cham, Switzerland: Springer, 2021.
- [3] IALA, "Iala guideline 1082 an overview of ais edition 1.0," 2016, accessed May, 21th, 2024. [Online]. Available: https://www.e-navigation.nl/sites/default/files/1082-Ed.2-Overview-of-AIS_June2016-1.pdf
- [4] J. M. Mou, C. van der Tak, and H. Ligteringen, "Study on collision avoidance in busy waterways by using ais data," *Ocean Engineering*, vol. 37, no. 5, pp. 483–490, 2010. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S002980181000020X>
- [5] G. Pallotta, M. Vespe, and K. Bryan, "Traffic knowledge discovery from ais data," in *Proceedings of the 16th International Conference on Information Fusion*, 2013, pp. 1996–2003.
- [6] A. Alessandrini, F. Mazzarella, and M. Vespe, "Estimated time of arrival using historical vessel tracking data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 7–15, 2019.
- [7] K. Omholt-Jensen. (2021, May) Ais and the main categories of ais challenges. [Accessed March 22, 2024]. [Online]. Available: <https://www.maritimeoptima.com/insights/ais-and-the-main-categories-of-ais-challenges>
- [8] R. B. Games, "Introduction to a*," Jan 2020, accessed May, 20th, 2024. [Online]. Available: <https://www.redblobgames.com/pathfinding/a-star/introduction.html>
- [9] K. Isufaj, M. M. Elsharif, S. Abbar, and M. Mokbel, "Gti: A scalable graph-based trajectory imputation," in *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: <https://doi.org/10.1145/3589132.3625620>
- [10] M. M. Elsharif, K. Isufaj, and M. F. Mokbel, "Network-less trajectory imputation," in *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: <https://doi.org/10.1145/3557915.3560942>
- [11] B. B. Magnussen, N. Bläser, and H. Lu, "Daistin: A data-driven ais trajectory interpolation method," in *Proceedings of the 18th International Symposium on Spatial and Temporal Data*, ser. SSTD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 75–84. [Online]. Available: <https://doi.org/10.1145/3609956.3609961>
- [12] K. Zhang, Z. He, L. Zheng, L. Zhao, and L. Wu, "A generative adversarial network for travel times imputation using trajectory data," *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, no. 2, pp. 197–212, 2021. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12595>
- [13] C. Chen, S. Jiao, S. Zhang, W. Liu, L. Feng, and Y. Wang, "Tripimputor: Real-time imputing taxi trip purpose leveraging multi-sourced urban data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3292–3304, 2018.
- [14] M. T. Asif, N. Mitrovic, L. Garg, J. Dauwels, and P. Jaillet, "Low-dimensional models for missing data imputation in road networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 3527–3531.
- [15] M. Musleh and M. F. Mokbel, "Kamel: A scalable bert-based system for trajectory imputation," *Proc. VLDB Endow.*, vol. 17, no. 3, p. 525–538, nov 2023. [Online]. Available: <https://doi.org/10.14778/3632093.3632113>
- [16] Y. Shi, H. Gao, and W. Rao, "Tigan: Trajectory imputation via generative adversarial network," in *Advanced Data Mining and Applications*, X. Yang, H. Suhartanto, G. Wang, B. Wang, J. Jiang, B. Li, H. Zhu, and N. Cui, Eds. Cham: Springer Nature Switzerland, 2023, pp. 195–209.
- [17] Z. Guo, Y. Wan, and H. Ye, "A data imputation method for multivariate time series based on generative adversarial network," *Neurocomputing*, vol. 360, pp. 185–197, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231219308306>
- [18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
- [19] Llyods List Intelligence. The essential guide to the automatic identification system (ais). Accessed April, 26th, 2024. [Online]. Available: <https://www.lloydslistintelligence.com/knowledge-hub/data-storytelling/essential-guide-automatic-identification-system-ais-signals>
- [20] E. Maria, E. Budiman, Haviluddin, and M. Taruk, "Measure distance locating nearest public facilities using haversine and euclidean methods," *Journal of Physics: Conference Series*, vol. 1450, no. 1, 02 2020, copyright - © 2020. This work is published under <http://creativecommons.org/licenses/by/3.0/> (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2023-11-28. [Online]. Available: <https://www.proquest.com/scholarly-journals/measure-distance-locating-nearest-public/docview/2569099652/se-2>
- [21] D. Mount, "Lecture 19 geometric sampling, vc-

- dimension, and applications,” 2021, accessed May, 20th, 2024. [Online]. Available: <https://www.cs.umd.edu/class/fall2021/cmsc754/Lects/lect19-vc-dim.pdf>
- [22] C. Kingsford, “Kd-trees,” 2008, accessed May, 20th, 2024. [Online]. Available: <https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/kdtrees.pdf>
- [23] Cuemath, “Linear interpolation formula - derivation, formulas, examples,” accessed May, 24th, 2024. [Online]. Available: <https://www.cuemath.com/linear-interpolation-formula/>
- [24] J. R. Dan Margalit, “Interactive linear algebra,” accessed May, 20th, 2024. [Online]. Available: <https://textbooks.math.gatech.edu/ila/least-squares.html#:~:text=So%20a%20least%2Dsquares%20solution,difference%20b%20%E2%88%92%20Ax%20is%20minimized.>
- [25] MarineTraffic, “How often do the positions of the vessels get updated on marinetraffic?” accessed May, 22th, 2024. [Online]. Available: <https://help.marinetraffic.com/hc/en-us/articles/217631867-How-often-do-the-positions-of-the-vessels-get-updated-on-MarineTraffic>
- [26] E. Alizadeh, “An introduction to dynamic time warping,” accessed May, 22th, 2024. [Online]. Available: <https://builtin.com/data-science/dynamic-time-warping>
- [27] J. P. Figueira, “Fast discrete fréchet distance,” accessed May, 22th, 2024. [Online]. Available: <https://towardsdatascience.com/fast-discrete-fr%C3%A9chet-distance-d6b422a8fb77>

ACRONYMS

- AIS** Automatic Identification System. 1–3, 5, 11, 14, 17
- DGIVT** Depth-Map enhanced Graph Imputation for Vessel Trajectories. 5
- DTW** Dynamic Time Warping. 11, 13, 14
- ETA** Estimated Time of Arrival. 1, 2
- GAN** Generative Adversarial Network. 3
- GB** Gigabyte. 11
- GPS** Global Positioning System. 1–3
- IMO** International Maritime Organization. 1
- MMSI** Maritime Mobile Service Identity. 2, 3, 17
- NLP** Natural language processing. 3

APPENDIX A

AIS DATA PREPARATION

The study utilizes AIS data obtained from the Danish Maritime Authority. Although the available dataset covers the period from 2006 to 2024, our analysis specifically focuses on the data spanning 397 days, from March 1st, 2023, to April 1st, 2024.

A1 Initial Data Filtration

The 397 days of AIS encompasses a total of 5,339,615,793 vessel samples reports. To refine the scope of our analysis, we apply the following filtration criteria:

- Each vessel sample must contain a unique eight-digit MMSI.
- The vessel type of the vessel sample must correspond to one of the vessel types listed in Table 10.
- Only vessel reports with Class A vessels are considered.
- The longitude and latitude in the vessel sample must be within Danish maritime waters
- Draught value must not be negative, as this indicates the vessel sample does not belong to a vessel

While this paper primarily examines Class A vessels within Danish maritime waters, the proposed methodology is applicable to other regions and Class B vessels as well.

After applying the filtration criteria outlined above, the total number of vessel samples is reduced from 5,339,615,793 to 2,981,044,561. This reduction is detailed in Table 7.

Table 7
Amount of Vessel Samples after Filtrating invalid Vessel Samples

Days	Vessel Samples before Filtration	Vessel Samples after Filtration
Total	5,339,615,793	2,981,044,561
Min	9,064,616	6,029,364
Average	13,449,914	7,508,928
Max	31,593,836	9,524,507
Quantile 25%	11,090,954	7,148,215
Median	13,433,568	7,370,149
Quantile 75%	15,372,159	7,692,112


Table 8
Amount of Vessel Trajectories before and after Filtration on Draught

	Vessel Trajectories per Day	Vessel Trajectories After Filtration	Vessel Samples per Vessel Trajectory
Total	663,077	573,305	2,524,715,052
Min	1,250	1,079	1
Average	1,670	1,444	4,402
Max	2,492	2,167	43,491
Quantile 25%	1,459	1,261	701
Median	1,563	1,347	701
Quantile 75%	1,821	1,572	7,868

With the filtration of vessel samples completed, we extract vessel trajectories by grouping vessel samples based on MMSI and ordering them by timestamp. To ensure consistency within each vessel trajectory for vessels using the same MMSI, we separate vessel samples in the vessel trajectory that do not share the same length, width, name, and vessel type. Additionally, each vessel trajectory must comprise at least two vessel samples. Following these criteria, we identified a total of 663,007 vessel trajectories, as documented in Table 8

We use draught values to determine a vessel's capability to navigate to specific destinations. Therefore, we perform an additional filtration step, requiring that at least one vessel sample in each vessel trajectory has a non-negative draught value. Vessel trajectories not meeting this criterion are excluded, resulting in a final count of 573,305 vessel trajectories, corresponding to 2,524,715,052 vessel samples, detailed further in Table 8.

A2 Vessel Trajectory Splitting

The vessel trajectories include both moving and stationary periods, however, our focus is exclusively on the periods where the vessel is in motion. To identify whether a vessel is traveling, we utilize harbor data combined with the vessel's navigational status and speed over ground, such that vessel trajectories that travel between harbors with intervening stops, are split accordingly. Furthermore, vessel trajectories that are not traveling at all, are discarded. The detailed implementation of this approach can be accessed in our repository at [GitHub](#) 

After processing and splitting the vessel trajectories, we increase the number of vessel trajectories from 573,305 to 815,806, while decreasing the total number of vessel samples from 2,524,715,052 to 1,489,884,660, as described in Table 9.

Table 9
Vessel Trajectories after Splitting

	Vessel Trajectories after Split per Day	Vessel Samples per Vessel Trajectory	Distance Traveled in Meters per Vessel Trajectory
Total	815,806	1,489,884,660	80,985,367,191.31
Min	990	2	0.06
Average	2,055	1,826	99,270.38
Max	3,013	39,924	36,893,632.36
Quantile 25%	1,839	158	7,110.91
Median	2,025	651	29,506.38
Quantile 75%	2,282	2,507	129,460.45

For the total 815,806 vessel trajectories, Table 10 illustrates the distribution across different vessel types. The 'Passenger' category represents the majority of vessel trajectories, while 'Towing (Long/Wide)' appears least frequently.

Table 10
Total number of vessels

Vessel Type	Occurrences
Anti-Pollution	750
Cargo	188,675
Diving	1,029
Dredging	21,389
Fishing	50,920
Law Enforcement	5,859
Passenger	386,339
Pilot	39,685
Pleasure	1,418
Port Tender	7,244
Tanker	80,645
Towing	3,635
Towing (Long/Wide)	656
Tug	27,562
Total	815806

APPENDIX B

AALBORG TO KATTEGAT DYNAMIC TIME WARPING AND FRECHET DISTANCE DATA

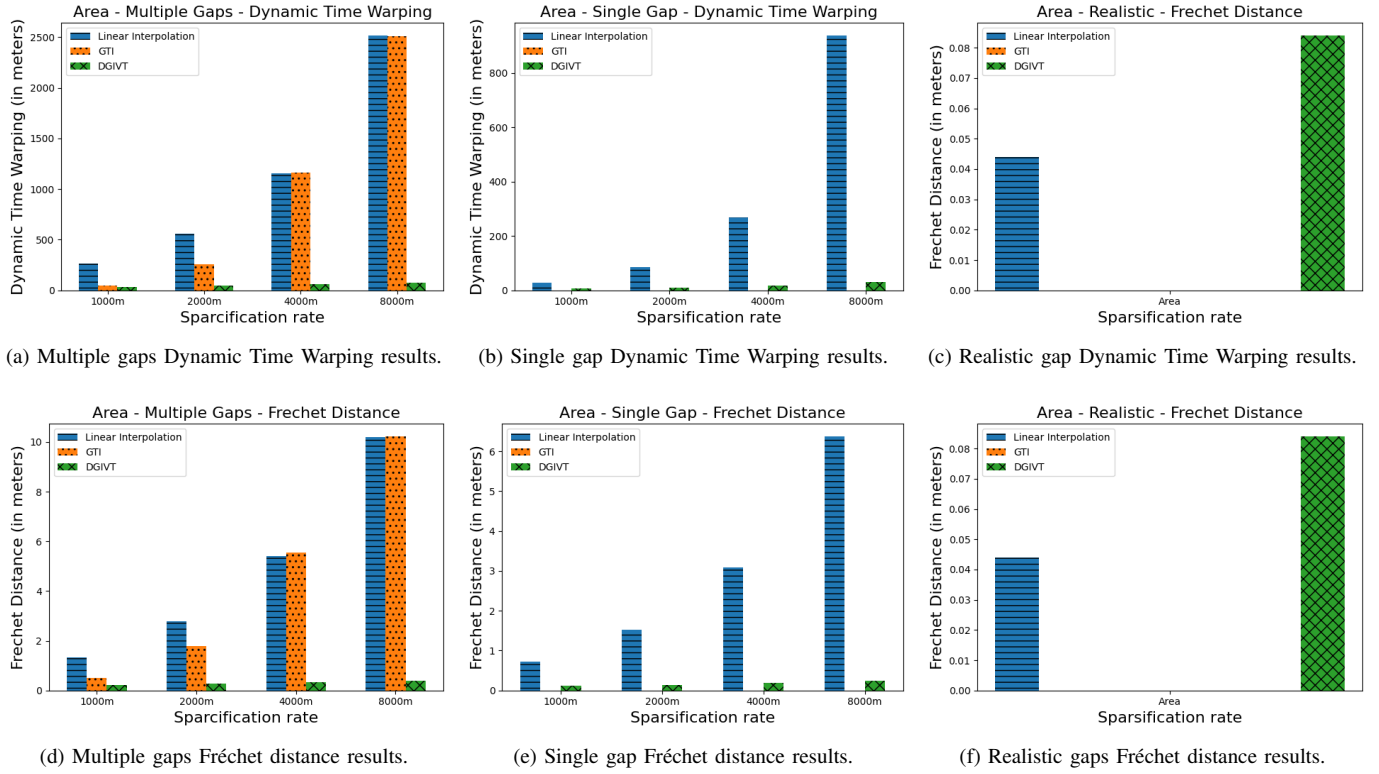


Figure 13. Dynamic Time Warping and Fréchet data for Aalborg to Kattegat region. The y-axis is given in meters and x-axis is the distance gap.