Effective Ship Trajectory Imputation with Multiple Coastal Cameras

Song Wu*†, Kristian Torp†, Alexandros Troupiotis-Kapeliaris‡, Dimitris Zissis‡, Esteban Zimányi*, Mahmoud Sakr*
*Université libre de Bruxelles, Brussels, Belgium
†Aalborg University, Aalborg, Denmark
‡University of the Aegean, Syros, Greece
*{song.wu, esteban.zimanyi, mahmoud.sakr}@ulb.be, †{songw, torp}@cs.aau.dk, ‡{alextroupi, dzissis}@aegean.gr

***Abstract*—The ship trajectories collected by the Automatic Identification System (AIS) are widely used in maritime applications. However, a significant issue with AIS data is that large AIS gaps occur. Existing trajectory imputation methods for AIS data have three main limitations: (1) the temporal aspect is ignored; (2) the methods fall short when dealing with complex ship movements; (3) the common-route assumption does not always hold. To overcome these limitations, we propose TrajImpMC, a tracking-based framework that uses polygon-based ship location estimates from multiple cameras to impute large AIS gaps. TrajImpMC combines speed constraints and Kalman filters, and can return imputed trajectories that contain both spatial and temporal information. Extensive experiments are conducted on real datasets. In terms of the quality of the imputed trajectories, TrajImpMC improves the RMSE errors by at least one order of magnitude over two existing state-of-the-art AIS imputation methods. In addition, a visual comparison shows that the imputed trajectories of TrajImpMC align very well with the real ship trajectories during AIS gaps. The code for this paper is available at: https://github.com/songwu0001/TrajImpMC.**

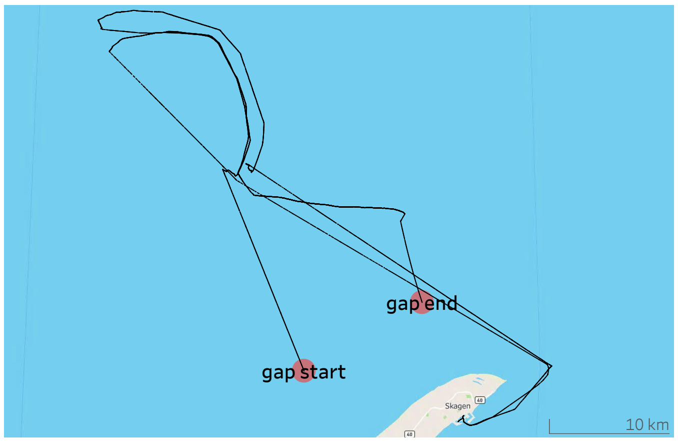


Fig. 1. A real trajectory that starts and ends at the Skagen harbor in Denmark. The two red circles show where a 5-hour gap exists, and the whereabouts of this ship during the gap period are thus unknown from AIS data.

I. Introduction

In the maritime domain, the Automatic Identification System (AIS) has become the most popular technology for ship tracking [1]. AIS allows ships to broadcast their location, speed, and navigational information to nearby ships, terrestrial stations, or satellites. In the past two decades, the extensive deployment of AIS worldwide has led to the collection of huge amounts of ship trajectory data. Such data plays a critical and indispensable role in numerous applications, such as constructing a maritime transport network [2], [3], estimating CO_2 emissions from ships [4], [5], and detecting fishing activities [6], [7].

However, despite its immense application value, a significant issue with AIS data is the large spatial and/or temporal gaps that often occur [8], [9]. These gaps result in consecutive AIS points that span a long spatial distance up to several kilometers or have a long time interval up to hours or even days. For example, Fig. 1 illustrates a trajectory in which a 5-hour gap exists. Large AIS gaps such as this pose challenges to applications whose accuracy is based on high-quality AIS trajectories [8], such as collision avoidance and trajectory prediction.

To handle AIS gaps, many trajectory imputation methods have been proposed to estimate the missing ship movement during an AIS gap [8]–[15]. These methods handle AIS gaps mainly from two perspectives: numerical analysis and historical data-driven [8]. Numerical analysis methods fill a gap either by applying well-established mathematical formulas, e.g., Lagrange interpolation [10] and cubic spline interpolation [12], or by considering the ship's kinematic information such as speed and acceleration [11]. Although these methods are efficient and easy to implement, they do not take into account geographic context. Therefore, they are prone to return unrealistic trajectories such as those that cross land [8]. In contrast, data-driven methods fill AIS gaps by discovering movement patterns from massive historical AIS data [8], [9], [13]–[15], and their interpolated trajectories conform better with reality.

Despite these advances, there are three main limitations for existing trajectory imputation methods.

1) *The temporal aspect is ignored.* Most existing methods focus only on the spatial path during an AIS gap, and interpolated trajectories thus lack time information. However, time information plays a key role in certain applications such as harbor surveillance [16]. Also, a ship may stop during the gap period, particularity in certain areas. In such cases, restoring time information is necessary to know if a ship is moving or not at any given time during the gap.

Last but not least, existing methods are mostly suited for imputing short-term gaps rather than large gaps like the one in Fig. 1.

2) *Existing methods fall short when dealing with complex ship movements*. Complex movements occur when a ship performs frequent maneuvers and exhibits movement patterns sufficiently different from the common ones. In such cases, the real movement of the ship during the gap cannot be accurately recovered by existing methods.
3) *The common-route assumption does not always hold [14]*. Unlike vehicle trajectories constrained by road networks, ships can move freely in the open sea, resulting in numerous ways to move between locations. This characteristic may break the common-route assumption in some data-driven methods, making it difficult or even impossible to find the most representative route to fill AIS gaps.

To overcome these limitations, a promising method is to perform data fusion with other sensors. The idea is that the ship locations during an AIS gap can still be monitored by other sensors. As far as we know, only few studies exist along this line of research. For example, the method proposed in [13] fills AIS gaps by detecting ship locations from Sentinel satellite images [17], [18]. However, these satellites have a revisit time of 2 to 3 days in high-coverage areas [13]. Therefore, the sparsity of satellite imagery restricts the application of this method in a large scale.

In this work, we explore the possibility of filling large AIS gaps by using data from coastal cameras, which are also widely used for monitoring ship traffic [19], [20]. Specifically, the idea is that by processing video data from multiple cameras, time-aware ship location estimates can be obtained for ships appearing in camera views. Since there can be many ships in camera views, the challenging task is how to identify the location estimates belonging to the ship that exhibits a large AIS gap. To this end, our contributions are as follows.

- We propose a framework named TrajImpMC (**Tra**jectory **Imp**utation using **M**ultiple **C**ameras). Using ship location estimates from multiple cameras, TrajImpMC transforms the trajectory imputation problem into a trajectory tracking problem. Next, TrajImpMC employs ship speed constraints and adapts Kalman filters to recover the most likely ship trajectory during a large AIS gap, and this trajectory contains both spatial and temporal information.
- Extensive experiments are conducted on real datasets to compare TrajImpMC with two state-of-the-art trajectory imputation methods for AIS data. Results show that TrajImpMC improves the RMSE error by at least one order of magnitude. Furthermore, visual analysis shows that the imputed trajectories by TrajImpMC align very well with the real ship trajectories during AIS gaps.

II. Related Work

Existing trajectory imputation methods for AIS data fall into three categories:

- Numerical analysis methods [10]–[12], [21]. These methods perform trajectory imputation by applying well-established mathematical formulas or by utilizing ship kinematic information. One common method in this category is the widely used linear interpolation [21]. This method assumes a constant speed during the gap and uses a straight line to fill the gap. Other studies use more advanced mathematical tools to fill the AIS gaps, such as Lagrange interpolation [10] and cubic spline interpolation [12]. In contrast, the study in [11] fills AIS gaps using kinematic information. In [11], the variation of acceleration during the gap is first modeled, then the variation of ship speed is derived from the acceleration function. Finally, the ship locations during the gap are derived from the speed function. The advantage of numerical methods is that they are efficient and easy to implement. However, since they do not take into account environmental restrictions, these methods tend to return unrealistic trajectories that may cross land [8]. Also, these methods fall short when dealing with complex ship movements.
- Data-driven methods [8], [9], [13]–[15]. Methods in this category fill AIS gaps by discovering movement patterns from massive historical AIS data. These methods use historical data in different ways.
  - Feature trajectory-based methods. The study in [14] proposes an algorithm called AISClean which fills an AIS gap by manipulating the most representative historical trajectory that passes near the gap locations. AISClean first retrieves a set of potential trajectories around the gap. Next, the potential trajectory that has the smallest Dynamic Time Warping (DTW) distance to the other potential trajectories is selected as the feature trajectory. Finally, an AIS gap is filled by translating the feature trajectory to match the gap.
  - Graph-based methods. These methods capture movement patterns by building a transition graph from historical AIS data [8], [13]. The nodes in this graph are either raw AIS points [8] or equal-sized spatial cells [13]. In [8], the edges are created between AIS points that satisfy both a neighborhood criterion and a direction criterion. In contrast, edges in [13] are created between neighboring cells and edge weights reflect the spatial distance and transition probabilities between cells. To fill a gap between a and b, both methods in [8] and [13] use the transition graph and the $A*$ algorithm to compute the shortest path between a and b.
  - Deep learning-based methods. Several studies have also emerged that fill AIS gaps by using deep-learning technology, which has gained great success in recent years. For example, the study in [15] employs the convolutional network U-Net [22] for ship trajectory reconstruction, and the study in [9] focuses on long-term ship trajectory imputation and proposes a physics-guided probabilistic diffusion model to fill AIS gaps.
- Methods based on data fusion. When AIS signal is lost, it is likely that ship movement can be monitored by other types of sensors, which play a complementary role to AIS.

Therefore, data fusion with other sensors can help determine the missing ship movement during an AIS gap. However, data fusion is a challenging task and only few studies exist. For example, the study in [13] fills AIS gaps by fusing with images captured by the Sentinel satellites [17], [18]. First, ship targets in the satellite images are detected. Then real-world locations of these targets are estimated. Next, the identities of these targets are determined by correlating targets with AIS data using the nearest-neighbor principle. However, the sparsity of satellite data restricts the application of this method at a large scale.

III. Problem Definition

Definition 1. (Trajectory). A trajectory $T = \{p_1, \cdots, p_n\}$ is a sequence of timestamped AIS points ordered by time. Each AIS point p_i $(1 \leq i \leq n)$ is a quadruple $(t,\ loc,\ sog,\ cog)$, where $p_i.t$ is the timestamp, $p_i.loc$ is the longitude and latitude, $p_i.sog$ is the speed over ground, and $p_i.cog$ is the course over ground of this point.

Definition 2. (Large AIS Gap). Given a trajectory T, a large AIS gap is defined as any two consecutive AIS points gap^{start} and gap^{end} in T satisfying:

$$gap^{end}.t - gap^{start}.t \geq \theta_t$$

where θ_t is a time threshold, and it can be adjusted based on the real application requirements. Clearly, gap^{start} is the last point before a gap and gap^{end} is the first point after a gap. Hereafter, $gap^{start}.loc$ and $gap^{end}.loc$ are also referred to as "gap locations"; $gap^{end}.t - gap^{start}.t$ is referred to as "gap duration"; and the Euclidean distance between $gap^{start}.loc$ and $gap^{end}.loc$ is referred to as "gap distance".

Problem Definition. Given a large AIS gap, trajectory imputation aims to recover the missing ship movement during this gap by adding points between gap^{start} and gap^{end}.

IV. Methodology

To overcome the limitations in existing trajectory imputation methods for AIS data, this work combines the ship location estimation method proposed in [23] and Kalman filter-based trajectory tracking [24].

Given multiple coastal cameras monitoring an area of interest, the method in [23] estimates the locations of ships inside the area by processing images captured by these cameras. Since there is no one-to-one correspondence between discrete pixel coordinates and continuous spatial (longitude/latitude) coordinates, the algorithm in [23] returns a set of polygons rather than points as the estimated ship locations, and these polygons do not share their interiors. In this way, the actual locations of most ships within the area are contained in these polygons.

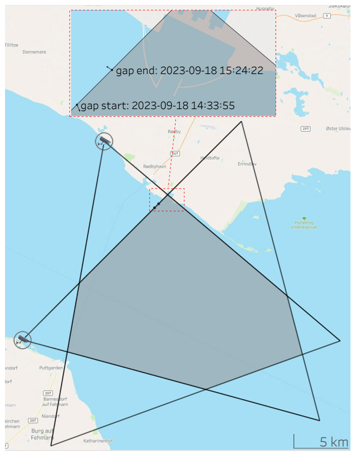
Based on the study in [23], this work transforms the trajectory imputation problem into a trajectory tracking problem during an AIS gap. The main idea is to use the polygon-based ship location estimates from cameras as extra knowledge. It can be inferred that a ship s must visit a sequence of polygons during the gap if s stays inside the monitored region all the time. The key points here are how to identify the sequence of polygons visited by s and from this polygon sequence how to construct a realistic trajectory that connects gap^{start} and gap^{end}. Fig. 2 illustrates the overall design of our proposed solution, which is composed of four steps: (1) Refinement of tracking space; (2) Kalman filter-based polygon sequence selection; (3) Shortest-path computation over the polygon sequence; and (4) Timestamp assignment. These steps are detailed as follows.

Algorithm 1: RefineTrackingSpace

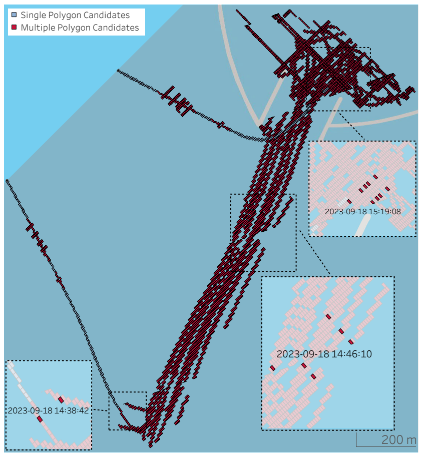
```
Input: $gap$: defines the imputation task
       $V$: the maximum ship speed
       $\Delta t$: the time step size
Output: $poly^{refi}$: the set of polygons after refinement
        at timestamps determined by $\Delta t$
1  $poly^{refi} \leftarrow dict()$
2  $ts \leftarrow \{\}$
3  $prev \leftarrow \{gap^{start}.loc\}$
4  $currT \leftarrow gap^{start}.t + \Delta t$
5  while $currT < gap^{end}.t$ do
6      $polys \leftarrow$ ship location estimates at $currT$
7      $polys^{MP} \leftarrow$ multi-pixel polygons in $polys$
8      $now \leftarrow \{\}$
9      foreach $p_1 \in polys^{MP}$ do
10         if $\exists p_2 \in prev : dist^{euc}(p_1, p_2) \leq V * \Delta t$ then
11             append $p_1$ to $now$
12     $poly^{refi}[currT] \leftarrow now$
13     $prev \leftarrow now$
14     append $currT$ to $ts$
15     $currT \leftarrow currT + \Delta t$
16 $next \leftarrow \{gap^{end}.loc\}$
17 $nextT \leftarrow gap^{end}.t$
18 for $t \in reverse(ts)$ do
19     $polys \leftarrow poly^{refi}[t]$
20     $polys' \leftarrow \{\}$
21     foreach $p_1 \in polys$ do
22         $\Delta t' \leftarrow nextT - t$
23         if $\exists p_2 \in next : dist^{euc}(p_1, p_2) \leq V * \Delta t'$ then
24             append $p_1$ to $polys'$
25     $poly^{refi}[t] \leftarrow polys'$
26     $next \leftarrow polys'$
27     $nextT \leftarrow t$
28 return $poly^{refi}$
```

A. Refinement of Tracking Space

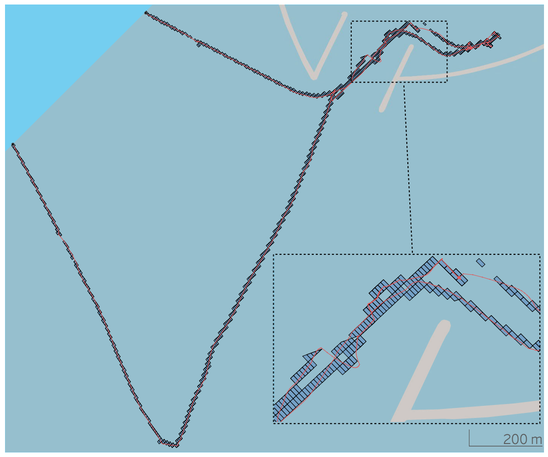
Combining the polygon-based ship location estimates from cameras over time gives the whole tracking space. However, not all polygons are important for filling the AIS gap of a given ship s. First, limited by its maximum movement capabilities, s is unable to reach some polygons. Second, trajectory imputation implies that s should finally reach gap^{end}, thus further limiting the polygons that s can visit over time. Therefore, this refinement step keeps only polygons that meet these two



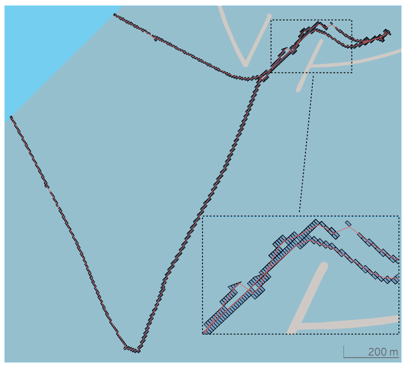
(a)



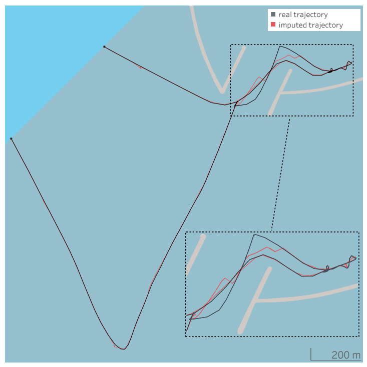
(b)



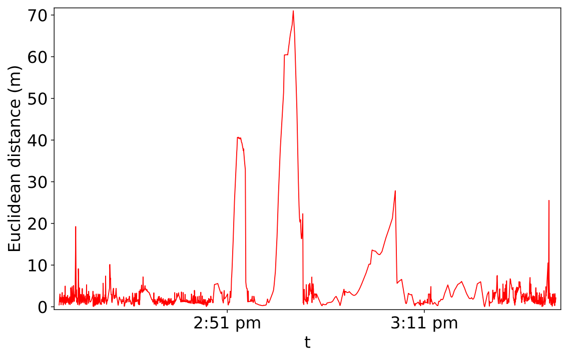
(c)



(d)



(e)



(f)

Fig. 2. Workflow of the proposed solution. **(a) Example of a multi-camera monitoring scenario and an AIS gap.** The grey polygon depicts the area of interest which is visible by both cameras. The ship is assumed to stay inside this polygon during the gap; **(b) Refinement of tracking space.** The possible ship locations over time are constructed using speed constraints and location estimates (as polygons) from multiple cameras. Close-up is shown for three randomly-chosen timestamps at which there are multiple choices for the ship location; **(c) Kalman filter-based polygon sequence selection.** Kalman filter is used to select the most probable sequence of polygons visited by a ship during an AIS gap. Here, the obtained sequence is shown in blue and the red linestring is the estimated ship trajectory by Kalman filter. Note that this linestring may not cross all of the blue polygons. **(d) Shortest path computation over the polygon sequence.** To make the imputed trajectory smooth and realistic, the shortest path is computed that begins at the gap start, visits the sequence of polygons in order, and finally reaches the gap end. **(e) Timestamp assignment and comparison with the real ship trajectory.** The Fréchet distance between the real trajectory and the imputed trajectory is 48.8 m, showing that the imputed trajectory is very close to the real trajectory. **(f) Distances between the imputed trajectory points and the real trajectory points over time.**

considerations. To achieve this, a speed-threshold method is applied. Algorithm 1 lists the pseudo-code.

Note that in this work, we apply an enhanced version of the algorithm proposed in [23] where the predicate of whether a pixel set is contained in another pixel set is not checked. Removing this check guarantees that the location of any ship in the monitored area is contained in one of the returned polygons.

Algorithm 1 consists of a forward pass and a backward pass. The input parameter Δt specifies the time granularity. The forward pass (lines#3-15) uses the maximum speed constraint to remove polygons that cannot be reached by a ship. Since this work uses a multi-camera scenario, we assume that the ship during an AIS gap is always visible to at least two cameras (see the grey polygon in Fig. 2a). This assumption is also motivated by the finding that using multiple cameras can improve the accuracy of location estimation by an order of magnitude than using only one camera [23]. Therefore, only polygons created by multiple pixels are taken into account in the forward pass (line#7). Note that in line#10, $dist^{euc}$ denotes the Euclidean distance. Next, the backward pass (lines#16-27) further removes polygons that do not lead a ship to gap^{end} in the end. Finally, $poly^{refi}$ is returned. It contains all the polygons that can be visited by a ship during an AIS gap.

B. *Kalman filter-based Polygon Sequence Selection*

In this step, the trajectory imputation problem is treated as a single-target trajectory tracking problem, where $poly^{refi}$ gives all the tracking possibilities. Since there can be multiple choices of polygons at a timestamp (see the close-up in Fig. 2b), the next challenge is how to comply with movement characteristics and identify the most likely polygon visited. To this end, this step employs the Kalman filter tracking to identify the most likely sequence of polygons visited by a ship during its gap.

Kalman filter [25] is a renowned state estimation algorithm for linear Gaussian systems. The algorithm is widely used for noise calibration [26] and robust tracking [27], [28]. Generally, the Kalman filter can optimally estimate the true states underlying a sequence of measurements. Measurements are usually the reported locations by sensors, e.g., GPS locations, which can be noisy and inaccurate. True states are the real locations (and/or speed, heading) of a moving object.

The formulation of a Kalman filter requires a motion model and a sensor model. The motion model specifies how the true state transits from one to another, and the sensor model specifies how a measurement is generated from the true state. Based on this, the Kalman filter works by recursively alternating between a prediction step and an update step. In the prediction step, a Kalman filter uses the motion model to predict the state at the current time step t based on the previous state at $t - \Delta t$. In the update step, the predicted state at t is updated based on a measurement, and this final updated state is considered to be the most likely true state at t. For more details on Kalman filter, we refer readers to [25], [29].

In this work, the polygon candidates at a timestamp cannot be directly used as measurements for a Kalman filter, because each polygon contains infinitely many locations that can be used as measurements. To solve this issue, we resort to the covariance matrix in a Kalman filter, which represents the level of uncertainty around a predicted location [26]. Therefore, we generate for each polygon the most probable measurement *w.r.t.* the predicted location and the covariance matrix. Two cases can be distinguished here. If the predicted location is inside a polygon, the measurement is then the predicted location itself. Otherwise, the point on the boundary of the polygon that has the smallest Mahalanobis distance is selected as the measurement. Fig. 3 illustrates an example of this technique.

By generating measurements from candidate polygons at each time step, a Kalman filter can now be created and run to select the most likely sequence of polygons visited. Algorithm 2 lists the pseudo code. A Kalman filter kf is initialized using the information in gap^{start} (line#2). Then in each iteration, kf selects the polygon as follows:

1) The speed threshold is used again (lines#5-6) to determine eligible polygons and the results are stored in $cand^{'}$.
2) Prediction is made for the current time step, and measurements are created for polygons in $candi^{'}$ (lines#7-14).
3) The best measurement is chosen using the Mahalanobis distance, and it is used to update kf (lines#15-16).
4) The polygon associated with the best measurement is recorded (line#18).

Fig. 2c shows the selected sequence of polygons in blue and the Kalman-filtered ship trajectory in red. One can observe that some red trajectory points fall outside the blue polygons; Therefore, this Kalman-filtered trajectory is not suitable to be returned as the final imputed trajectory. It is hypothesized that the imputed trajectory should pass all the selected polygons. To comply with this and make the imputed trajectory more smooth and realistic, in the next step we resort to compute the shortest path that visits a sequence of polygons in a given order.

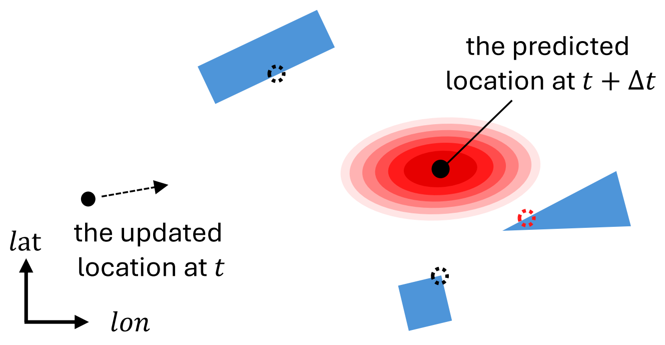

the predicted location at $t + \Delta t$
lat
the updated location at t
lon

Fig. 3. Measurement generation from polygons. The left solid dot shows the updated ship location by a Kalman filter at time t, and the right solid dot shows the predicted ship location at $t+\Delta t$. The concentric ellipses depict the associated uncertainty covariance matrix with this predicted location, where inner ellipses have a higher probability than outer ellipses. Suppose there are three polygon candidates (in blue) at $t + \Delta t$, then each polygon generates its most probable measurement (shown as hollow dots) *w.r.t.* the predicted location and the covariance matrix. In this example, the red hollow dot is the measurement used to update the predicted location, and the rightmost polygon will be considered to be the polygon visited by the ship at $t + \Delta t$.

Algorithm 2: PolygonSelection

```
Input: $gap$: defines the imputation task
       $poly^{refi}$: polygon candidates over time
       $V$: the maximum ship speed
       $\Delta t$: the time step size
Output: $polys$: the selected sequence of polygons
1  $polys \leftarrow dict()$
2  $kf \leftarrow InitializeKalmanFilter(gap^{start})$
3  $prev \leftarrow gap^{start}.loc$
4  for $t \in poly^{refi}$ do
5      $cand \leftarrow poly^{refi}[t]$
6      $cand' \leftarrow \{e \in cand \mid dist^{euc}(e, prev) \leq V * \Delta t\}$
7      $loc^p \leftarrow kf.pred(t)$
8      $measurements \leftarrow \{\}$
9      foreach $poly \in cand'$ do
10         if $poly.contains(loc^p)$ then
11             $loc^d \leftarrow loc^p$
12         else
13             $loc^d \leftarrow \arg\min_{e \in \partial poly} dist^{mah}(e, loc^p)$
14         append $(loc^d, poly)$ to $measurements$
15     $e \leftarrow \arg\min_{e \in measurements} dist^{mah}(e.loc^d, loc^p)$
16     $kf.update(t, e.loc^d)$
17     $prev \leftarrow e.poly$
18     $polys[t] \leftarrow e.poly$
19 return $polys$
```

C. Shortest-Path Computation over the Polygon Sequence

The shortest path we compute is well-studied in the field of computational geometry [30], [31] and referred to as the *Touring Polygons Problem*. When polygons are convex and disjoint, the shortest path can be efficiently computed using a data structure called *Last Step Shortest Path Map* (LSSPM) [30]. When the convex polygons can intersect, the time complexity of the solution increases by a factor of k compared to the non-intersecting case, where k is the number of polygons in the sequence [30], [31].

Algorithm 3 lists the pseudo code of this step. Note that a ship may take several time steps to pass a polygon; therefore, in line#1 we first extract the sequence of different polygons ($polys'$) using run-length encoding. In our scenario, the polygons are convex, but they are not necessarily disjoint because they may share part of their boundaries, see Fig. 2c. Therefore, a negative buffer technique is applied in lines#2-11 to ensure polygons are disjoint. This technique has two benefits: (1) lower time complexity for building the $LSSPM$; and (2) the shortest path is guaranteed to pass through the interior of the polygons rather than bounce occasionally at the polygon boundary. Note that just a small negative buffer size is enough for this technique; otherwise, the 'shortest' property of the path may be compromised. Next, the $LSSPM$ data structure is built (line#12), and the shortest path can be computed by issuing a query against $LSSPM$ (line#13). For details on $LSSPM$, we refer readers to [30]. Finally, the shortest path is returned along with $times$ which records the time period a ship is within a polygon in $polys'$.

Algorithm 3: ShortestPathOverPolygons

```
Input: $gap$: defines the imputation task
       $polys$: the selected sequence of polygons
       $buf^{neg}$: the negative buffer size
Output: $path$: the shortest path
        $times$: the number of consecutive
                 occurrences of polygons in $polys$
1  $polys', times \leftarrow RunLengthEncoding(polys)$
2  $polys'' \leftarrow \{\}$
3  foreach $poly \in polys'$ do
4      $buf' \leftarrow buf^{neg}$
5      while true do
6          $poly^{shrinked} \leftarrow Buffer(poly, buf')$
7          if $poly^{shrinked}$ is not empty then
8              append $poly^{shrinked}$ to $polys''$
9              break
10         else
11             $buf' \leftarrow buf'/2$
12 $LSSPM \leftarrow BuildLSSPM(gap^{start}.loc, polys'')$
13 $path \leftarrow LSSPM.query(gap^{end}.loc)$
14 return $path, times$
```

D. Timestamp Assignment

This last step adds the time information to the imputed trajectory points, and Algorithm 4 lists the pseudo code. On one hand, the $times$ given by Algorithm 3 records the number of time steps that the ship uses to pass each polygon. On the other hand, the theory in [30] states that a point in the $path$ given by Algorithm 3 is the very first point at which the

Algorithm 4: TimestampAssignment

```
Input: $gap$: defines the imputation task
       $path$: the shortest path over polygons
       $times$: the number of time steps for polygons
       $\Delta t$: the time step size
Output: $traj$: the imputed spatio-temporal trajectory
1  $traj \leftarrow \{gap^{start}\}$
2  $ts \leftarrow gap^{start}.t$
3  foreach $idx, p \in enumerate(path)$ do
4      $ts \leftarrow ts + \Delta t$
5      append $[ts, p]$ to $traj$
6      if $times[idx] > 1$ then
7          if $idx + 1 < len(path)$ then
8              $p^{next} \leftarrow path[idx + 1]$
9          else
10             $p^{next} \leftarrow gap^{end}.loc$
11         $k \leftarrow times[idx] - 1$
12         $points^{middle} \leftarrow Lint(p, p^{next}, k)$
13         foreach $p^{middle} \in points^{middle}$ do
14             $ts \leftarrow ts + \Delta t$
15             append $[ts, p^{middle}]$ to $traj$
16 append $gap^{end}$ to $traj$
17 return $traj$
```

shortest path enters the corresponding polygon. Therefore, if the ship uses more than one time step to pass a polygon, say $k+1$, then k additional evenly distributed points are generated using linear interpolation ($Lint$ in line#12) between the entry point of this polygon and its successor on the path. In this way, we can return a fine-grained spatio-temporal trajectory.

V. Experiments

This section conducts experiments to compare different algorithms for imputing large AIS gaps. Our algorithm TrajImpMC is implemented in Python and is available on Github.[1] The experiments were run on a Linux server equipped with 256 GB RAM and AMD Epyc Genoa CPU at 3.7GHz.

A. Datasets and Pre-processing

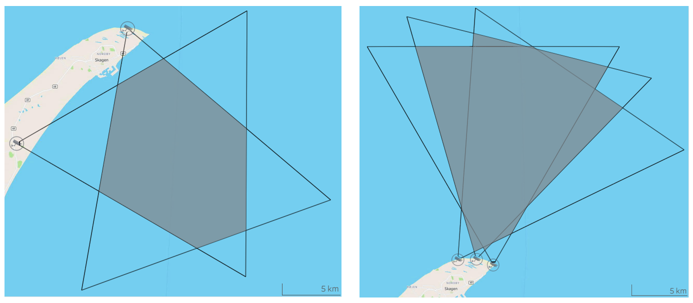
Datasets. In the experiments, the AIS data published by the Danish Maritime Authority is used.[2] This data is freely available and used in many studies [5], [7], [9], [32], [33]. Specifically, the region of interest is selected to be around Skagen, which is the north tip of Denmark. This region is chosen because it is one of the busiest water channels in the world, mixing traffic that transits between the North Sea and the Baltic Sea or between Norway and Denmark. As a result, ships in this region exhibit relatively complex movements, making the trajectory imputation task more challenging.

In this work, two multi-camera settings are designed: InsideSkagen and OutsideSkagen. The InsideSkagen setting

[1]https://github.com/songwu0001/TrajImpMC
[2]https://web.ais.dk/aisdata/

TABLE I. Configuration of the two multi-camera settings in the experiments. Note that $cam.\theta^H$ is the horizontal orientation of a camera.

name	$cam1.\theta^H$	$cam2.\theta^H$	$cam3.\theta^H$	$(cam1.lon,\ cam1.lat)$	$(cam2.lon,\ cam2.lat)$	$(cam3.lon,\ cam3.lat)$
InsideSkagen	90°	160°	-	(10.460241, 57.662182)	(10.622148, 57.751757)	-
OutsideSkagen	0°	14°	34°	(10.649634, 57.746461)	(10.623467, 57.751201)	(10.594881, 57.750756)



(a) InsideSkagen (b) OutsideSkagen

Fig. 4. Two multi-camera settings are used in the experiments. The grey polygon represents the area that can be monitored by at least two cameras.

monitors the shipping traffic in the anchorage area of the Skagen harbor, whereas the OutsideSkagen setting monitors the shipping traffic just outside Skagen. Fig. 4 depicts the settings, in which the grey polygon represents the area that is visible by at least two cameras. Table I shows the configuration of these two multi-camera settings. The configuration is designed such that an adequate coverage of the shipping traffic in the region of interest can be achieved. The other parameters shared by the two settings are fixed as follows: the vertical orientation of a camera is set as -2°; the camera height *w.r.t.* the sea surface is set as 100 meters; the horizontal field of view of cameras is set as 60°; the vertical field of view of cameras is set as 35.98°; the maximum monitoring range of cameras is set as 20 kilometers; and the video resolution captured by cameras is set as 1,920 by 1,080 pixels. Most of these parameter values are the same as in the previous study [23].

In the experiments, one-month AIS data for September 2024 is used as the training dataset to construct the baseline methods introduced later. One-week of AIS data from October 1st to October 7th, 2024 is used as the test dataset to simulate the ship location estimates from cameras, generate trajectory imputation tasks, and compare results from different imputation methods.

Pre-processing. The following pre-processing steps are applied to the datasets:

1) Spatial filtering. AIS messages are discarded whose location falls outside the area of interest. For the training dataset, this area of interest is shown as the grey polygons in Fig. 4. For the test dataset, this area of interest refers to the union of the monitored regions by all cameras.
2) Type filtering. Only AIS messages of type ‘Class A’ or ‘Class B’ are kept. These are the two main types of AIS data that contain ship location reports.
3) Duplicate removal. If multiple AIS messages have the same timestamp and MMSI, then one of them is randomly chosen and retained, and the other duplicates are removed.
4) Trajectory extraction. Each MMSI number in the AIS data is treated as a different ship. For each ship, a new trajectory is created whenever an AIS point has a time difference larger than 10 minutes or its speed exceeds 50 knots per hour *w.r.t.* its predecessor.
5) Short trajectory removal. Trajectories with a duration below 10 seconds are discarded because they are too short to be useful for analysis.

Table II shows statistics for the two datasets after pre-processing. An obvious difference between the two multi-camera settings is that the InsideSkagen setting contains fewer trajectories than the OutsideSkagen setting; however, trajectories in the InsideSkagen setting have a much longer duration than those in the OutsideSkagen setting.

TABLE II. Dataset characteristics after pre-processing

dataset	setting	# of points	# of traj.	avg. traj. duration
training	InsideSkagen	2,167,404	2,788	5.79 hours
	OutsideSkagen	1,101,775	4,839	0.59 hours
test	InsideSkagen	406,131	919	4.13 hours
	OutsideSkagen	448,484	1,179	0.96 hours

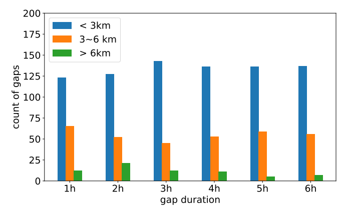
B. Trajectory Imputation Task Generation

In this work, we assume that during an AIS gap, a ship always stays inside the area that can be monitored by at least two cameras. Therefore, the imputation tasks in the experiments are generated randomly from the sub-trajectories that pass through the grey polygons in Fig. 4. From the pre-processed trajectories, 581 sub-trajectories are obtained for the InsideSkagen setting, and 1,062 sub-trajectories are obtained for the OutsideSkagen setting.

For each setting, we create 6 groups of imputation tasks that have a gap duration from 1 to 6 hours respectively. Each group contains 200 tasks which are uniformly sampled from valid gap candidates in the sub-trajectories. In this way, each task represents a different AIS gap to be imputed.

For each sub-trajectory T^{sub}, the valid gap candidates in T^{sub} are determined as follows. First, any trajectory window in T^{sub} with the desired duration (from 1 to 6 hours) is considered a gap candidate.[3] Second, for a gap candidate to be valid, two requirements should be satisfied: the distance between the gap start and the gap end is at least 500 meters, and the sinuosity of the trajectory window is larger than a threshold (1.1 is used in this work). These two requirements make sure that the generated gaps are not as easy as straight lines for which the simple linear interpolation is enough. Furthermore, it is worth noting that imputation tasks longer

[3]The term ‘trajectory window’ also means a sub-trajectory from a larger trajectory. It is used here to avoid confusion with the ‘sub-trajectories’ that pass through the grey polygons in Fig. 4.


(a) InsideSkagen

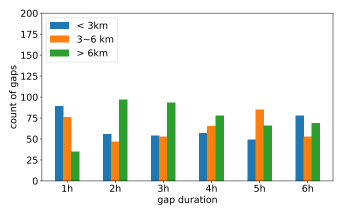
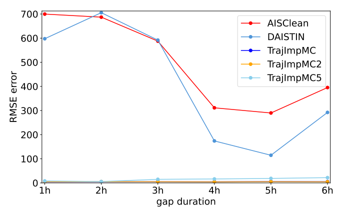
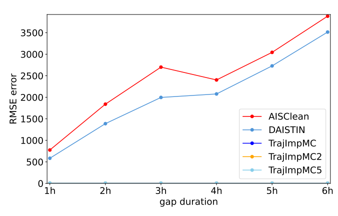
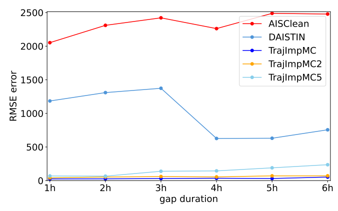

(b) OutsideSkagen

Fig. 5. Distribution of gap imputation tasks based on spatial distance


(a) InsideSkagen, DTW-based


(b) OutsideSkagen, DTW-based


(c) InsideSkagen, Fréchet-based

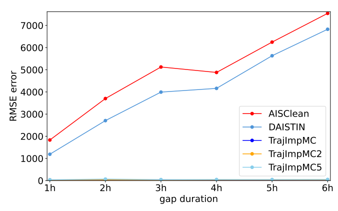

(d) OutsideSkagen, Fréchet-based

Fig. 6. RMSE for all methods

than 6 hours are excluded because in data exploration we observe that valid gap candidates longer than 6 hours are rare in the OutsideSkagen setting.

Fig. 5 shows the distribution of the imputation tasks based on the spatial distance of an AIS gap. Clearly, most of the imputation tasks in the InsideSkagen setting have a gap distance below 3 km, whereas imputation tasks in the OutsideSkagen setting have a more balanced distribution between the three distance ranges. This difference results from the large anchorage area of the Skagen harbor, which implies that a ship tends to stop for a longer time in the InsideSkagen setting than in the OutsideSkagen setting. On the other hand, this difference also indicates the diversity of the generated imputation tasks, and this diversity allows the evaluation of trajectory imputation algorithms in various cases.

C. Baseline Methods and Evaluation Metrics

Baseline Methods. For comparison, we use two state-of-the-art trajectory imputation methods for AIS data:

- DAISTIN [8]. DAISTIN imputes a gap by computing the shortest path over a graph built from massive historical AIS data. For the sake of scalability, DAISTIN uses a technique called geometric sampling to select representative points for graph construction. Since AIS points in the InsideSkagen setting are more concentrated than in the OutsideSkagen setting, we have sampled 10% of AIS points for graph construction in the InsideSkagen setting and 30% of AIS points for graph construction in the OutsideSkagen setting.
- AISClean [14]. AISClean also uses historical AIS data for trajectory imputation. Specifically, AISClean imputes a gap by translating and scaling the most representative historical AIS trajectory that passes near the gap locations. For our datasets, a threshold radius of 250 meters is used to determine if an AIS trajectory passes near the gap locations, and this threshold is half of the minimum spatial distance used for imputation task generation in Section V-B.

Note that both DAISTIN and AISClean may fail when there are no historical AIS data near the gap locations or the AIS data nearby is sparse. When this happens, we use a simple linear interpolation method as the fallback option for both DAISTIN and AISClean.

For our proposed algorithm TrajImpMC, the maximum speed of a ship is set as 50 knots, which is the same as in the pre-processing steps and also used in previous studies [34], [35]. The negative buffer size is set as -0.01 meter, and it works properly in the experiments. Furthermore, we compare three variants of TrajImpMC for which the time step size Δt is set as 1 second, 2 seconds, and 5 seconds, respectively. Hereafter, these variants are denoted as TrajImpMC, TrajImpMC2, and TrajImpMC5.

Metrics. To evaluate the quality of the imputed trajectories, we use two popular trajectory similarity measures: Fréchet distance [36], and normalized DTW distance [37]. Fréchet distance focus on the largest deviation between two trajectories, whereas the DTW distance measures the overall trajectory similarity by computing the sum of distances between matching points along a warping path. Since the original DTW distance depends on the number of points in the input trajectories, in this work, we compute the normalized DTW distance by using the length of the optimal warping path.

For each imputation task, these two distance measures are computed between the real trajectory during the gap and the imputed trajectories. The Root Mean Squared Error (RMSE) [8] is then reported for each distance measure and for each group of imputation tasks.

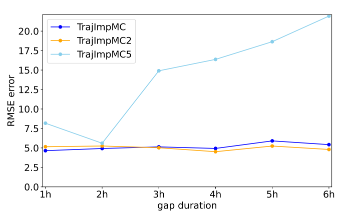
TABLE III. Overall RMSE errors based on DTW and Fréchet distances

	AISClean	DAISTIN	TrajImpMC	TrajImpMC2	TrajImpMC5
DTW	1893.8	1625.2	4.3	4.5	11.6
Fréchet	4040.8	3249.0	32.9	47.5	114.2

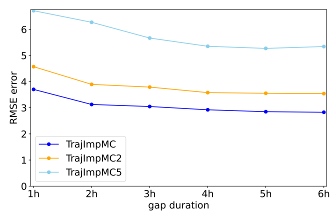
D. Comparison of Trajectory Imputation Results

RMSE Analysis. Fig. 6 shows the RMSE errors for the algorithms compared. Overall, RMSE errors of the three variants of TrajImpMC are significantly lower than the two baselines AISClean and DAISTIN. Table III shows the overall RMSE errors under the two settings. Clearly, the TrajImpMC framework reduces the RMSE error by at least one order of magnitude. This substantial improvement shows the advantage of using data from other sensors for imputing large AIS gaps.

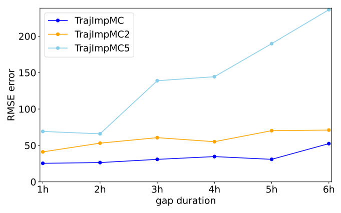
To investigate the effect of the time step Δt, Fig. 7 shows in more detail the RMSE errors of the three variants TrajImpMC,



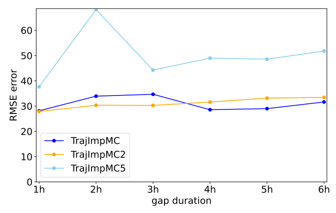
(a) InsideSkagen, DTW-based



(b) OutsideSkagen, DTW-based



(c) InsideSkagen, Fréchet-based



(d) OutsideSkagen, Fréchet-based

Fig. 7. RMSE for TrajImpMC, TrajImpMC2 and TrajImpMC5

TrajImpMC2, and TrajImpMC5. On one hand, TrajImpMC and TrajImpMC2 perform better than TrajImpMC5 in all cases. On the other hand, TrajImpMC has smaller RMSE errors than TrajImpMC2 in Fig. 7b and Fig. 7c; however, they are tied in Fig. 7a and Fig. 7d.

Furthermore, an interesting point to note is that a longer gap duration does not necessarily lead to a larger RMSE error. For example, when the gap duration varies from 1 to 6 hours, the RMSE shows a downward trend in Fig. 7b and fluctuates in Fig. 7d. This reveals that the complexity of filling AIS gaps is probably more affected by the real traffic situation during the gap period. For example, if the ship s underlying an AIS gap has more other ships in its vicinity, it will then be more challenging for the tracking-based TrajImpMC framework to correctly identify the ship locations for s.

Visualization Analysis. To get a better understanding of the imputed trajectories by different algorithms, Fig. 8 shows representative examples. After a closer examination, the following insights can be made:

- The baselines AISClean and DAISTIN cannot deal with complex ship movements. All the real trajectories in Fig. 8 exhibit complex movement patterns, and the imputed trajectories by AISClean and DAISTIN deviate largely from the actual ship movements. Such results highlight the difficulty of imputing large AIS gaps in an open-sea scenario. Since not constrained by a road network, there are numerous ways for a ship to move between locations. Therefore, it is hard and even impossible to find a historical trajectory (as done in AISClean) that qualifies as the most representative route between the gap locations. Similarly, computing the shortest path (as done in DAISTIN) is also unlikely to recover the missing ship trajectory during a large AIS gap.
- Most of the time, TrajImpMC, TrajImpMC2, and TrajImpMC5 return similar results, and their imputed trajectories align well with the real ship trajectory.
- Occasionally, TrajImpMC2 and TrajImpMC5 give rise to noticeable deviations in the imputed trajectories. For example, both TrajImpMC2 and TrajImpMC5 lose track of the second half of the real ship trajectory in Fig. 8c, and TrajImpMC5 loses track of the real ship trajectory from the beginning in Fig. 8d. These cases mainly occur when the tracking module in the TrajImpMC framework is misled by the locations of other ships. However, it is worth noting that such cases are rare in the overall experimental results, and they are added in Fig. 8 for the sake of completeness.

Runtime Analysis. Fig. 9 shows the average runtime for imputing AIS gaps by each algorithm. The baselines AISClean and DAISTIN are much faster than the three variants of TrajImpMC. This is because AISClean and DAISTIN only focus on the spatial aspect and do not depend on the gap duration at all. On the contrary, the TrajImpMC framework is based on tracking, and its execution time increases linearly *w.r.t.* the gap duration. Given the superior quality of the imputed trajectories by the three TrajImpMC variants, we argue that it is worthwhile for the TrajImpMC framework to exchange execution time for the quality of imputed trajectories.

In addition, Fig. 9 shows that using larger values for the time step Δt can effectively reduce the execution time, because larger Δt values lead to fewer tracking steps in the TrajImpMC framework. Combined with previous quality analysis, a time step of 2 seconds seems a recommended value to balance between execution time and the quality of imputed trajectories.

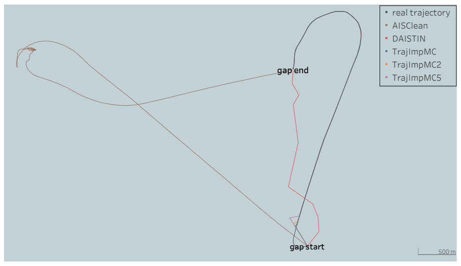
VI. Conclusion

To overcome the limitations of existing trajectory imputation methods for AIS data, we propose TrajImpMC, a tracking-based framework for imputing large AIS gaps. Built on top of time-aware and polygon-based ship location estimates from multiple coastal cameras, TrajImpMC works in four steps. First, TrajImpMC uses speed constraints to construct the tracking space, which contains all the possibilities for the missing ship trajectory. Second, TrajImpMC employs a Kalman filter to find the most likely sequence of polygons that a ship has passed during its AIS gap. Third, the shortest path is computed that visits in turn this sequence of polygons. Finally, time information is restored such that the imputed trajectory contains both spatial and temporal information. Extensive experiments are conducted on real datasets to compare TrajImpMC with two state-of-the-art trajectory imputation methods for AIS data. Results show that TrajImpMC reduces the RMSE error by at least one order of magnitude. In addition, visual analysis shows that the imputed trajectories by TrajImpMC are the closest to the real ship trajectories during AIS gaps.

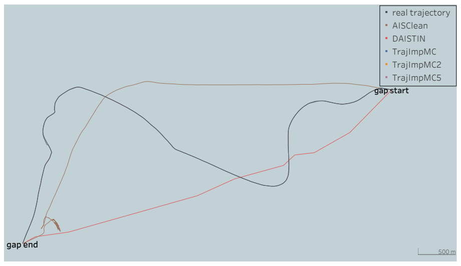
For future work, we will design techniques to further optimize the execution time of TrajImpMC. Also, it is interesting to investigate how TrajImpMC can be extended to deal with the situation where a ship is only monitored by one camera during its AIS gap.

Acknowledgment

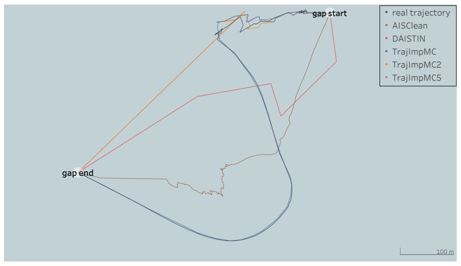
This publication has been developed under the framework of the “Data Engineering for Data Science” (DEDS) project



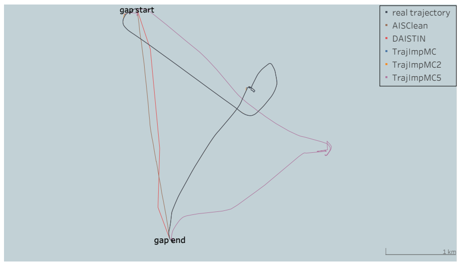
(a) InsideSkagen: gap duration = 1 hours



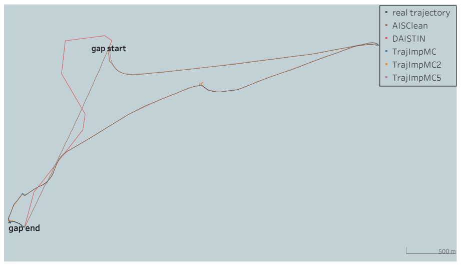
(b) InsideSkagen: gap duration = 2 hours



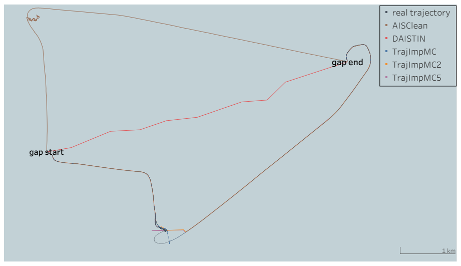
(c) InsideSkagen: gap duration = 3 hours



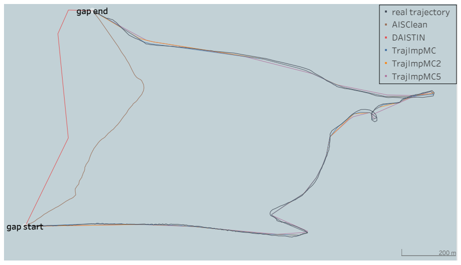
(d) InsideSkagen: gap duration = 4 hours



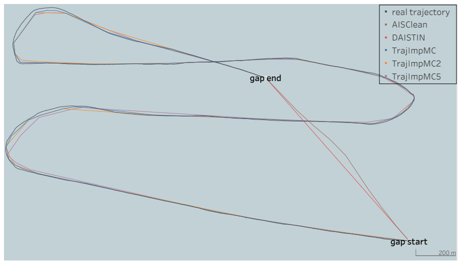
(e) InsideSkagen: gap duration = 5 hours



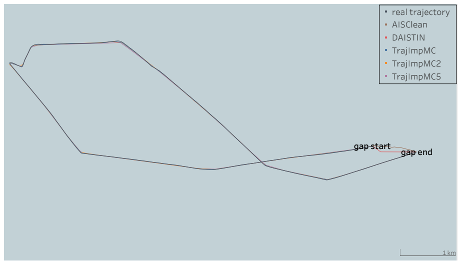
(f) InsideSkagen: gap duration = 6 hours



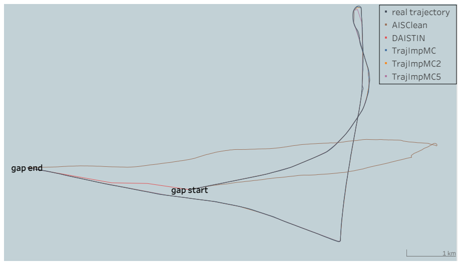
(g) OutsideSkagen: gap duration = 1 hours



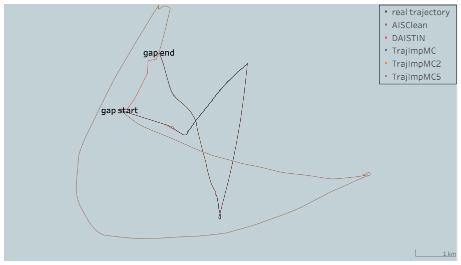
(h) OutsideSkagen: gap duration = 2 hours



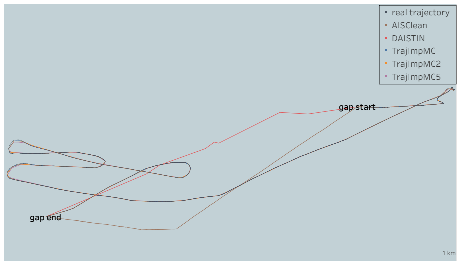
(i) OutsideSkagen: gap duration = 3 hours



(j) OutsideSkagen: gap duration = 4 hours

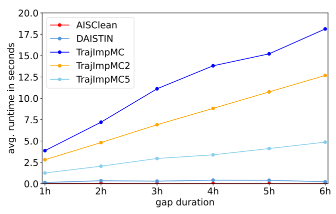


(k) OutsideSkagen: gap duration = 5 hours

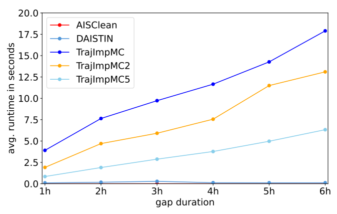


(l) OutsideSkagen: gap duration = 6 hours

Fig. 8. Examples of trajectory imputation results. One example is chosen for each combination of multi-camera setting and gap duration. The imputed trajectories by TrajImpMC, TrajImpMC2, and TrajImpMC5 may highly overlap with the real trajectory when they are very close to each other.



(a) InsideSkagen



(b) OutsideSkagen

Fig. 9. Runtime analysis

that has received funding from the European Union's Horizon 2020 programme (call identified: H2020-MSCA-ITN-EJD-2020) under grant agreement No 955895. Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

References

[1] K. Bereta, K. Chatzikokolakis, and D. Zissis, "Maritime reporting systems," *Guide to Maritime Informatics*, pp. 3–30, 2021.

[2] L. Eljabu, M. Etemad, and S. Matwin, "Spatial clustering method of historical ais data for maritime traffic routes extraction," in *2022 IEEE International Conference on Big Data (Big Data)*, 2022, pp. 893–902.

[3] N. Bläser, B. B. Magnussen, G. Fuentes, H. Lu, and L. Reinhardt, "Matnec: Ais data-driven environment-adaptive maritime traffic network construction for realistic route generation," *Transportation Research Part C: Emerging Technologies*, vol. 169, p. 104853, 2024.

[4] T. Hensel, C. Ugé, and C. Jahn, "Green shipping: using ais data to assess global emissions," in *Sustainability Management Forum—NachhaltigkeitsManagementForum*, vol. 28, no. 1. Springer, 2020, pp. 39–47.

[5] S. Wu, K. Torp, M. Sakr, and E. Zimanyi, "Evaluation of vessel CO_2 emissions methods using ais trajectories," in *Proceedings of the 18th International Symposium on Spatial and Temporal Data (SSTD)*. ACM, 2023, pp. 65–74.

[6] S. Arasteh, M. A. Tayebi, Z. Zohrevand, U. Glässer, A. Y. Shahir, P. Saeedi, and H. Wehn, "Fishing vessels activity detection from longitudinal ais data," in *Proceedings of the 28th International conference on advances in geographic information systems*, 2020, pp. 347–356.

[7] S. Wu, E. Zimányi, M. Sakr, and K. Torp, "Semantic segmentation of ais trajectories for detecting complete fishing activities," in *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2022, pp. 419–424.

[8] 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*, 2023, pp. 75–84.

[9] Z. Zhang, Z. Fan, Z. Lv, X. Song, and R. Shibasaki, "Long-term vessel trajectory imputation with physics-guided diffusion probabilistic model," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 4398–4407.

[10] I. Kontopoulos, I. Varlamis, and K. Tserpes, "A distributed framework for extracting maritime traffic patterns," *International Journal of Geographical Information Science*, vol. 35, no. 4, pp. 767–792, 2021.

[11] S. Guo, J. Mou, L. Chen, and P. Chen, "Improved kinematic interpolation for ais trajectory reconstruction," *Ocean Engineering*, vol. 234, p. 109256, 2021.

[12] B. Zaman, D. Marijan, and T. Kholodna, "Interpolation-based inference of vessel trajectory waypoints from sparse ais data in maritime," *Journal of Marine Science and Engineering*, vol. 11, no. 3, p. 615, 2023.

[13] A. Troupiotis-Kapeliaris, D. Zissis, K. Bereta, M. Vodas, G. Spiliopoulos, and G. Karantaidis, "The big picture: An improved method for mapping shipping activities," *Remote Sensing*, vol. 15, no. 21, p. 5080, 2023.

[14] M. Liang, J. Su, R. W. Liu, and J. S. L. Lam, "Aisclean: Ais data-driven vessel trajectory reconstruction under uncertain conditions," *Ocean Engineering*, vol. 306, p. 117987, 2024.

[15] S. Li, M. Liang, X. Wu, Z. Liu, and R. W. Liu, "Ais-based vessel trajectory reconstruction with u-net convolutional networks," in *2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*. IEEE, 2020, pp. 157–161.

[16] M. J. Loomans, P. H. de With, and R. G. Wijnhoven, "Robust automatic ship tracking in harbours using active cameras," in *2013 IEEE International Conference on Image Processing*. IEEE, 2013, pp. 4117–4121.

[17] D. Geudtner, R. Torres, P. Snoeij, M. Davidson, and B. Rommen, "Sentinel-1 system capabilities and applications," in *2014 IEEE geoscience and remote sensing symposium*. IEEE, 2014, pp. 1457–1460.

[18] F. Gascon, E. Cadau, O. Colin, B. Hoersch, C. Isola, B. L. Fernández, and P. Martimort, "Copernicus sentinel-2 mission: products, algorithms and cal/val," in *Earth observing systems XIX*, vol. 9218. SPIE, 2014, pp. 455–463.

[19] E. Gülsoylu, P. Koch, M. Yıldız, M. Constapel, and A. P. Kelm, "Image and ais data fusion technique for maritime computer vision applications," *arXiv preprint arXiv:2312.05270*, 2023.

[20] Y. Guo, R. W. Liu, J. Qu, Y. Lu, F. Zhu, and Y. Lv, "Asynchronous trajectory matching-based multimodal maritime data fusion for vessel traffic surveillance in inland waterways," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[21] S. Mao, E. Tu, G. Zhang, L. Rachmawati, E. Rajabally, and G.-B. Huang, "An automatic identification system (ais) database for maritime trajectory prediction and data mining," in *Proceedings of ELM-2016*. Springer, 2018, pp. 241–257.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[23] S. Wu, A. Troupiotis-Kapeliaris, D. Zissis, K. Torp, E. Zimányi, and M. Sakr, "Uncertainty-aware ship location estimation using multiple cameras in coastal areas," in *2024 25th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2024, pp. 109–118.

[24] J. Barr, O. Harrald, S. Hiscocks, N. Perree, H. Pritchett, S. Vidal, J. Wright, P. Carniglia, E. Hunter, D. Kirkland *et al.*, "Stone soup open source framework for tracking and state estimation: enhancements and applications," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXXI*, vol. 12122. SPIE, 2022, pp. 43–59.

[25] G. Welch and G. Bishop, "An introduction to the kalman filter," USA, Tech. Rep., 1995.

[26] J. Wang, N. Wu, X. Lu, W. X. Zhao, and K. Feng, "Deep trajectory recovery with fine-grained calibration using kalman filter," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 3, pp. 921–934, 2019.

[27] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[28] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[29] W.-C. Lee and J. Krumm, "Trajectory preprocessing," in *Computing with spatial trajectories*. Springer, 2011, pp. 3–33.

[30] M. Dror, A. Efrat, A. Lubiw, and J. S. Mitchell, "Touring a sequence of polygons," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, 2003, pp. 473–482.

[31] X. Tan and B. Jiang, "Efficient algorithms for touring a sequence of convex polygons and related problems," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2017, pp. 614–627.

[32] A. S. Klitgaard, L. E. Josefsen, M. V. Mikkelsen, and K. Torp, "A distributed spatial data warehouse for ais data," in *2024 25th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2024, pp. 211–218.

[33] D. Nguyen and R. Fablet, "Traisformer-a generative transformer for ais trajectory prediction," *arXiv e-prints*, pp. arXiv–2109, 2021.

[34] K. Patroumpas, E. Chondrodima, N. Pelekis, and Y. Theodoridis, "Trajectory detection and summarization over surveillance data streams," *Big Data Analytics for Time-Critical Mobility Forecasting: From Raw Data to Trajectory-Oriented Mobility Analytics in the Aviation and Maritime Domains*, pp. 85–120, 2020.

[35] A. Tritsarolis, Y. Kontoulis, N. Pelekis, and Y. Theodoridis, "Masec: discovering anchorages and co-movement patterns on streaming vessel trajectories," in *Proceedings of the 17th International Symposium on Spatial and Temporal Databases*, 2021, pp. 170–173.

[36] H. Alt and M. Godau, "Computing the fréchet distance between two polygonal curves," *International Journal of Computational Geometry & Applications*, vol. 5, no. 01n02, pp. 75–91, 1995.

[37] B.-K. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Proceedings 14th International Conference on Data Engineering*. IEEE, 1998, pp. 201–208.