Uncertainty-Aware Ship Location Estimation using Multiple Cameras in Coastal Areas

Song Wu*†, Alexandros Troupiotis-Kapeliaris‡, Dimitris Zissis‡, Kristian Torp†, 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*—Recent advances, especially in deep learning, allow to effectively detect ship targets in surveillance videos. However, the translation of these detections to the real-world locations of ships has not been sufficiently explored. The common approach in the literature is using a transformation matrix to convert a pixel to a real-world coordinate. However, this approach has three shortcomings: first, a set of reference point pairs has to be manually prepared to establish the matrix; second, the matrix always maps a pixel to the same real-world coordinate, ignoring that there is no one-to-one correspondence between discrete pixel coordinates and continuous real-world coordinates; third, this approach can only work with one camera. In light of this, we propose a technique *PixelToRegion* that explicitly takes into account the uncertainty in coordinate conversion by mapping each pixel to a spatial polygon. Next, we propose a new algorithm *MCbSLE* that can estimate ship locations using pixel sets from multiple cameras. The precision of location estimation by *MCbSLE* is enhanced through spatial intersection between polygons from different cameras. Experiments are conducted under 16 carefully designed multi-camera settings to evaluate *MCbSLE w.r.t.* four factors: different ports, the number of cameras, the distance between cameras, and camera headings. Results on one-day ship trajectory data show that (1) an 79.8% accuracy in the number of coordinates can be achieved by *MCbSLE* when there are no more than 10 ships in camera views; (2) using multiple cameras can improve the precision of location estimation by one order of magnitude compared with using one camera.**

***Index Terms*—Trajectory, Location, Camera, AIS, Ship**

I. Introduction

In the maritime domain, ship tracking is a fundamental task and leads to the collection of massive ship trajectory data over time. Such data enables many important applications [1], such as extracting traffic routes [2], detecting fishing activities [3], predicting future locations [4], and estimating CO_2 emissions [5]. In practice, ship tracking is achieved using various sensors, and two commonly-used are the Automatic Identification System (AIS) and surveillance cameras.

The AIS technology was initially introduced for the purpose of collision avoidance. Basically, ships with AIS onboard can proactively broadcast their navigational information to nearby ships, coastal AIS stations, and even satellites. AIS data can be divided into three types [6]: (1) static data such as the MMSI (a ship's identity), ship type, and ship size; (2) dynamic data such as the current ship location, speed, and heading; and (3) voyage-related data such as destination and estimated time of arrival. Usually, a ship sends AIS data every 2 to 10 seconds when it is moving and every 3 minutes when it is anchored [7]. In the past decades, AIS has been extensively deployed worldwide, making it the most used data source for tracking ships today [8]–[10]. However, some issues still exist for AIS and the main problem is intentional AIS switch-off. The collaborative nature of AIS means that the crew onboard can switch off AIS if they want to hide their whereabouts from others. Ships switching off AIS thereby become "dark" and cannot be tracked using AIS [8].

Cameras are also widely used in maritime surveillance for monitoring inbound, outbound, and passing traffic [11] [12], [13]. They can be installed onboard ships, in harbors, and along waterways. The benefit of cameras is that they can provide visual appearance of ships and monitor ships in a real-time manner. However, camera-based ship tracking also has limitations, among which the main problem is that ship identity and other useful information in AIS are hard to determine using images alone [12].

Given the pros and cons of a single sensor, multi-sensor data fusion has become popular over time. In the past years, many studies have emerged to fuse AIS and video data [12], [14], [15]. The main purpose of these studies is to recognize identities of the ship targets in videos and enrich these targets with various information from AIS [12], [16]. Since these studies deal with only one camera, data fusion can be performed either in the lon/lat or in the pixel coordinate space. Most of the previous studies do the latter and choose to convert lon/lat to pixel coordinates [12], [15], [17].

Contrary to the purpose of enhancing video data with AIS data in previous studies, this work considers the data fusion problem from the other perspective: improving AIS data quality with video data. Concretely, large temporal gaps are found in AIS data from time to time [18], ranging from minutes to hours or even days. For these large gaps, the common linear interpolation technique does not help much. For example, the interpolated trajectory may cross land (e.g. islands). Moreover, this technique implies that a ship is constantly moving all the time during the gap, which is not always the case. On the other hand, surveillance videos have been used for decades in coastal areas to monitor ship traffic in a real-time manner, and

2375-0324/24/$31.00 ©2024 IEEE
DOI 10.1109/MDM61037.2024.00034

large amounts of video data are collected over time. However, one pillar that is lacking in the literature is how to effectively extract the ship locations from these video data, which is thus the research focus in this work.

In summary, the research gap we are trying to fill in this work is estimating ship locations using multiple cameras, which has not been well studied yet. A multi-camera setting is chosen because we will explicitly take into account the uncertainty in coordinate conversion between pixels and lon/lat. Using multiple cameras can thus reduce such uncertainty and increase the precision of location estimation. Also, a multi-camera setting is common in reality especially when the area of interest is large and needs to be monitored in various directions [19], [20]. The estimated ship locations can be used to identify small or "dark" ships without AIS signals and warn others about their presence, thus enhancing navigational safety. Note that ship detection itself is an orthogonal and well-studied problem [16], [17] and beyond the scope of this work. It is thus assumed that ships in images have already been ideally detected. The main contributions in this work are as follows:

- We highlight the uncertainty in converting discrete pixel coordinates to continuous lon/lat coordinates. A technique $PixelToRegion$ is then proposed that models such uncertainty by mapping a pixel coordinate to a spatial polygon.
- We propose an algorithm $MCbSLE$ that estimates ship locations using pixel sets from multiple cameras. This algorithm improves the precision of location estimation through spatial intersection between pixel polygons from different cameras.
- To evaluate how the estimation results by $MCbSLE$ are affected by different ports, the number of cameras, the distance between cameras, and camera headings, extensive experiments are conducted using one-day AIS data under 16 carefully designed multi-camera settings.
- Results show that the accuracy of $MCbSLE$ in the number of coordinates is mostly affected by the number of coordinates in the ground truth, and an accuracy of 79.8% can be achieved when there are no more than 10 ships in camera views. In addition, using multiple cameras can improve the precision of location estimation by one order of magnitude compared with using only one camera.

The rest of the paper is organized as follows: Section II gives a brief review of related work. Section III defines some notations and provides the problem definition. Section IV presents our proposed methodology. Section V shows the main experimental results for evaluation of the proposed solutions. Finally, Section VI concludes the paper and discusses directions for future research.

II. Related work

Video-based ship detection and tracking. As a traditional and widely-used surveillance means, cameras allow maritime administrations to monitor ship traffic in a real-time manner. To aid human decision-making, two tasks are usually performed on surveillance videos: ship detection and ship tracking. Ship detection is the task of detecting ship targets appearing in images. Detection results are usually returned as a list of bounding boxes, where each bounding box is considered as a ship target. Based on ship detection results, ship tracking aims to build association between bounding boxes from adjacent images such that bounding boxes from the same ship are assigned the same ID. For ship detection, the best performance is currently obtained using deep-learning-based object detection models, such as the YOLO (You Only Look Once) family and their variants [21]–[24]. For ship tracking, representative studies include DeepSORT-based ones [12], [16], [25] and overlapping-ratio-based ones [11], [20]. Therefore, this work acknowledges the previous studies and does not aim at proposing new approaches for ship detection and tracking. Instead, we focus on another problem that has received less attention and not been explored much: estimating real-world locations of the detected ship targets.

Ship location estimation using video data. In the literature, only a few relevant studies exist [16], [26]–[28]. The study in [26] mainly uses the ship length to estimate the distance and relative azimuth between a ship and a camera. In [26], the width of a bounding box is assumed to be the projected length of a ship. Therefore, the accuracy of distance estimation by [26] decreases when the relative angle between the ship course and the camera bearing is small or the predicted bounding box is inaccurate. Experiments in [26] are conducted near the English Channel, and results show that the mean absolute error of distance estimation is 0.6975 km for ships within 5 km from the camera, 0.9767 km for ships between 5 km and 10 km from the camera, and 0.8514 km for ships between 10 km and 15 km from the camera. Such large deviations thus limit its applications in real-world scenarios. The studies in [16], [27] employs a transformation matrix to convert a pixel coordinate to a lon/lat coordinate. The main drawback of this matrix approach is that a set of point pairs has to be manually prepared to establish the matrix. Moreover, this approach ignores the fact that there is no one-to-one correspondence between discrete pixel coordinates and continuous lon/lat coordinates. Therefore, this matrix approach only works well when the monitored region is small enough such as a circle with a radius of hundreds of meters centered at the camera location. The study in [28] also employs the matrix approach, but the difference is that it estimates the ship location using multiple cameras rather than one camera as in [16], [27]. Basically, the method in [28] locates a ship by using the least-squares technique to solve a linear system formed by multiple matrices. However, the method in [28] is not ready to be used in reality: the homologous points in different cameras are manually selected in the experiments in [28]; it is therefore not viable to run the method automatically in real-world applications.

III. Preliminaries

A. Notation

A camera monitoring the area of interest is denoted by cam, and it has the following attributes:

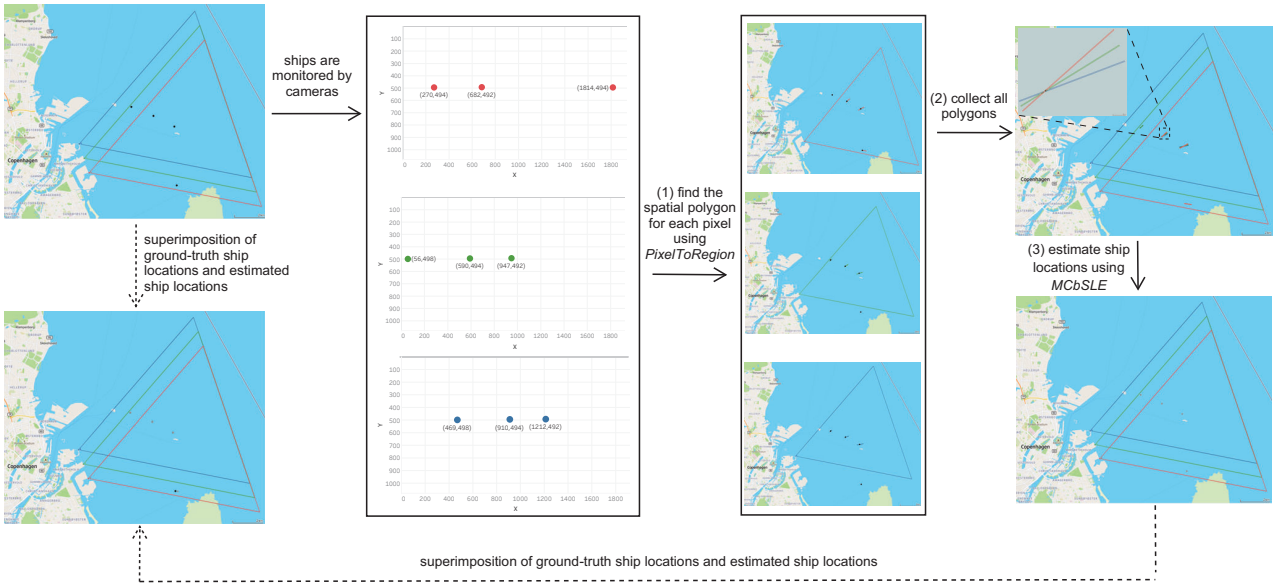
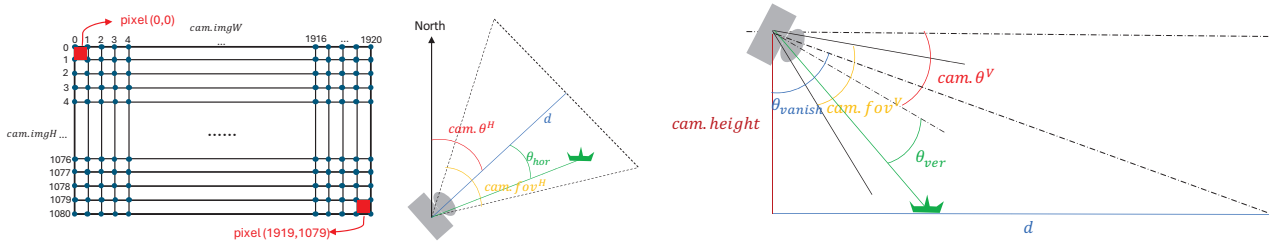


Fig. 1. An overview of the proposed solutions. Three "virtual" cameras are placed near Copenhagen, and the monitored areas are shown in red, green, and blue respectively. Black dots are locations of the 4 ships in the camera views. By processing video data in each camera, the pixel locations of these ships are shown next. Then the proposed solutions come into play: (1) The proposed algorithm $PixelToRegion$ retrieves the spatial polygon for each pixel location; (2) All spatial polygons are collected as input; (3) The proposed algorithm $MCbSLE$ estimates the ship locations using the polygons. Finally, the real ship locations and the estimated ship locations are shown together.



(a) Each pixel is treated as a square (b) Horizontal orientation (c) Vertical orientation

Fig. 2. Illustration of camera notations

1) $cam.lon$ and $cam.lat$ are the longitude and latitude of the location of the camera respectively.
2) $cam.height$ is the height of the camera $w.r.t.$ the water surface.
3) $cam.\theta^H$ is the horizontal orientation of the camera. It is expressed as the clockwise angle $w.r.t.$ the due north direction (0°).
4) $cam.\theta^V$ is the vertical orientation of the camera. It is negative for cameras pointing downwards and positive for cameras pointing upwards.
5) $cam.fov^H$ and $cam.fov^V$ are the horizontal and vertical field of view (in degrees) of the camera respectively.
6) $cam.imgW$ and $cam.imgH$ are the width and height (in pixels) of videos taken by the camera.

Fig. 2b and Fig. 2c illustrate some of the above notations.

B. Problem definition

The problem studied in work is referred to as ***Ship Location Estimation using Multiple Cameras***: Given a multi-camera setting $\{cam_i\}$, where $1 \leq i \leq n$, and the monitored region by $\{cam_i\}$ is denoted as R. The ships located inside R are denoted as S, and their locations (in lon/lat) are denoted as C. Suppose the pixels corresponding to C in the n cameras are $\{P_i\}$, where each P_i is a set of pixels in cam_i and a pixel is indexed by a pair of integers. The goal is then to estimate and return a set of lon/lat coordinates C' using $\{P_i\}$ such that C' is as close to C as possible.

Note that in this work we assume there exists an approach $\mathbb{A}$ that can process video data and select the correct pixel for each detected ship target. The selected pixels in each camera are thus fed as input in this work. The approach $\mathbb{A}$ itself is

beyond the scope of this work and not our research focus.

IV. Methodology

The proposed methodology in this work is inspired by the observation that there is no one-to-one correspondence between discrete pixel coordinates and continuous lon/lat coordinates, which is ignored in previous studies. For example, given a pixel location p, methods based on transformation matrix always map p to the same lon/lat coordinate. Overall, our methodology is composed of two parts. The first part explicitly models the uncertainty in coordinate conversion between pixels and lon/lat by mapping each pixel to its corresponding spatial region. The second part benefits from a multi-camera setting and can significantly improve the precision of location estimation through the operation of spatial intersection.

A. Mapping pixels to spatial regions

Conceptually, a camera is a function that maps a spatial region in real world to the pixel space in images. Given that there is a finite number of pixels and the spatial region is continuous, there is no one-to-one correspondence between a pixel and a lon/lat coordinate. This observation implies that a pixel p actually corresponds to a certain region R, and all lon/lat coordinates in R are mapped to the same pixel p.

In this work, we propose a new algorithm $PixelToRegion$ that maps a pixel to a spatial region. $PixelToRegion$ extends the pinhole imaging model [25] by treating each pixel as a square, and Algorithm 1 shows the pseudo-code. The input d is the maximum monitoring distance of a camera along the horizontal shooting direction. It is mainly affected by the elevation and the field of view of cameras. Thus, objects farther away than d are assumed to become too small to be detected and vanish from the field of view. In real scenarios, the value of d can be taken based on various factors, such as the width of the river facing cameras, the size of a harbor, etc.

Algorithm 1 has mainly two steps. The first step (line#1 to line#19) transforms each boundary point (blue dots in Fig. 2a) of pixels to a lon/lat coordinate based on the pinhole principle. Such transformation is not performed if the distance exceeds d. The blue dots in Fig. 2a are indexed as (x, y), with x ranging from 0 to $cam.imgW$ and y ranging from 0 to $cam.imgH$. The $minY$ variable records the minimum y of blue dots that satisfy the d threshold. For boundaries points (with y being $minY - 1$) that just exceed the d threshold, d is used in their transformation (line#18 to line#19). Also note that pixels have different index ranges, with x from 0 to $cam.imgW - 1$ and y from 0 to $cam.imgH - 1$. The function $forward$ in line#16 computes the coordinate that has a distance A to the camera location and a relative angle θ_{hor} to the camera heading. Then the second step (line#20 to line#28) uses the lon/lat coordinates of the four boundary points of a pixel p to create a polygon, and this polygon is assigned to p. Finally, a dictionary D that maps a pixel to its assigned polygon is returned as output.

Given that each pixel corresponds to a spatial region, uncertainty exists when trying to transform a pixel to a lon/lat coordinate. Therefore, the novelty in this work is using multiple cameras for reducing such uncertainty.

For a lon/lat coordinate loc, it is easy to compute the pixel p whose associated spatial region contains loc. But given p, it is difficult to compute backwards and obtain loc due to uncertainty. The idea in this work is inspired by a multi-camera setting. Considering a set of cameras $cam_1, cam_2, ..., cam_n$, let us assume that loc is inside the field of view of each cam_i. For each cam_i, the pixel p_i can be computed whose spatial region contains loc. Therefore, the uncertainty can probably be substantially reduced if we take the intersection of spatial regions for $p_1, p_2, ...,$ and p_n. Based on this idea, next we present the algorithm that recovers real-world coordinates from multi-camera pixel sets.

Algorithm 1: PixelToRegion

Input: cam: a camera configuration
d: maximum monitoring distance
Output: a dictionary D that maps each pixel to a spatial region

1 $D \leftarrow$ dict();
2 $D_{aux} \leftarrow$ dict();
3 $\theta_{vanish} \leftarrow \arctan(\frac{d}{cam.height})$;
4 $minY \leftarrow -1$;
5 **for** $x \leftarrow 0$ **to** $cam.imgW$ **by** 1 **do**
6 **for** $y \leftarrow 0$ **to** $cam.imgH$ **by** 1 **do**
7 $\theta_{hor} \leftarrow \arctan(\frac{x - \frac{cam.imgW}{2}}{\frac{cam.imgW}{2}} * \tan(\frac{cam.fov^H}{2}))$;
8 $\theta_{ver} \leftarrow \arctan(\frac{y - \frac{cam.imgH}{2}}{\frac{cam.imgH}{2}} * \tan(\frac{cam.fov^V}{2}))$;
9 **if** $90 + cam.\theta^V - \theta_{ver} \geq \theta_{vanish}$ **then**
10 **continue**;
11 **if** $minY == -1$ **then**
12 $minY \leftarrow y$;
13 $\theta'_{ver} \leftarrow 90 + cam.\theta^V - \theta_{ver}$;
14 $C \leftarrow cam.height * \tan(\theta'_{ver})$;
15 $A \leftarrow \frac{C}{\cos(\theta_{hor})}$;
16 $lon, lat \leftarrow forward(cam, cam.\theta^H + \theta_{hor}, A)$;
17 $D_{aux}[x, y] \leftarrow lon, lat$;
18 **for** $x \leftarrow 0$ **to** $cam.imgW$ **by** 1 **do**
19 $D_{aux}[x, minY - 1] \leftarrow d$;
20 **for** $x \leftarrow 0$ **to** $cam.imgW - 1$ **by** 1 **do**
21 **for** $y \leftarrow 0$ **to** $cam.imgH - 1$ **by** 1 **do**
22 **if** $x, y \in D_{aux}$ **then**
23 $p_1 \leftarrow D_{aux}[x, y]$;
24 $p_2 \leftarrow D_{aux}[x + 1, y]$;
25 $p_3 \leftarrow D_{aux}[x + 1, y + 1]$;
26 $p_4 \leftarrow D_{aux}[x, y + 1]$;
27 $D[x, y] \leftarrow polygon(p_1, p_2, p_3, p_4)$;
28 **return** D;

Algorithm 2: MCbSLE

Input: $\{cam_i\}$: a list of cameras, where $1 \leq i \leq n$
$\{P_i\}$: a list of P_i, where $1 \leq i \leq n$, and P_i is a set of pixels in cam_i
d: maximum monitoring distance
Output: C: a set of lon/lat coordinates

1 $C \leftarrow \{\}$ // initialize an empty result;
2 **foreach** cam_i **do**
3 $cam_i.D \leftarrow PixelToRegion(cam_i, d)$;
4 $cam_i.R \leftarrow \bigcup cam_i.D.values()$
5 $S \leftarrow \{\}$ //a set of valid pixel-combination candidates;
6 **for** $k \leftarrow n$ **to** 1 **by** -1 **do**
7 $P^k \leftarrow$ all k-combinations of $\{P_i\}$;
8 $prod \leftarrow \{\}$;
9 **foreach** $comb \in P^k$ **do**
10 $prod' \leftarrow$ the cartesian product of sets in $comb$;
11 append all elements in $prod'$ to $prod$;
12 $S^k \leftarrow$ dict();
13 **foreach** $e \in prod$ **do**
14 $regions \leftarrow \{\}$;
15 **foreach** $pixel \in e$ **do**
16 $r \leftarrow$ the associated polygon of $pixel$;
17 add r to $regions$;
18 $r' \leftarrow$ the spatial intersection of $regions$;
19 **if** r' *is not empty* **then**
20 $S^k[e] \leftarrow r'$;
21 **foreach** $e \in S^k$ **do**
22 **if** $\nexists e' \in S : e \subset e'$ **then**
23 $r \leftarrow S^k[e]$;
24 $rest \leftarrow$ cameras not appearing in e;
25 **foreach** $cam_j \in rest$ **do**
26 $r \leftarrow diff(r, cam_j.R)$;
27 **if** r *is not empty* **then**
28 append e to S;
29 **if** r *is a MultiPolygon* **then**
30 **foreach** $poly \in r$ **do**
31 add the centroid of $poly$ to C;
32 **else**
33 add the centroid of r to C;
34 **return** C;

B. Multi-camera-based ship location estimation

Algorithm 2 shows the pseudo-code of $MCbSLE$ (Multi-Camera-based Ship Location Estimation) that estimates lon/lat coordinates of ships using multiple cameras. The algorithm takes as input a set of pixel sets $\{P_i\}$, where each P_i belongs to a camera cam_i.

First, all cameras in $\{cam_i\}$ are initialized, and the visible region of each cam_i is computed by taking the union of all spatial regions in $cam_i.D$ (line#1 to line#4). Then all pixel combinations are examined in the decreasing order of cardinality (line#6). Thus, spatial intersection involving more pixels is checked before that involving less pixels, and the purpose is to avoid repeated processing. For example, if we already know that a ship is located in the intersection of spatial regions for pixels p_1, p_2, p_3, we then do not need to check again the intersection by any two pixels in p_1, p_2, p_3. If the spatial intersection r' is not empty, it will be added to S^k (line#13 to line#20) for further validity check.

The validity check happens from line#21 to line#33. It starts by checking if we can skip the pixel combination based on the reason mentioned above. Therefore, a combination e in S^k will not be processed further if e is a subset of any valid combination e' in S. If e cannot be skipped, the location estimation continues as follows. The set of cameras $rest$ (possibly empty) are first retrieved that do not appear in e. Since it is implied that the lon/lat coordinate underlying e is invisible in the cameras $rest$, the spatial intersection r from e is then refined by taking the difference between r and the monitored areas of cameras in $rest$. Afterwards, if the refined r is a MultiPolygon $mPoly$, the centroid points of child polygons in $mPoly$ are added to the final result; otherwise, the centroid point of the refined r is added to the final result. Lastly, the final result C is returned as the set of estimated lon/lat coordinates for ships. An overview of the proposed methodology in shown in Fig. 1.

There are two cases in which Algorithm 2 may not return a correct number of coordinates. The first case happens when there are two ships s_1, s_2 in camera views and the pixels for s_1 is a subset of the pixels for s_2 or the other way around. As a result, either s_1 or s_2 will be missed due to the technique to avoid repeated processing in Algorithm 2. However, experiments in Section V will show that this case happens rarely. The second case happens when the polygons for pixels from different cameras intersect with each other but the pixel combination does not correspond to a ship in camera views. These undesired intersections thus make Algorithm 2 return more coordinates as results.

V. Empirical studies

This section will conduct experiments to investigate how the location estimation results by $MCbSLE$ are affected by various factors (different ports, number of cameras, distance between cameras, and camera headings) and thereby answer mainly two questions:

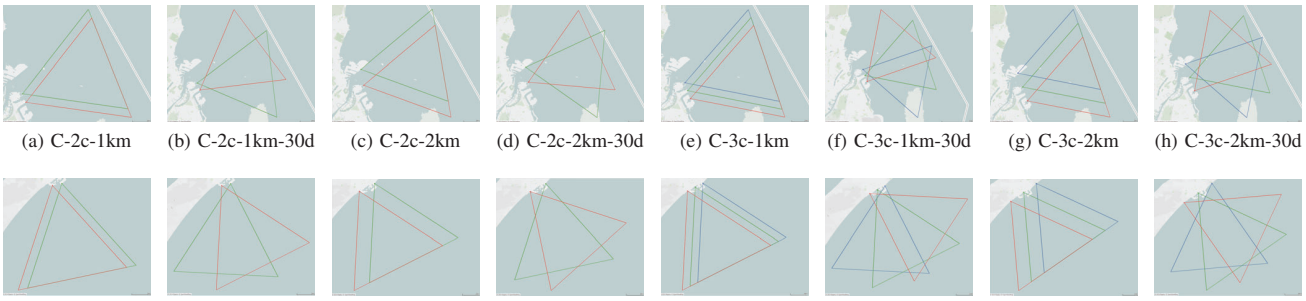
1) How spatially close are our estimated locations *w.r.t.* the ship locations in the ground truth?
2) Can our method return the correct number of ship locations as in the ground truth?

At a timestamp t, the ground truth C_t is the set of lon/lat coordinates of the ships in the camera views.

It is worth noting that no previous study is suitable as baseline for comparison due to the following reasons: (1) the methods based on matrix transformation (e.g. [16], [27]) can only work with one camera; (2) the only previous study that uses multiple cameras for ship location estimation relies on manually-chosen corresponding points across different cameras [28], so it is not clear how the approach can be run automatically in real scenarios.

TABLE I. Parameters in different multi-camera settings

name	$cam1.\theta^H$	$cam2.\theta^H$	$cam3.\theta^H$	($cam1.lon$, $cam1.lat$)	($cam2.lon$, $cam2.lat$)	($cam3.lon$, $cam3.lat$)
Copenhagen-2c-1km	68°	68°	-	(12.635886, 55.687784)	(12.629929, 55.696112)	-
Copenhagen-2c-1km-30d	53°	83°				
Copenhagen-2c-2km	80°	80°	-	(12.629453, 55.697396)	(12.623927, 55.715087)	-
Copenhagen-2c-2km-30d	65°	95°				
Copenhagen-3c-1km	71°	71°	71°	(12.639732, 55.680441)	(12.634555, 55.688933)	(12.629377, 55.697425)
Copenhagen-3c-1km-30d	41°	71°	101°			
Copenhagen-3c-2km	71°	71°	71°	(12.639134, 55.680495)	(12.628779, 55.697479)	(12.618414, 55.714463)
Copenhagen-3c-2km-30d	41°	71°	101°			
Skagen-2c-1km	168°	168°	-	(10.585117, 57.713291)	(10.601528, 57.715157)	-
Skagen-2c-1km-30d	153°	183°				
Skagen-2c-2km	153°	153°	-	(10.575398, 57.714702)	(10.605303, 57.722851)	-
Skagen-2c-2km-30d	138°	168°				
Skagen-3c-1km	153°	153°	153°	(10.575320, 57.714657)	(10.590271, 57.718732)	(10.605225, 57.722806)
Skagen-3c-1km-30d	123°	153°	183°			
Skagen-3c-2km	145°	145°	145°	(10.562887, 57.709338)	(10.590378, 57.719635)	(10.617884, 57.729926)
Skagen-3c-2km-30d	115°	145°	175°			



(a) C-2c-1km (b) C-2c-1km-30d (c) C-2c-2km (d) C-2c-2km-30d (e) C-3c-1km (f) C-3c-1km-30d (g) C-3c-2km (h) C-3c-2km-30d

(i) S-2c-1km (j) S-2c-1km-30d (k) S-2c-2km (l) S-2c-2km-30d (m) S-3c-1km (n) S-3c-1km-30d (o) S-3c-2km (p) S-3c-2km-30d

Fig. 3. The monitored area of each multi-camera setting: "C" ("S") is short for Copenhagen (Skagen); three cameras are colored differently.

A. Setup

Datasets. The input of our algorithm is a set of pixel sets, and in real scenarios this input should come from ship detection results on video data. However, to the best of our knowledge, there is no public multi-camera video dataset that monitors ship traffic at the same time and from different angles in the literature. Therefore, this work uses "virtual cameras" and simulates the images using AIS data as follows.

Some pre-processing steps are first applied to AIS data: records with invalid coordinates are removed, then for the same ship, records with duplicate timestamps are also removed. Next, suppose the monitored region under a multi-camera setting is R, the algorithm input can thus be generated through position interpolation for each ship in AIS data. Given a ship s, interpolation is performed for each timestamp t between the start timestamp t^s_{start} and the end timestamp t^s_{end} of s. If the interpolated position loc^s_t is located inside R, loc^s_t is added to the set of ground truth coordinates for t. After processing all ships in AIS data, the set of coordinates C_t for a timestamp t is used to generate the set of pixel sets $\{P_i\}$. Basically, for each lon/lat coordinate c in C_t, we find the pixel whose associated spatial polygon contains c for each camera, and the pixel is then added to $\{P_i\}$. The experiments thereby aim to evaluate how well the proposed approach can reconstruct C_t from $\{P_i\}$.

Without loss of generality, the up-to-date and free Danish AIS data[1] is used in experiments, which is also widely used in previous studies (e.g. [3], [29], [30]). Specifically, we have chosen randomly the one-day AIS data on Jan 6, 2024 for the experiments, and similar results are expected on other days.

Multi-camera settings. Generally, multiple cameras can be installed around a region in many different ways. For example, cameras can have different headings and elevations; distances between cameras can be different; and cameras may be arranged in a particular shape (e.g. a circle). Therefore, the actual installation of multiple cameras really depends on the situation on the spot. In this work, we consider some of the most important aspects of a multi-camera setting:

1) different ports. Generally, the traffic characteristics differ from port to port. In this work, two ports are selected for experiments, namely Copenhagen and Skagen. Copenhagen is chosen because nearby ships move in various directions. Copenhagen sees both ships entering and leaving itself and ships just passing between the North Sea and the Baltic Sea. Unlike Copenhagen, Skagen has a large anchorage area just outside the port, and many ships stopped in this area, as depicted by the vessel monitoring service MarineTraffic[2]. As a result, ships near Skagen do not move as much as ships near Copenhagen. These two

[1]https://web.ais.dk/aisdata/
[2]https://www.marinetraffic.com/

ports thus represent different movement characteristics for the experiments.

2) the number of cameras. In real scenarios, the number of cameras needed depends on the demand at hand. In this work, we will compare two choices: two cameras and three cameras.
3) the distance between cameras. Cameras should be placed at some distances from each other to achieve a better coverage of the region of interest. In this work, we will compare two choices: the distance between two adjacent cameras is either 1 km or 2 km.
4) camera headings. Depending on the actual situation, cameras may or may not have the same headings. In this work, we will compare both cases.

As a result, we designed a total of 16 different multi-camera settings for experiments. Table I gives a summary of these settings, and Fig. 3 depicts the monitored area of each setting. The other irrelevant camera parameters are fixed as follows: $cam.\theta^V$ = -2°, $cam.height$ = 40 meters, $cam.fov^H$ = 60°, $cam.fov^V$ = 35.98°, $cam.imgW$ = 1920, $cam.imgH$ = 1080, and the max. monitoring distance is set as 10 km.

Implementation. The proposed algorithm is implemented in Python. For geospatial functionalities, three python libraries are used: *pyproj*, *shapely*, and *geopandas*. For reproducibility, the source code is made available in Github[3].

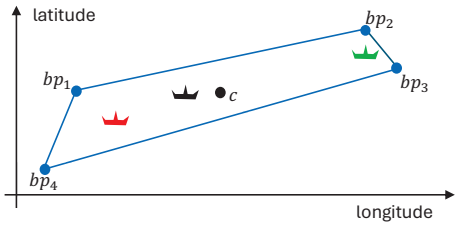


Fig. 4. Wherever a ship is located inside the blue polygon, the centroid point c is returned as the estimated ship location. This estimate is more accurate for the black ship than for the green and red ships. The upper bound of the distance between c and the real ship location is thus the maximum distance between c and the boundary points bp_1, bp_2, bp_3, and bp_4.

TABLE II. The max. upper bound of distance error (in meters) in different visibility zones, where $zone_1$ is visible in only one camera; $zone_2$ is visible in two cameras; and $zone_3$ is visible in three cameras.

name	$zone_1$	$zone_2$	$zone_3$
Skagen-2c-1km	919.6	80.06	-
Skagen-2c-1km-30d	919.6	70.88	-
Skagen-2c-2km	919.6	39.62	-
Skagen-2c-2km-30d	919.7	35.66	-
Skagen-3c-1km	919.6	80.38	39.61
Skagen-3c-1km-30d	919.7	73.11	31.41
Skagen-3c-2km	919.7	39.64	19.32
Skagen-3c-2km-30d	919.7	35.13	16.96

B. Analysis on the distance between the estimated coordinates and the ground truth

For distance analysis, we resort to compute the upper bound of distance deviations for the returned coordinates as follows. If there is only one ship s in the camera views, a polygon or a multi-polygon (which happens rarely) will be generated by the spatial operations on the pixel polygons for s in Algorithm 2. For one polygon $sPoly$, our approach will return its centroid point; and for a multi-polygon $mPoly$, our approach will return the centroid point of each polygon $poly$ contained in $mPoly$. The real location of s is then expected to be the centroid point or one of the centroid points. Actually, for any location in $sPoly$ ($mPoly$), the proposed approach will return the same centroid(s) as estimates. In other words, wherever the ship s is located inside $sPoly$ ($mPoly$), the approach will return the same location estimation. Therefore, the upper bound of distance error for a centroid c (also referred to as the precision of location estimation in this work) can be computed as follows based on the polygon $poly$ where c comes from:

$$upper\_bound(\,c\,) \;=\; \max(\,distance(\,c,\,bp\,)\,)$$

where bp is a boundary point of $poly$ and Fig. 4 gives an example for illustration. The spatial distribution of the upper bound of distance error is then approximated as follows: the monitored area by each setting is split into grids of size 100 meters by 100 meters, and the grid edges are parallel or perpendicular to the average heading of cameras. The upper bound of distance error is then computed for all grid corners. Fig. 5 illustrates results for the settings near Skagen, and the results for Copenhagen are omitted because they are nearly the same. Base on visibility, the monitored area under each setting is divided into two or three zones: $zone_1$ is visible in only one camera; $zone_2$ is visible in two cameras; and $zone_3$ is visible in three cameras.

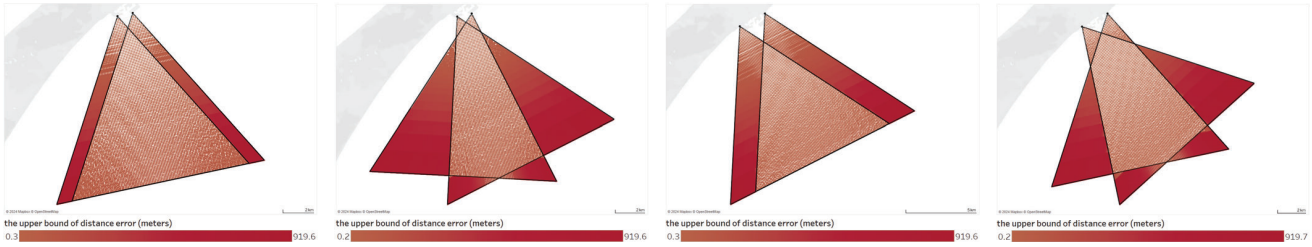
Effect of the number of cameras. Taking "Skagen-3c-1km" for example, the maximum upper bound of distance error decreases significantly from 919.6 meters in $zone_1$ to 80.38 meters in $zone_2$ and 39.61 meters in $zone_3$. It means that using more cameras can effectively reduce the uncertainty for ship location estimation.

Effect of the distance between cameras. Taking "Skagen-2c-1km" and "Skagen-2c-2km" for example, the maximum upper bound of distance error in $zone_2$ decreases from 80.06 meters to 39.62 meters. This can possibly be explained as follows: increasing the distance between cameras makes angle differences larger between polygons from different cameras, so the result of spatial intersection is a more regular polygon rather than a needle-like polygon. The upper bound of distance error is thus decreased.

Effect of the camera headings. Taking "Skagen-3c-2km" and "Skagen-3c-2km-30d" for example, the maximum upper bound of distance error decreases slightly from 39.64 meters to 35.13 meters in $zone_2$ and from 19.32 meters to 16.96 meters in $zone_3$. Although not as salient as in the previous paragraph, this can probably be explained by the same reason regarding the angle difference between intersecting polygons.

To sum up, Fig. 5 shows that using multiple cameras can significantly reduce the uncertainty in ship location estimation

[3]https://github.com/songwu0001/MCbSLE



(a) Skagen-2c-1km (b) Skagen-2c-1km-30d (c) Skagen-2c-2km (d) Skagen-2c-2km-30d

(e) Skagen-3c-1km (f) Skagen-3c-1km-30d (g) Skagen-3c-2km (h) Skagen-3c-2km-30d

Fig. 5. The spatial distribution of the upper bound of distance error under multi-camera settings near Skagen.

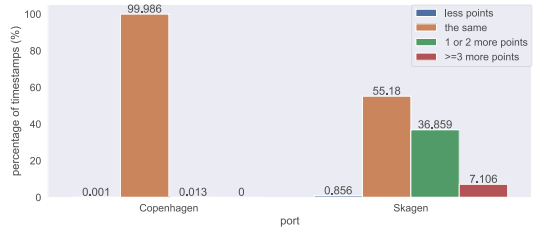


Fig. 6. The percentage of timestamps between the four classes aggregated separately for each port

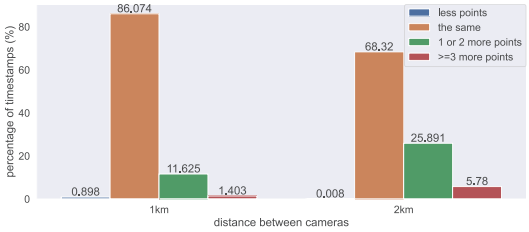


Fig. 8. The percentage of timestamps between the four classes aggregated separately by the distance between cameras

and gives accurate location estimates. Also note that the closer a point is to the cameras, the smaller uncertainty is expected for location estimation. Therefore, a multi-camera setting should be properly designed such that $zone_1$ mostly corresponds to pixels with larger y values. Besides that, visualizations as in Fig. 5 can serve as a useful tool for maritime authorities to inspect the expected precision of location estimation under any multi-camera setting. In this way, maritime authorities can perform simulation-first analysis before deploying a new setup of cameras for tracking ships.

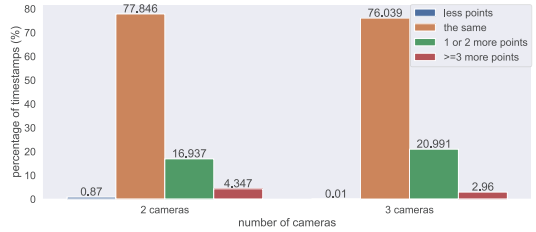


Fig. 7. The percentage of timestamps between the four classes aggregated separately using the number of cameras

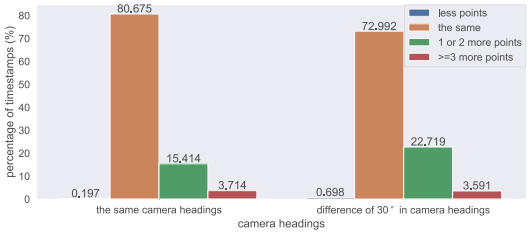


Fig. 9. The percentage of timestamps between the four classes aggregated separately based on camera headings

C. Analysis on the number of returned coordinates

Besides the precision of location estimation, it is also desired that *MCbSLE* can return a correct number of coordinates as in the ground truth.

Table III shows quartiles for the number of visible ships over time, which is aggregated separately for settings near Copenhagen and settings near Skagen. For Copenhagen, the number of visible ships ranges from 1 to 9, whereas for Skagen it ranges from 1 to 25. Therefore, more ships seem to appear simultaneously in Skagen than in Copenhagen. This can be

TABLE III. Statistics on the number of visible ships over time

	min.	1^{st} quartile	median	3^{rd} quartile	max.
Copenhagen	1	2	3	4	9
Skagen	1	12	15	19	25

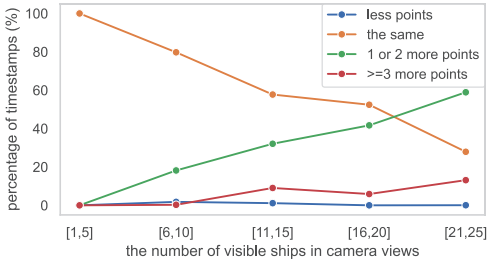


Fig. 10. The percentage of timestamps between the four classes aggregated separately for each group of the number of visible ships

explained by the fact that there is no large anchorage area near Copenhagen as near Skagen, so ships passing Copenhagen only appear in camera views for a shorter time. Therefore, both ports together represent the diversity of algorithm input.

Under a multi-camera setting S and at a timestamp t, the number of visible ships in S is num^{truth}, and the number of returned coordinates (ships) is num^{pred}. The experiment results are thus a list of quadruples $(S, t, num^{truth}, num^{pred})$. It is desired that num^{pred} should be as close to num^{truth} as possible. Next, we will investigate how the deviation $num^{pred} - num^{truth}$ is affected by the four dimensions in the 16 multi-camera settings. To facilitate analysis, the value of $num^{pred} - num^{truth}$ is divided into four classes:

- **"less points"** means some coordinates are missed by our approach, i.e. $num^{pred} < num^{truth}$.
- **"the same"** means our approach returns the same number of coordinates as in the ground truth, i.e. $num^{pred} = num^{truth}$. Therefore, the percentage of this class in the results can be considered as the accuracy of $MCbSLE$ for the number of coordinates.
- **"1 or 2 more points"** means our approach returns slightly more points than in the ground truth, i.e. $num^{pred} - num^{truth}$ = 1 or 2. This case can still be considered acceptable.
- **">= 3 more points"** means our approach returns significantly more point than in the ground truth, i.e. $num^{pred} - num^{truth} \geq$ = 3. This case should be avoided as much as possible.

Effect of ports. The 16 settings are divided into two groups based on which port is involved. Fig. 6 shows the results for each group. It is clear that for the 1^{st} group, our approach almost always returns the correct number of points (99.98% accuracy). In contrast, the accuracy for the 2^{nd} group is only 55.18%. Such a large difference shows that the accuracy is mostly affected by the number of visible ships in camera views. However, only 1 or 2 additional points are returned around 37% of the time, which is not that bad. For both groups, the percentage for the class *"less points"* is pretty low (below 1%), meaning that the case of missing points happens rarely in practice and is thus not a main issue.

Effect of the number of cameras. The 16 settings are divided into two groups based on the number of cameras used. Fig. 7 shows the results for each group. Overall, the distributions for each group are similar. This can be explained as follows: on one hand, the addition of a 3^{rd} camera can reduce the number of undesired intersections caused by two pixels; on the other hand, a 3^{rd} camera also leads to more polygons involved in computation, increasing the probability of (undesired) intersections. As a result, these two factors cancel each other, and adding more cameras thus does not necessarily increase the accuracy in the number of coordinates.

Effect of the distance between cameras. The 16 settings are divided into two groups based on the distance between cameras. Fig. 8 shows the results for each group. The accuracy for the 1^{st} group is about 18% higher than the second group (86% v.s. 68%). The probable reason is that when increasing the distance between cameras, the angle differences between polygons from different cameras tend to be larger, thus increasing the probability of undesired intersections.

Effect of the camera headings. The 16 settings are divided into two groups based on whether all cameras have the same heading. Fig. 9 shows the results for each group. The accuracy for the 1^{st} group is about 8% higher than the second group (81% v.s. 73%). For the same reason, this may be caused by the larger angle differences between polygons from different cameras.

Effect of the number of visible ships. The analysis in *"Effect of ports"* shows that the accuracy is largely affected by the number of ships in the ground truth. For further elaboration, Fig. 10 depicts the distributions between the four classes against the number of ships in the ground truth. In Fig. 10, the different number of visible ships are divided into five groups: [1,5], [6,10], [11,15], [16,20], and [21, 25]. Obviously, the accuracy gradually decreases from 99.9% in the 1^{st} group to 28% in the 5^{th} group, whereas the percentage for the class *"1 or 2 more points"* increases gradually from 0.008% in the 1^{st} group to 58.9% in the 5^{th} group. Another important observation is that the percentage for the class *"1 or 2 more points"* is significantly higher than the class *">= 3 more points"* for all the five groups. This means that if an incorrect number of coordinates are returned, our approach will only return 1 or 2 more coordinates most of the time.

To sum up, the above analysis reveals the following insights: (1) it happens rarely that our approach returns a less number of coordinates than in the ground truth; (2) the accuracy of $MCbSLE$ is mostly affected by the number of visible ships in camera view(s); and (3) the accuracy can grow by properly decreasing the distance between cameras and angle differences between camera headings.

VI. Conclusion

In this work, we propose $MCbSLE$, an algorithm that can estimate real-world ship locations using video data from multiple cameras. $MCbSLE$ considers the uncertainty in coordinate

conversion by mapping a pixel coordinate to a spatial polygon. Therefore, the precision of location estimation by $MCbSLE$ can be significantly improved through intersection of multiple polygons. Experiments are conducted using a one-day AIS trajectory dataset under 16 carefully designed multi-camera settings. Results show that using multiple cameras enhances the precision of location estimation by one order of magnitude compared with using one camera. The estimated ship locations can be used in many ways, such as identifying small or "dark" ships without AIS signals, fusing with other data sources for more robust ship tracking, etc. Since $MCbSLE$ does not directly deal with video data, $MCbSLE$ can be run whenever the input pixel sets are available, regardless of how they are generated from video data. It is envisioned that $MCbSLE$ can be implemented in major harbors and near crucial shipping channels, such as Shanghai, Singapore, the Strait of Gibraltar, and the English Channel.

For future work, several directions can be explored. For example, a method would be desired that can automatically design the optimal multi-camera setting under a specific surveillance scenario. Additionally, this work assumes the ship targets in video data have already been perfectly detected, so it is also interesting to investigate how this work can be extended to work with less accurate detection results on video data.

ACKNOWLEDGMENT

This publication has been developed under the framework of the "Data Engineering for Data Science" (DEDS) project that has received funding from the European Union's Horizon 2020 programme (call identified: H2020-MSCA-ITN-EJD-2020) under grant agreement No 955895.

REFERENCES

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

[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] 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.

[4] E. Chondrodima, N. Pelekis, A. Pikrakis, and Y. Theodoridis, "An efficient lstm neural network-based framework for vessel location forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 4872–4888, 2023.

[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] Y. Zhou, W. Daamen, T. Vellinga, and S. P. Hoogendoorn, "Ship classification based on ship behavior clustering from ais data," *Ocean Engineering*, vol. 175, pp. 176–187, 2019.

[7] Z. Wei, X. Xie, and X. Zhang, "AIS trajectory simplification algorithm considering ship behaviours," *Ocean Engineering*, vol. 216, p. 108086, 2020.

[8] I. Kontopoulos, K. Chatzikokolakis, D. Zissis, K. Tserpes, and G. Spiliopoulos, "Real-time maritime anomaly detection: detecting intentional ais switch-off," *International Journal of Big Data Intelligence*, vol. 7, no. 2, pp. 85–96, 2020.

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

[10] I. Kontopoulos, G. Spiliopoulos, D. Zissis, K. Chatzikokolakis, and A. Artikis, "Countering real-time stream poisoning: An architecture for detecting vessel spoofing in streams of ais data," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing*, 2018, pp. 981–986.

[11] H. Park, S.-H. Ham, T. Kim, and D. An, "Object recognition and tracking in moving videos for maritime autonomous surface ships," *Journal of Marine Science and Engineering*, vol. 10, no. 7, p. 841, 2022.

[12] 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.

[13] M. H. Zwemer, R. G. Wijnhoven, and P. H. de With, "Ship detection in harbour surveillance based on large-scale data and cnns." in *VISIGRAPP (5: VISAPP)*, 2018, pp. 153–160.

[14] W. Man and L. Zhiyong, "The information fusion based on ais and video data," in *2016 5th International Conference on Computer Science and Network Technology (ICCSNT)*. IEEE, 2016, pp. 336–339.

[15] 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.

[16] Z. Huang, Q. Hu, Q. Mei, C. Yang, and Z. Wu, "Identity recognition on waterways: A novel ship information tracking method based on multimodal data," *The Journal of Navigation*, vol. 74, no. 6, pp. 1336–1352, 2021.

[17] R. W. Liu, Y. Guo, J. Nie, Q. Hu, Z. Xiong, H. Yu, and M. Guizani, "Intelligent edge-enabled efficient multi-source data fusion for autonomous surface vehicles in maritime internet of things," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1574–1587, 2022.

[18] A. Graser, "An exploratory data analysis protocol for identifying problems in continuous movement data," *Journal of Location Based Services*, vol. 15, no. 2, pp. 89–117, 2021.

[19] Ø. K. Helgesen, A. Stahl, and E. F. Brekke, "Maritime tracking with georeferenced multi-camera fusion," *IEEE Access*, vol. 11, pp. 30 340–30 359, 2023.

[20] N. Wawrzyniak, T. Hyla, and A. Popik, "Vessel detection and tracking method based on video surveillance," *Sensors*, vol. 19, no. 23, 2019.

[21] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 to yolov8 and beyond," *arXiv preprint arXiv:2304.00501*, 2023.

[22] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[23] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[24] Z. Jiang, L. Su, and Y. Sun, "Yolov7-ship: A lightweight algorithm for ship object detection in complex marine environments," *Journal of Marine Science and Engineering*, vol. 12, no. 1, 2024. [Online]. Available: https://www.mdpi.com/2077-1312/12/1/190

[25] J. Qu, R. W. Liu, Y. Guo, Y. Lu, J. Su, and P. Li, "Improving maritime traffic surveillance in inland waterways using the robust fusion of ais and visual data," *Ocean Engineering*, vol. 275, p. 114198, 2023.

[26] Y. Lu, H. Ma, E. Smart, B. Vuksanovic, J. Chiverton, S. R. Prabhu, M. Glaister, E. Dunston, and C. Hancock, "Fusion of camera-based vessel detection and ais for maritime surveillance," in *2021 26th International Conference on Automation and Computing (ICAC)*. IEEE, 2021, pp. 1–6.

[27] B. Carrillo-Perez, S. Barnes, and M. Stephan, "Ship segmentation and georeferencing from static oblique view images," *Sensors*, vol. 22, no. 7, p. 2713, 2022.

[28] F. A. Palmieri, F. Castaldo, and G. Marino, "Harbour surveillance with cameras calibrated with ais data," in *2013 IEEE Aerospace Conference*. IEEE, 2013, pp. 1–8.

[29] P. W. Anita Graser and M. Dragaschnig, "The m³ massive movement model: a distributed incrementally updatable solution for big movement data exploration," *International Journal of Geographical Information Science*, vol. 34, no. 12, pp. 2517–2540, 2020.

[30] A. S. Andersen, A. D. Christensen, P. Michaelsen, S. Gjela, and K. Torp, "Ais data as trajectories and heat maps," in *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, 2021, pp. 431–434.