IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 28, 2019, accepted July 29, 2019, date of current version September 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2935794

Extraction of Maritime Road Networks From Large-Scale AIS Data

GUILING WANG[1,2], JINLONG MENG[1], AND YANBO HAN[1]

[1]Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, North China University of Technology, Beijing 100144, China
[2]Ocean Information Technology Company, China Electronics Technology Group Corporation (CETC Ocean Corp.), Beijing 100144, China

Corresponding author: Guiling Wang (wangguiling@ncut.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1402500, in part by the National Natural Science Foundation of China under Grant 61832004 and Grant 61672042, in part by the Beijing Natural Science Foundation under Grant 4172018, and in part by the University Cooperation Projects Foundation of CETC Ocean Corp.

ABSTRACT Extracting road network information including lane boundaries, lane centerlines, junctions and their relationship from AIS data plays an important role in location based services, urban computing and intelligent transportation systems, etc. However, AIS data are large scale, high noisy, the density and quality are very uneven in different areas, extracting a whole, continuous and smooth maritime road network with rich information from such data is a challenging problem. To address these issues, this paper proposes an adaptive maritime road network extraction approach that can extract both lane boundaries and centerlines for a large sea area from AIS data. Based on a road network definition including nodes, segments and segment curves, the approach designs parallel grid merging and filtering algorithms to determine if a grided area is inside lane or not. Lane boundaries are smoothed through jagged edge filtering and Simple Moving Average algorithms before centerline extraction. We evaluate our method based on real world AIS data in various area across the world's seas. Experimental results show the advantage of our method beyond the close related work.

INDEX TERMS AIS data, road network, spatio-temporal data mining, trajectory data mining, trajectory computing, visual analysis.

I. INTRODUCTION

The advance in IoT and Cloud Computing technologies generated and collected massive spatial trajectories representing the locations of vehicles, such as automobiles, bicycles and mobile phones. Such unprecedented massive trajectory data can be used for transportation monitoring and early warning, trajectory prediction, anomaly detection, travel route plan etc., fostering or enhancing a broad range of applications in location based services, urban computing and intelligent transportation systems etc [1], [2]. Extraction of road networks from massive trajectory data is one of such important research topic attracting the attention of researchers [3]–[5].

The traditional way to acquire geographical road network information is through special equipped vehicles taken by trained, dedicated personnel driving on the streets [6], through analyzing high resolution satellite images [7], [8], or through manual crowdsourcing like OpenStreetMap (OSM) project [9] based on the editing and updating contribution from volunteers. Compared with the traditional approaches, extraction of road network from trajectories is more cost-effective, flexible and has higher time performance [10], [11]. Since the trajectory data can be classified and analyzed according to different kinds of vehicles, the factual and detailed road information for different kinds of vehicles can be extracted, and the changes of road can be reflected in a timely manner [12]. Trajectory data for urban computing are often collected from the Global Positioning System (GPS) terminals, while in maritime traffic, Automatic Identification System (AIS) terminals can provide us reliable source information for trajectories of vessels. Although there are no artificially constructed roads in physical form on the sea, in fact, vessels typically follow some de facto standardized maritime routes due to some reasons such as fuel-efficiency aspects, existing protected sea areas where maritime traffic is prohibited and some well-known security threats. As extracting the road information from the crowdsourced GPS data, extracting shipping lanes on the sea accurately from the crowdsourced AIS data also offers interesting applications.

The associate editor coordinating the review of this article and approving it for publication was Shuiguang Deng.

However, extraction of maritime road networks from AIS data is challenging because of the following aspects:

1) **Large volume, high noise, uneven density and quality.** Trajectories collected from large number of vessels for a long period are in large volume and tend to be distorted with noisy data hiding the vessel's real location. In our global AIS dataset, there are billions of trajectory points (records) per month, over 100GB and over 370,000 active ships, and over 1TB per year. The sampling frequency of the trajectory points in the offshore and nearshore water area ranges from 5 seconds to 100 seconds and from 2 minutes to 10 minutes in the open sea. The density and quality of AIS data in the offshore and near shore areas are very different. The vessel trajectory points in the offshore and nearshore water areas are naturally more densely distributed than in the open sea areas. Additionally, the extraction precision requirement is low in open sea but high in offshore and nearshore water area. These characteristics have posed huge challenges to road network extraction algorithms.
2) **Extract continuous and smooth maritime road networks.** Maritime road networks could include not only centerlines or lanes, but also boundary information, for boundary information is very useful in some scenarios such as warning of a vessel out of a secure or legal area. We also believe a whole lane must be continuous and smooth, or else it is only a segment of a lane and cannot be well used in real application. Extracting such a whole, continuous and smooth maritime road network with rich information in a large area is non-trivial.

To meet the above challenges, we propose a framework called ''Maritime Road Network Extraction from Crowdsourced Trajectories (MaritimeNET)'' to extract maritime road network from massive AIS data. The contributions of the research are:

1) We propose a maritime road network extraction approach based on adaptive grid merging and filtering and Delaunay Triangulation. The 3-phase approach includes: a) Through parallel trajectory insertion, deletion, segmentation and clustering based on GeoHash encoding, quality of trajectory data is improved and volume is reduced as grid data. b) Through parallel grid merging and filtering algorithms, those grids inside lanes are selected; through Delaunay triangles construction and filtering, polygon based lane boundaries are generated and extracted. c) Through maritime traffic boundary smoothing, second triangles construction and triangles classification and filtering, lane centerlines and junctions are extracted and maritime road networks are built up. Since the first two stages have been introduced in another paper, this paper mainly focuses on the third stage.
2) Experimental results show that the proposed method can automatically effectively extract maritime road network from crowdsourced data in a large area with high noise and very uneven density and quality, by comparing with the related work.

The rest of this paper is organized as follows. Section II reviews related work. In Section III, the basic concepts throughout the paper are introduced and the problems to be solved are described. Section IV describes the preprocessing algorithm. Section V describes the grid merging, grid filtering, and the maritime road boundary extraction algorithms. Road centerlines extraction and road network building algorithms are described in Section VI. Section VII evaluates our approach and compares it with current methods. Section VIII concludes the paper.

II. RELATED WORK

The existing related research work on road network extraction from trajectory data can be classified into three categories:

1). Vector-based approaches: Transform trajectory points into vectors and then extract the road network information. Most of vector-based approaches are based on clustering algorithms. Point clustering methods cluster the trajectory points into way points and then find the information to connect them into roads or lanes [13]–[19]. Segment clustering methods cluster trajectory segments to extract road segments [20]–[22]. Others use statistical method of motion attributes to find way points and lines [23]–[25] or extract the road lines by incrementally inserting and merging tracks based on their geometric relations and/or shape similarity [26]. The above methods can be used to extract road lines or segments but is difficult to be used to extract lane boundaries or whole lane boundaries in a large area. Furthermore, the methods are applicable to densely sampling traces, but not robust to position points with noises and varying density.

2). Image-based approaches: Transform the location data of all the trajectory points into row and column coordinates of image, and then recognize and/or extract the lane information using digital image processing technology [5], [27], [28]. The image-based methods are also applicable to densely sampling traces, but not robust to noisy data and have efficient problem when applying to massive trajectory data.

3). Grid-based approaches: Transform trajectory points into grids, and then extract road information from grid data [10], [29]–[31]. Some authors [10], [29], [31] apply Delaunay Triangulation to extract road boundaries and road centerlines. Others [30] use the number of trajectory points in grids (grid heat value) to determine the road direction, however the lane boundaries and junctions have not been considered in their paper. The grid-based approach can be used to extract both boundary information and centerlines, and is more robust to data with noises and varying density than the other two approaches. The existing research papers haven't discussed how to apply their algorithms on massive AIS data and how to extract a continuous and smooth lane in a large area, which is very challenging when the data volume is large, the data quality is low and the trajectory points density is quite uneven.

Different from these related work, this paper extracts both marine lane boundary and lane centerlines. The framework and approach proposed in this paper can deal with massive trajectory data efficiently using algorithms based on parallel computing, and the challenges of extraction lane networks from noisy and density uneven trajectory data are also tackled by various ways such as grid merging, sliding-window filtering and boundary smoothing algorithms.

III. DEFINITIONS AND OVERVIEW

This section first gives a few basic concepts, then describes the problem we are aiming to solve and overviews the approach.

A. DEFINITIONS

Definition 1 (Vessel Trajectory): A vessel Trajectory T_{v_i} is a spatio-temporal point sequence of a vessel v_i, that represents the sequence of positions of v_i over a period of time. $T_{v_i} = (v_i, (p_0, p_1, \ldots, p_N))$, where v_i represents the maritime mobile service identity (MMSI) of the vessel, $p_j = (x_{i,j}, y_{i,j}, t_j)$ indicates the position of the vessel at a certain moment, where t_j is the sampling (or collecting) time of the position, $x_{i,j}$, and $y_{i,j}$ represents the longitude and latitude of v_i at t_j.

Definition 2 (Marine Lane Boundary): A marine lane boundary P_{lane} is a two-dimensional polygon representing the area where ships are allowed to sail in. $P_{lane} = (p_0, p_1, \ldots, p_n)$, in which the set of vertices p_i constitutes a polygon P in a clockwise direction.

Definition 3 (Grid): A grid is a rectangular area on a map. By dividing the 2D geospatial space through horizontal and vertical direction, the whole geographical space is divided into multi equal rectangles in size. Each rectangle is called a grid. Grid can be described as $Grid = (Code, Dsy)$, where:

Code: GeoHash code of a grid. Code can be obtained based on the GeoHash algorithm [32] encoding the location of the grid center including longitude and latitude. It is a string constituted by 0 or 1.

Dsy: grid density. *Dsy* is defined as the number of AIS points in one grid g, $Dsy = |\{g_{sw_{lon}} < p_{lon} < g_{ne_{lon}}, g_{sw_{lat}} < p_{lat} < g_{ne_{lat}}, p \in P\}|$, where p indicates a AIS point, p_{lon} and p_{lat} indicate the longitude and latitude of an AIS point p, g_{sw} and g_{ne} indicate the south-west and north-east boundary point of g.

Definition 4 (Parent Grid): If a grid is divided into four sub-grids by separating the grid into two parts in both the latitude and longitude directions respectively. The divided grid is called the parent grid of the four sub-grids. In fact, for the parent $Grid_{par} = (Code_{par}, Dsy_{par})$, $Code_{par}$ is the prefix of any of its sub-grid's *Code*. That is, $Code_{par} = subString(Code_{sub}, 0, |Code_{sub}| - 2)$, and the parent grid's density is the sum of four sub-grids' density: $Dsy_{par} = \sum Dsy_{sub}$.

Definition 5 (Marine Lane Extraction Precision): GeoHash offers properties like arbitrary precision and the possibility of gradually removing characters from the end of the code to reduce its size (and gradually lose precision) [32]. If we let the size of the GeoHash code ($|Code|$) of a grid represent the precision of this grid, given a trajectory data set in an area to extract marine lane, the mean precision of all the grids in this area used to extract the marine lane is called the marine lane precision. That is, $Precision_{lane} = \frac{\sum precision_g}{|G|}$, where G indicates the grid set used to extract marine lane, g is a grid in G.

Definition 6 (Marine Lane Network Node): A node *ver* in a marine lane network is a lane centerline junction.

Definition 7 (Segment): A segment of a marine lane network is an edge or arc $e = (ver_i, ver_j)$, which is expressed by two adjacent nodes ver_i and ver_j where there exists a centerline between them. An edge with direction information is called arc *arc*.

Definition 8 (Segment Curve): A segment curve of a marine lane network is a curve formed by discrete points $curve = ((c_0, c_1, \ldots, c_k), (g_0, g_1, \ldots, g_l)$, where $c_i = (x_{c_i}, y_{c_i})$, x_{c_i} and y_{c_i} are the longitude and latitude of a position point c_i, g_i is the grid (Definition 3) which this segment curve passes.

Definition 9 (Maritime Road Network): A maritime road network $NET = (V, E, C)$ is a 3-tuple graph, where V is a set of nodes (Definition 6) on maritime road network, E is a set of graph edges i.e. segments (Definition 7) and C is a set of segment curves (Definition 8).

B. PROBLEM AND APPROACH OVERVIEW

Our goal in this paper is to extract rich marine lane network information from massive crowdsourced trajectory data, that is, given an AIS trajectory dataset $\{T_v\}$, we aim to extract the marine lane network *LaneG* in a certain area.

As analyzed before in Section I, it is a very challenging problem to extract rich marine lane network information from the large-scale, high-noisy, and density uneven crowdsourced AIS data. The approach has 3-phases:

The first phase is preprocessing. In this phase, we insert missing trajectory points, delete erroneous or redundant data, and reduce the data volume by clustering the trajectory data using an algorithm based on GeoHash encoding. In this phase, the quality of trajectory data is improved and volume is reduced, the trajectory dataset $\{T_v\}$ is transformed into a set of grids $\{Grid\}$. Thus the problem of marine lane network extraction is then transformed into a problem of filtering the grids which is not on the marine lane in the first place and then extract boundaries, centerlines and junctions based on the grids.

The second phase is marine lane boundaries extraction. As the density of trajectory data is uneven in the open sea and nearshore regions, we cannot use a global unified threshold to filter the grids. Therefore, the key is finding an adaptive thresholding method to filter them. The main idea to meet this challenge is to merge the sparse small grids with different lower densities by a grid merging algorithm based on QuadTree, and then use a sliding window filtering algorithm with different thresholds for different windows to filter the merged grids. After that, we triangulate the grids by

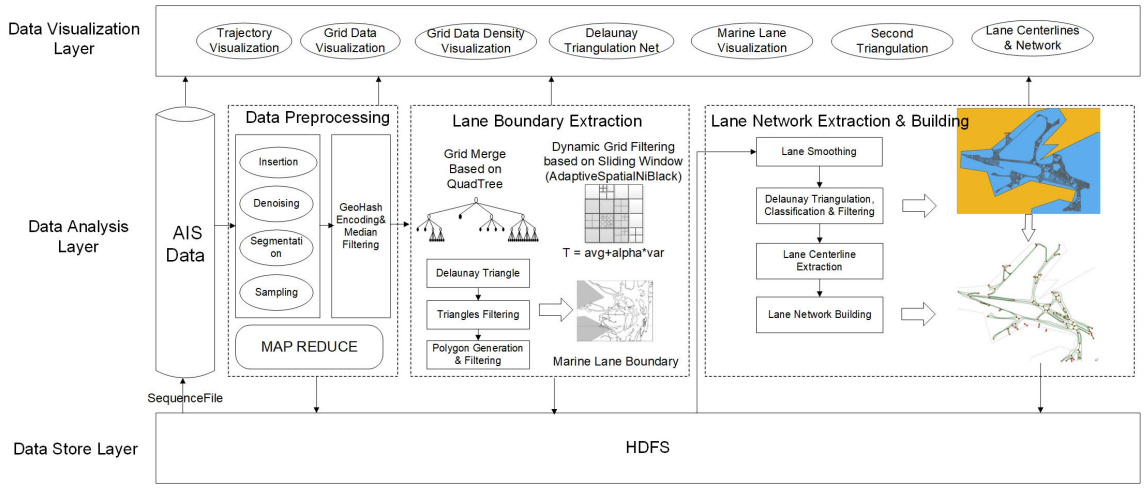


FIGURE 1. MaritimeNET: A framework for maritime road network extraction.

Delaunay, filter the triangles not in lanes and generate polygons as the lane boundaries.

The last phase is marine lane network building. Firstly we smooth the marine lane boundaries, then construct the second triangles, classify and filter them, extract and generate the lane centerlines and junctions. After that, the maritime road network is built up. Since the first two stages have been introduced in another paper, this paper focuses more on the third stage.

Figure 1 presents an overall framework called MaritimeNET of this approach. The data storage layer is responsible for storing all raw AIS data and intermediate process data as well as marine lane extraction results. The core algorithm layer includes data preprocessing and GeoHash encoding based on the MapReduce parallel computing framework. The marine lane grid information is obtained by a grid merge algorithm based on QuadTree and a dynamic sliding window filtering algorithm based on merged QuadTree. Then in this layer, we triangulate the marine lane grid by Delaunay and extract the results using a boundary extraction algorithm based on triangle circumcircle radius. After that, maritime road networks are built up. The data visualization layer is responsible for visualizing the intermediate and final results, observing the data exception, and providing parameter guidance for the algorithms.

IV. TRAJECTORY DATA PREPROCESSING

The aim of preprocessing is to improve the data quality. According to the statistical analysis and observation, three kinds of problems in the original AIS data can be solved in the preprocessing phase: 1). Unsegmented trajectory points. Trajectory points belonging to different trips but haven't been split into different segments; 2). Noisy trajectory points. Those points that cause the calculated speed deviating from normal reasonable ship speed and those too short trajectories can be seen as noisy data; 3). Missing trajectory points. If the time interval between two trajectory segments belongs to the same trip is larger than the average time interval between adjacent points, there must exist some missing points between these two trajectory segments. For the unsegmented points, we can segment trajectory data by setting a maximum segmentation time interval threshold. If the time interval between two adjacent points is less than this threshold, they belong to the same segment. For the noisy data, we can delete the noisy data directly instead of correct them. Deletion of some points will not have apparent negative influence, because the volume of AIS data is very large. For the missing points, we use linear interpolation method to insert some missing points. It is because different insertion methods don't have apparently different influence on the results of the next steps. So we just choose a linear interpolation method.

After the above preprocessing, we propose a trajectory point clustering method based on the GeoHash encoding algorithm to simplify the AIS data. GeoHash encoding is a classic method of encoding geographic data. This method recursively divides the entire geographical space into multi grids along the longitude and latitude direction separately, and obtains a grid map. Each grid corresponds to a GeoHash code. The AIS data points in the same grid have the same GeoHash code, and we can use the location of the center point of a grid to represent all the trajectory points in this grid. As Definition 3 defined, we use the number of AIS points in one grid to represent its density.

In this way, we can transform the trajectory points data (or AIS data) into the grid data in form of tuples (C_x, C_y, dsy). We also borrow the modified median filtering algorithm from image fuzzy processing to further remove isolated noise points and make the density variation of all grids smoother and more uniform.

V. THE ADAPTIVE AND PARALLEL MARINE LANE BOUNDARY EXTRACTION ALGORITHM

The main idea of the adaptive marine lane boundary extraction algorithm is that to merge the grids of those areas where

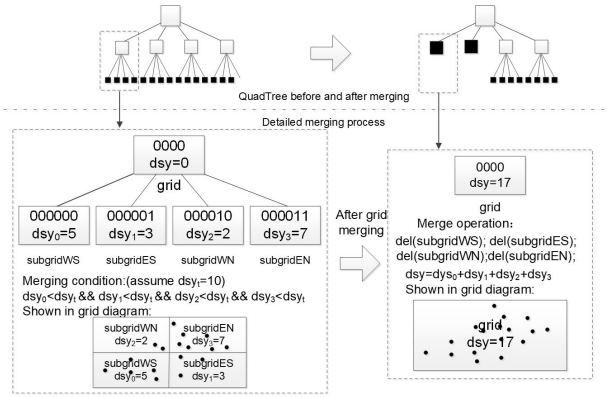


FIGURE 2. A grid merging example.

the overall average density is low, so that we can extract those areas with high density using high extraction precision and those areas with low density with low extraction precision. Then filter the merged grids with different thresholds for different windows using a sliding window filtering approach. After that, transform the merged grids as triangles based on Delaunay Triangulation, filter the triangles not in lanes and generate polygons as the lane boundaries.

We create a QuadTree to store the grid data for efficiency. Each node corresponds to a grid, stores a grid with the GeoHash code and density value as two attributes (according to Definition 3), except for the root node which does not store the code and density value. The more layers of this QuadTree, the smaller grid size and according Definition 5, the extraction precision will be higher. Given a demand of extraction precision range, the depth of this QuadTree can be determined.

For grid merging, we should set three parameter values: the highest marine lane extraction precision bit_{nummax} the lowest marine lane extraction precision bit_{nummin}, and the merge density threshold dsy_t. Then, the entire geographic range is divided into grids of the same size according to the highest grid precision. After that, the algorithm begins the merging process. The process is carried out with four sub-grids that have the same parent grid as a unit. If all the density value of the four sub-grids are lower than dsy_t, the four sub-grids are merged into one parent grid. And density value of the parent grid is updated to the sum of the four sub-grids. Otherwise, the merge operation will not be performed on the four sub-grids. After the first layer traversal is completed, the secondary high-precision layer begins to judge and merge in the same way. The algorithm thus traverses the QuadTree layer by layer from bottom (e.g. from the layer where the girds represent the highest precision) to up. The merging process is completed when the grid has reached lowest extraction precision (e.g. bit_{nummin}). Figure 2 shows a grid which satisfies the merge condition and performs the merge operation. The change of the data structure is also shown in Figure 2.

The parallel grid merging algorithm based on Spark [33] is shown in Algorithm 1:

In line 3, the algorithm uses the first N bits of GeoHash code as the key. According to the geographic characteristics of the GeoHash algorithm, those grids that have the same parent grid have the same prefix code. The parallel algorithm divides the geographic area into $2^{N/2}$ big grids, and then computes parallelly for all the divided big grids.

In line 12, the algorithm establishes a QuadTree T to store the grid data.

In lines 13 - 17, the algorithm traverses the QuadTree T from bottom to top layer by layer. If the precision of a node is not below the minimum precision threshold and not above the maximum precision threshold, the node is added to a queue Q.

In lines 18 - 23, the algorithm traverses the queue Q in reverse order, that is, starts from the second bottom layer, determines whether all the density values of the four sub-nodes of a node are smaller than the merge density threshold dsy_t. If it is true, set the density value of the node to the sum of all sub-node's density values and delete these four child nodes. Otherwise, do nothing.

Algorithm 1 The Parallel Marine Lane Extraction Algorithm

Input: G: grid data
Output: *Polygons*: lane boundary information

```
1:  for each grid g in G do
2:      function mapToPair(g)
3:          k=g.getNprefix(g.code)
4:          Out(k, (g.code, g.dsy))
5:      end function
6:  end for
7:  function reduceByKey(k, (g.code,g.dsy))
8:      Out(k, putAll(g.code, g.dsy))
9:  end function
10: function map(k, (g.code, g.dsy))
11:     for each g in partition do
12:         T = initGeohashTrie(g)
13:         for each node T do
14:             if node.depth in range(accuracy) then
15:                 Q.add(node)
16:             end if
17:         end for
18:         for q in Q do
19:             if g.subnodes.dsy < dsy_t then
20:                 g.dsy = sum(g.subnodes)
21:                 del(g.subnodes)
22:             end if
23:         end for
24:         Ret = AdaptiveSpatialNiBlack(T)
25:         saveAsTextFile(Ret, outputPath)
26:     end for
27: end function
28: generate Polygons based on Delaunay Triangulation
29: filter Polygons with an area threshold
```

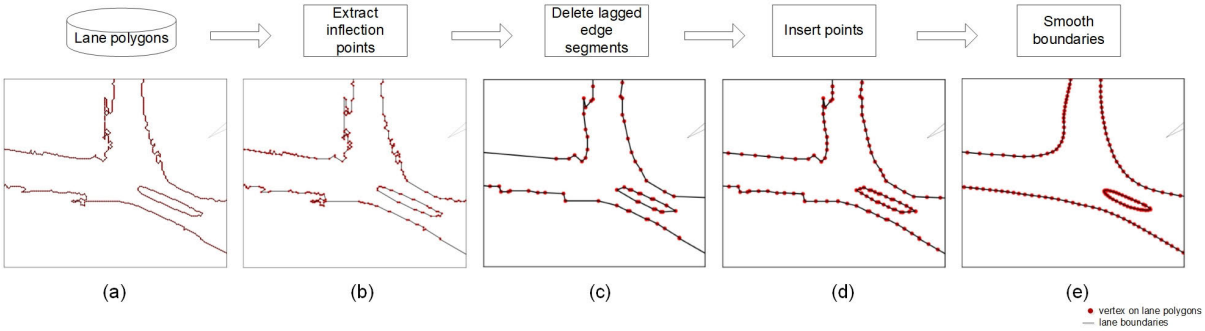


FIGURE 3. Framework for lane polygon preprocessing.

After the grid merging process, we get a set of grids satisfying our restrictions of extraction precision. Then we design an algorithm to distinguish which grids are on the marine lane and which are not. Instead of using a global threshold to filter all the grids, the idea is to filter the grids with different thresholds for different sets of grids. So we borrow the idea of the NiBlack binary filtering method [34] which is an image processing algorithm. We call it AdaptiveSpatialNiBlack algorithm (line 24). We look on each grid as a pixel with the density value as the pixel value, and then use a sliding window filtering similar with NiBlack to separate the grids as two categories that are inside lanes and that are outside lanes.

In AdaptiveSpatialNiBlack algorithm, we filter the grids in a sliding window based on the NiBlack equation. Let T represent the filtering threshold, which means if the density of the center grid is greater than T, keep it, otherwise filter it. The calculation equation of the filtering threshold T is as follows:

$$T = avg + alpha \times var \tag{1}$$

In equation 1, *avg* represents the average density value of all the grids in a sliding window, *var* represents the variance density value of all the grids in the window, and *alpha* represents the variance correction factor.

Details of the algorithm AdaptiveSpatialNiBlack are introduced in our previous paper [35].

After the above steps, the result grid data is all the grids inside marine lane. Before we start to extract the boundary of the marine lane, we use the mean value of density of the surrounding grids to replace the density value of the center grid-unit. This can further remove the clutter of result grid data and smooth the density distribution of the result grid data.

Then we can extract boundary information based on Delaunay Triangulation model. In line 28 and line 29, we firstly construct the Delaunay triangles for the result grid data. Because the triangles inside the lane are denser and thus has shorter circumscribed circle radius, we set a threshold to filter the edges of the result triangles. At last, we extract the boundary edge set using a classic algorithm for polygon generation, and filter those small polygons.

VI. MARITIME ROAD NETWORK EXTRACTION

In this section, we describe the core strategy and algorithms for solving the problem of maritime road network extraction. We first focus on extracting the inflection points, deleting jagged edges, inserting points and smoothing lane boundary (VI-A). We then analyze how to generate, filter and classify the triangles inside lanes based on the smoothed lane polygons (VI-B) and finally, we develop a centerline extraction method based on the classified triangles and the corresponding maritime road network extraction algorithm, we also introduce how the maritime road network can be used to transform trajectories into meaningful road segment series for further various applications (VI-C).

A. MARINE LANE SMOOTHING

As introduced in the above Section V, we can extract lane boundaries which are a set of polygons. However, most of the boundaries are not smooth. We can see sawtooth boundary shown in Figure 3(a), a part of lane extracted from the AIS data in Bohai Sea on the coast of Northeastern China. The centric line extracted from such sawtooth-shaped boundary will be jagged or serrated, unlike an ordinary road line. Therefore, lane polygon boundaries extracted from Section V should be smoothed before the lane centerline extraction is performed.

As shown in Figure 3, the whole lane boundary smoothing process includes inflection points extraction, short sides deletion, points interpolation, and line smoothing.

Firstly, we extract the inflection points of each lane polygon boundary. Equation 2 can be used to determine if a point is an inflection point on a lane boundary. Suppose p_{i-1}, p_i, p_{i+1} are three adjacent vertices on a lane boundary, s_i, the degree of bending (sinuosity) of point p_i can be calculated as Equation 2:

$$s_i = \frac{dis(p_{i-1}, p_i) + dis(p_i, p_{i+1})}{dis(p_{i-1}, p_{i+1})} \tag{2}$$

In this equation, $dis(p_i, p_j)$ is the distance between p_i and p_j. The value of s is always greater than or equal to "1". When s is equal to "1", the three points are on the same straight line. The greater the value of s, the higher the degree of sinuosity. If s is greater than a threshold $ThresS$, the point p_i can be

determined as an inflection point. Inflection points are kept and others are left out. Since the target of extracting inflection points is to smooth them, the threshold can not be set too great, or else much details will be omitted from the beginning. In our experiment, *ThresS* is set to "1.03".

Secondly, we delete jagged edge segments from the lane boundary. After the above processing, now the vertices of the lane polygon are composed of inflection points (As shown in Figure 3(b). Considering that the length of jagged edge segments is generally shorter than ordinary edges, we can make the boundary smoother by just deleting the short edges. So we calculate the length of each edge segment of a channel polygon, if the length an edge segment is less than a threshold *ThresDis*, delete this segment.

The threshold values of *ThresDis* can be determined by Equation 3, where *Mean* is the average value of all lengths of edge segments in that lane polygon, *StdDev* is their standard deviation values and α is an adjustment factor. The greater the value of α, more local variance be reduced, but if too great, too much points or edge segments will be deleted and the kurtosis of lane boundaries may be hidden.

$$ThresDis = Mean + \alpha * StdDev \tag{3}$$

Thirdly, after extracting the inflection points and deleting the jagged edge segments, the vertices of the lane polygons become sparse, as shown in Figure 3(c), which is not conducive to centerline extraction, so interpolation is required. Insert one or more points linearly between two adjacent vertices, in which the step length is the mean of the lengths of the polygon edge segments. The example result is shown in Figure 3(d).

After the operations described above, the jagged edges of the lane polygon are reduced a lot, finally, Moving Average (MA) algorithm and its variants can be used to further smooth the lane boundary. Here, for algorithm efficiency and at the same time taking account of the effectiveness, Simple Moving Average (SMA) algorithm is adopted. The principle of SMA is: start from the first vertex of the lane polygon, take the current vertex as the center vertex, generate a window of length *WinSize*, calculate the mean value of all vertex coordinates in the window, and use it to replace the coordinate of the center vertex, loop such replacement operation until all vertices in the polygon are traversed. The equation to calculate the coordinates of center vertex p_i is as Equation 4:

$$p_i = \frac{1}{WinSize} * \sum_{j=i-WinSize/2}^{i+WinSize/2} (p_j) \tag{4}$$

To determine the value of parameter *WinSize*, two metrics are introduced to assess the quality of smoothing effect. The first metric is to quantify the degree of smoothing. Given a lane boundary $P_{lane} = (p_0, p_1, ldots, p_n)$ (Definition 2), the first difference of sinuosity of P_{lane} is $\Delta S = (\Delta s_0, \Delta s_1, \ldots)$, where $\Delta s_i = s_{i+1} - s_i$, and s_i is defined in Equation 2, the roughness of a lane boundary can be defined as the standard deviation of ΔS shown in Equation 5:

$$Roughness = \sigma(\Delta S) \tag{5}$$

The greater the value of roughness, the lower the degree of smoothness, a straight line has roughness value of 0.

If we simply minimize roughness, lane boundary may be over-smoothed and straight lines may be produced, but we want to preserve some apparent road bending information. Therefore, the second metric is to quantify the degree of preservation of large scale deviations within the lane boundary. Given a series of S of P_{lane} with its mean μ and standard deviation σ, kurtosis of P_{lane} is defined as the fourth standardized moment as Equation 6:

$$Kurt[P] = \frac{E[(S-\mu)^4]}{E[(S-\mu)^2]^2} \tag{6}$$

Suppose P' is the smoothed lane of P, if $Kurt[P']$ is greater than or equal with $Kurt[P]$, the large deviation are strengthened after smoothing. Since S can be seen as a set of identically independently distributed random variables, roughness linearly decreases with increased window size, and kurtosis monotonically increases with window length for S with initial kurtosis less than 3, and decreases with initial kurtosis larger than 3. Binary search approach can be used to search the largest window that can minimize roughness and at the same time guarantee the kurtosis constraint [36].

After the above steps, we can finally get a smooth lane polygon, as shown in Figure 3(e).

B. TRIANGLE CLASSIFICATION AND FILTERING

Based on the smoothed marine lane polygons, we can build triangular network and classify the triangles, so that key points in the triangles can be extracted and connected as marine lane center lines. First, we use the method of Delaunay Triangulation to build a triangular network based on the smoothed marine lane polygons (as shown in Figure 4 (a)). Note that it is different from the triangular network built in Section V, which is based on the merged and filtered grid data and for the purpose of extracting the marine lane polygons.

The triangles are distributed inside the lane (as shown in Area(1) in Figure 4(a)), outside of the lane (as shown in Area(3-5) in Figure 4(a)) and inside the cavity(as shown in Area(2) in Figure 4(a)). Those triangles outside the lane and inside the cavity are meaningless for the extraction of center lines (called *non-lane* triangles), so we filter them according to their barycentric coordinate, and the result is shown in Figure 4(b).

In order to determine the key points on center lines, the next step is to classify triangles. We can classify triangles according to the number of edged shared by two triangles, which is called common edges (represented by heavy lines in Figure 5), into three categories: *one-neighbor* triangle, *two-neighbor* triangle and *three-neighbor* triangle. As shown in Figure 5, we can see each *one-neighbor* triangle has one common edge and is located at the road polygon exit, each

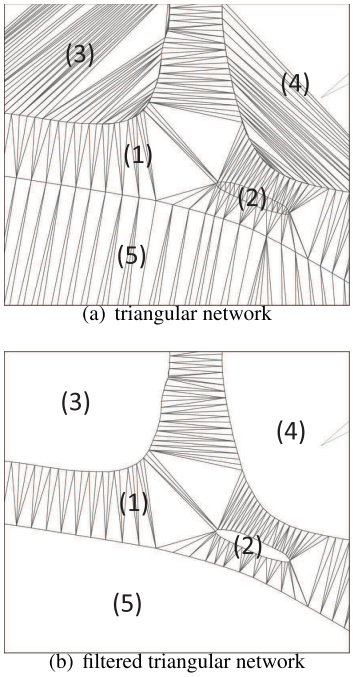


FIGURE 4. Triangulation and filter.

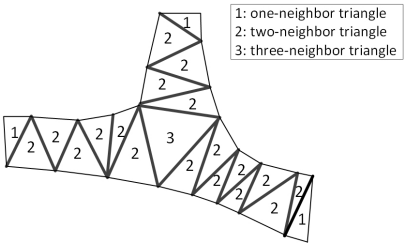


FIGURE 5. Triangle classification.

two-neighbor triangle has two common edges and is located on the road trunk, and each *three-neighbor* triangle has three common edges and is at the road intersection.

We build up three tables to maintain the triangle information: vertex table, edge table and triangle table (Table 1). Vertex table saves information of vertices such as *ID*, *longitude* and *latitude*. Edge table saves the *ID*, two vertices, edge type (private edge or common edge) of an edge. For the sake of algorithm efficiency, for each common edge, the identification list of the triangles that have this common edge are stored in edge table as a field *TrianglesList*. Triangle table saves information of triangles including *ID*, *ID*s of the three edges triangle types, the set of adjacent triangles *AdjTriList* and the number of visits to this triangle with an initial value of 0 *VisitedNum*. If a triangle is a *one-neighbor* triangle, *Edge*1 and *Edge*2 are private edges, and *Edge*3 is a common edge; for a *two-neighbor* triangle, *Edge*1 is a private edge, *Edge*2 and *Edge*3 are common edges; and for a *three-neighbor* triangle, *Edge*1 *Edge*2 and *Edge*3 are common edges. An example triangle table is shown in 1.

TABLE 1. Triangle table.

ID	Edge1	Edge2	Edge3	Type	AdjTriList	VisitedNum
0	1	2	3	1	[1]	0
1	1	5	6	2	[4,6]	0
2	7	9	10	3	[5,7,8]	0
…	…	…	…	…	[…]	…

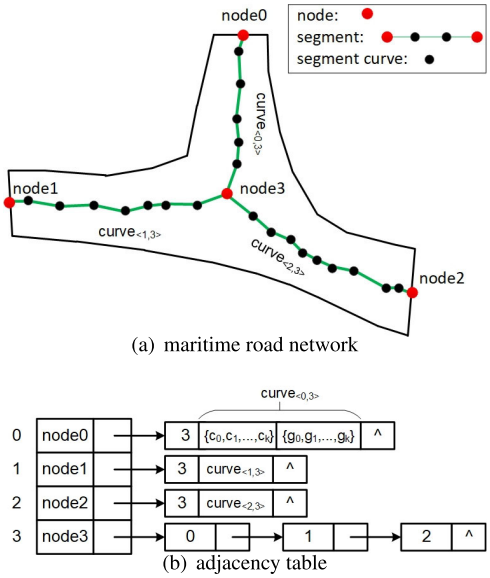


FIGURE 6. An example maritime road network and data structure.

C. CENTERLINE EXTRACTION AND ROAD NETWORK BUILDING

The main idea of maritime road network extraction and building is to extract different key points from different kinds of triangles generated from Section VI-B separately and connect them into lane centerlines, and then extract the information of nodes, segments and segment curves from the centerlines and add them to the road network data structure.

As defined in Definition 9, we should extract information from the triangles to build a maritime road network consisting of three elements: nodes, segments and segment curves. Figure 6(a) shows an example maritime road network, a node in a maritime road network is a road centerline junction, a segment is an edge expressed by two adjacent nodes, and a segment curve is a curve formed by discrete points. Maritime road networks can be maintained in an adjacency table as shown in Figure 6(b). Algorithm 2 is designed to extract a centerline and add the edge to this adjacency table one by one.

In the algorithm, for a *one-neighbor* triangle, we extract the midpoint of the common edge and the midpoint of the longer edge of the other two edges (point *o* and *p* in Figure 7, Lines 8-11 in Algorithm 2); for a *two-neighbor* triangle, extract the midpoints of the two common edges (point *p* and *q* in Figure 7, Lines 18-19); if it is a *three-neighbor* triangle,

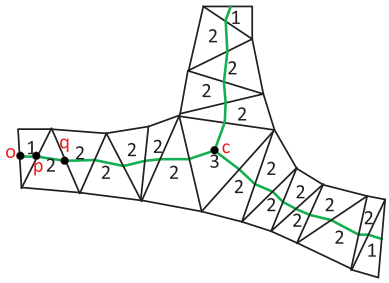


FIGURE 7. Center line of lane.

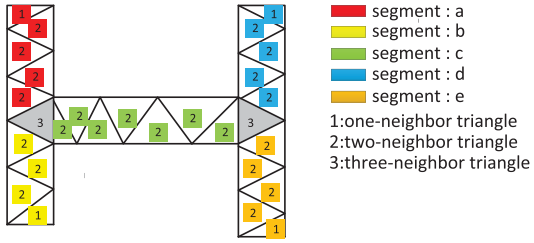


FIGURE 8. Centerline classification.

extract the midpoints of the three common edges and the center of gravity (point c in Figure 7, Line 24).

As shown in Algorithm 2, first, we define the processing methods for different types of triangles, processType1 (Line 3), processType2 (Line 15) and processType3 (Line 23) for dealing with *one-neighbor, two-neighbor* and *three-neighbor* triangles separately. For the *three-neighbor* triangle, each time the algorithm add the number of visits with 1 (Line 25). Then, the algorithm determines whether the number of visits to this triangle is greater than 3, if so, delete this triangle from the triangular network (Line 27).

According to the key points connection approach, centerlines can be classified into two types. The first type starts with a *one-neighbor* triangle and ends with a *one-neighbor* triangle or a *three-neighbor* triangle (as shown in a,b,d,e in Figure 8). The second type starts with a *three-neighbor* triangle and ends with a *three-neighbor* triangle (as shown in c in Figure 8). Algorithm 2 firstly traverses all the triangles of *one-neighbor* and extracts the first type of segment curve (Lines 31-51). Then traverses all the *three-neighbor* triangles and extracts the second type of segment curve (Lines 52-66).

From Figure 9, we can see how the algorithm works on an example triangular network 9(a). The algorithm starts to traverse the adjacent triangles from one of any *one neighbor* triangles, for example from a triangle with vertices (p_0, p_1, p_2), and c_0, c_1 is extracted. According to the common edge (p_1, p_2), determine the next triangle to visit is the triangle (p_1, p_2, p_3). Since this triangle is a *two-neighbor* triangle, extract midpoints of two common edges c_1 and c_2, and then determine to visit the triangle (p_1, p_3, p_7). Since this triangle is *three-neighbor* triangle, extract the center of gravity g_1. Now the traverse from the triangle (p_0, p_1, p_2) is finished, a segment curve r_1 (in green color) is extracted and

Algorithm 2 Maritime Road Network Extraction and Building Algorithm

Input: *VT*: vertex table
ET: edge table
TT: triangle table
Output: *NET*: road network

```
3:  function processType1(triangle)
4:      privateEdge1 = triangle.Edge1
5:      privateEdge2 = triangle.Edge2
6:      commonEdge = triangle.Edge3
7:      longPrivateEdge = privateEdge1
8:      if privateEdge2.length > privateEdge1.length then
        longPrivateEdge = privateEdge2
9:      end if
10:     p1 = longPrivateEdge.midpoint
11:     p2 = commonEdge.midpoint
12:     TT.delete(triangle)
13:     return (p1,p2)
14: end function
15: function processType2(triangle)
16:     commonEdge1 = triangle.Edge2
17:     commonEdge2 = triangle.Edge3
18:     p1 = commonEdge1.midpoint
19:     p2 = commonEdge2.midpoint
20:     TT.delete(triangle)
21:     return (p1,p2)
22: end function
23: function processType3(triangle)
24:     p = triangle.centerpoint
25:     triangle.visitedNum = triangle.visitedNum + 1
26:     if triangle.visitedNum == 3 then
27:         TT.delete(triangle)
28:     end if
29:     return (p)
30: end function
31: for triangle in TT.Type == 1 do
32:     curve.add(processType1(triangle) )
33:     for triangle.AdjTriList[t in TT] != null do
34:         triangle = t
35:         if triangle.Type == 1 then
36:             curve.add(processType1(triangle))
37:             NET.addEdge(curve)
38:             curve.clear
39:             break
40:         end if
41:         if triangle.type == 2 then
42:             curve.add(processType2(triangle))
43:         end if
44:         if triangle.type == 3 then
45:             curve.add( processType3(triangle))
46:             NET.addEdge(curve)
47:             curve.clear
48:             break
49:         end if
50:     end for
51: end for
```

Algorithm 2 *(Continued.)* Maritime Road Network Extraction and Building Algorithm

```
for triangle in TT.Type == 3 do
    curve.add(processType3(triangle) )
    for triangle.AdjTriList[t in TT] != null do
        triangle = t
        if triangle.type == 2 then
            curve.add(processType2(triangle))
        end if
        if triangle.type == 3 then
            curve.add(processType3(triangle))
            NET.addEdge(curve)
            curve.clear
            break
        end if
    end for
end for
```

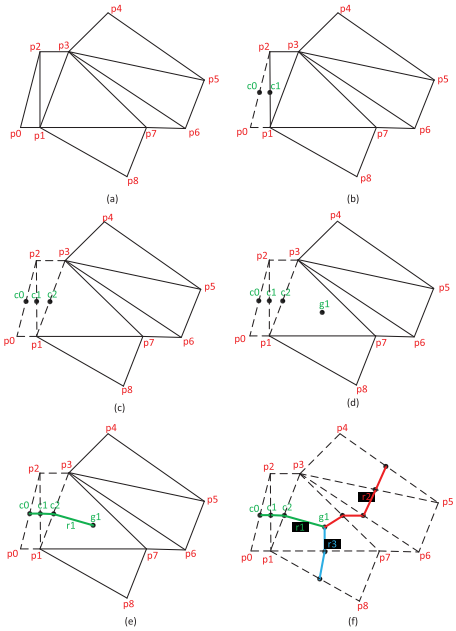


FIGURE 9. The extraction process of centerline.

the corresponding edge and node are added to the adjacency table of the existing maritime road network. For the unvisited triangles in 9(e), take the triangle (p_3, p_4, p_5) and (p_1, p_7, p_8) separately, find the corresponding centerline r_2 and r_3 and add the corresponding nodes and segment curves to the adjacency table.

Based on the maritime road network, ship trajectories can be matched and transformed into a series of road network elements. In Figure 10, the upper part of this figure shows the grids which the segment curves pass. The lower part shows when the trajectory is converted, the trajectory is first converted into a grid sequence, and then converted into a segment label sequence according to the segment label of

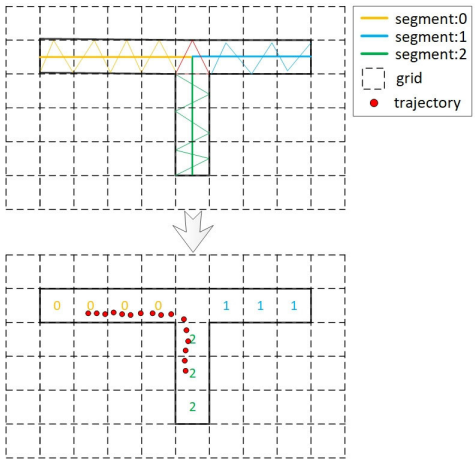


FIGURE 10. Trajectory transformation.

TABLE 2. Sample AIS data.

MMSI	Longitude	Latitude	Time
412351810	121.000671	30.449981	2016-6-10 12:00
412351810	121.000671	30.452728	2016-6-10 12:02
...	...	...	...
412351810	121.055603	31.045989	2016-6-10 16:10

the grid. In this example, the trajectory is converted into a segment label sequence {0,2}.

VII. EXPERIMENTS

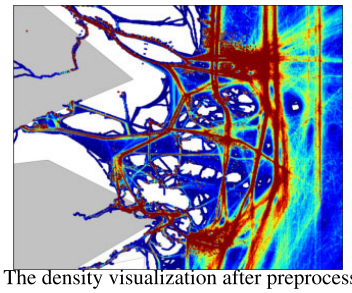
A. DATASET AND EXPERIMENTAL ENVIRONMENT

The data used in this experiment is the AIS data of all cargo ships from June 2016 to July 2016 including China, Malaysia, Singapore, Indonesia and some important ports. AIS data includes vessel's name, call number, MMSI, IMO, ship type, captain, ship width and other static information such as latitude, longitude, direction, speed and status. The experiment uses four columns of MMSI, time, longitude and latitude from the dataset. A dataset sample is shown in Table 2. The total amount of AIS data of global cargo ships collected from June 2016 to July 2016 is over 510G. After pre-processing, the total data volume is over 364G. After GeoHash encoding, the data volume is reduced to about 5.5G.

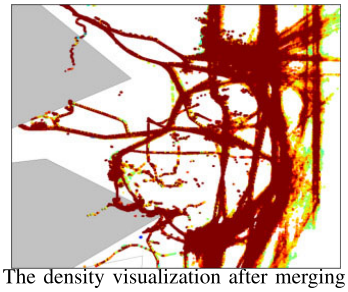
The experiment runs in a cluster of nine server nodes with CDH 5.11 and Spark 1.6.0 environment. The data pre-processing introduced in Section IV is run under Hadoop configuration. The parallel algorithms are run under Spark configuration, and the standalone algorithms are run in a CentOS release 7.0 environment.

B. EXPERIMENTAL RESULTS ON LANE BOUNDARY EXTRACTION

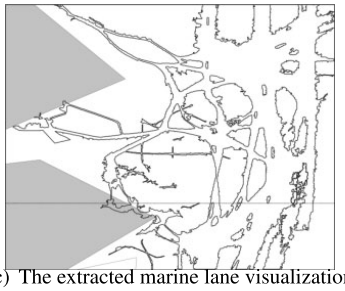
Take the area of East China Sea as an example, Figure 11(a) is a density visualization of the grid data after performing the preprocessing algorithm introduced in Section IV.



(a) The density visualization after preprocessing



(b) The density visualization after merging and filtering



(c) The extracted marine lane visualization

FIGURE 11. Extraction results of the marine lane boundaries in East China Sea.

Figure 11(b) is a density visualization of the grid data result after performing the parallel merging and filtering algorithm. Figure 11(c) is the marine lane boundary after performing the lane boundary extraction algorithm.

We compare our marine lane boundary extraction method proposed in Section V with OTSU based, NiBlack based lane extraction approach, KDE method [28] and DT method [37]. Note that OTSU [38] method and NiBlack method [34] are only image thresholding methods, we changed these methods following the similar idea of this paper and implemented two algorithms based on them, called OTSU-based and NiBlack-based method. We also implemented KDE method and DT method according to [28] and [28].

F_1 score is used to evaluate the experimental results of these five methods performed on the AIS data of five offshore areas near Qingdao, Shanghai, Jakarta, Singapore and Sorong respectively. Qingdao and Shanghai are offshore areas, Jakarta, Singapore and Sorong are open sea areas. As shown in Figure 12, it can be found that F_1 score is the best no matter in offshore or open sea area. So our method can adaptively extract more accurate marine lane results in the offshore and near shore areas simultaneously.

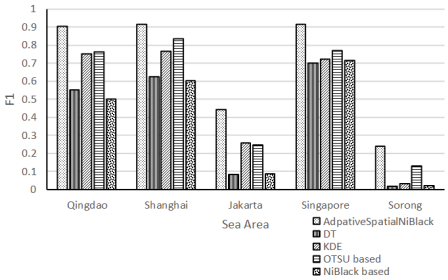


FIGURE 12. F_1 score of the five lane boundary extraction approaches in different sea areas.

C. EXPERIMENTAL RESULTS ON LANE CENTERLINE EXTRACTION AND ROAD NETWORK BUILDING

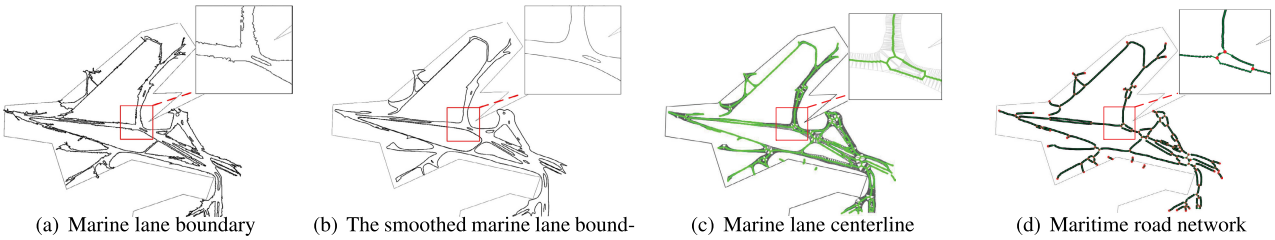
To evaluate the extraction results of maritime road network, we take the area of Bohai as an example. Figure 13(a) is the marine lane boundary visualization extracted by the approach introduced in Section V. Figure 13(b) is the smoothed marine lane boundary visualization after extracting inflection points, deleting lagged edge segments, inserting points, and smoothing boundaries, which is introduced in Section VI-A. Figure 13(c) is the marine lane centerline visualization after performing the lane centerline extraction algorithm. 13(d) is the extracted maritime road network.

As introduced of the related work in Section II, only [31] and [10] extract both boundary and centerline information. The main differences between our MaritimeNET approach and the approach introduced in [31] and [10] are: a). They didn't discuss how to apply their algorithms on massive AIS data for a large area, which is very challenging when the data volume is large, data density and quality are quite uneven; b). They did not propose and build road network with rich information including nodes, segments and segment curves; c). The approach to smooth lane boundaries are different. The smoothing approach introduced in [31] is through area filtering, which may be effective for urban GPS data but is not effective for AIS data analysis.

In order to verify the effectiveness of the proposed approach in this paper, we first compare our approach with Yang's work from the visualization results of the extracted centerlines, and then define some metrics to assess the boundary smoothing effectiveness to quantitatively compare the two approaches.

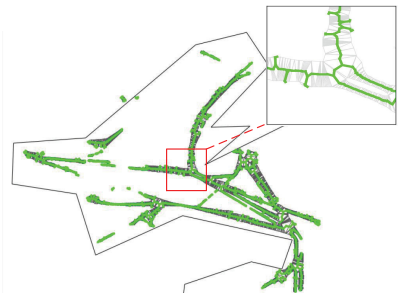
Figure 14(b) shows the centerline visualization of Yang's work and Figure 14(a) shows the result of MaritimeNET approach in this paper. From this figure we can see that Yang's approach based on filtering area filtered some triangles in the lane that should not be filtered, and some triangles outside the lane that should be filtered are not filtered. This results in discontinuous centerlines with more burred branches. In comparison, the centerlines extracted through MaritimeNET approach is continuous and without burrs.

We define two evaluation indicators, namely average curvature ($\bar{C}$) and average sinuosity ($\bar{S}$), to evaluate the smoothness of the extracted centerlines as in Equation 7 and 8,

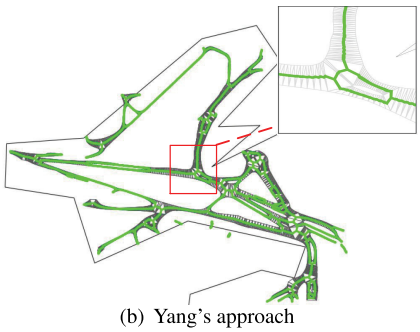


(a) Marine lane boundary (b) The smoothed marine lane bound- (c) Marine lane centerline (d) Maritime road network

FIGURE 13. Extraction results of the maritime road network in Bohai.

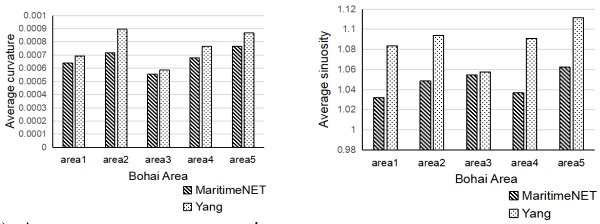


(a) MaritimeNET approach



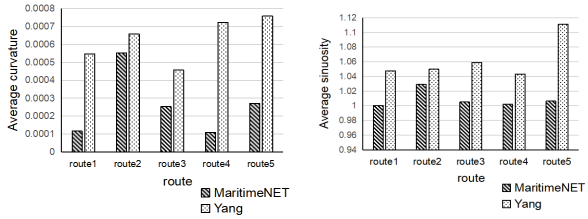
(b) Yang's approach

FIGURE 14. Comparison of the extracted marine lane centerlines in Bohai.



(a) Average curvature comparison (b) Average sinuosity comparison

FIGURE 15. Comparison of $\bar{C}$ and $\bar{S}$ of the two approaches in five different areas of Bohai.



(a) Average curvature comparison (b) Average sinuosity comparison

FIGURE 16. Comparison of the $\bar{C}$ and $\bar{S}$ of the two approaches in five different routes of Bohai.

where n represents the number of centerlines in a road network, m_i represents the number of points of the ith centerline, c_j is curvature of point p_j on a centerline, calculated by the reciprocal of the radius of the circumferential circle of a triangle consisting of three points (p_{j-1}, p_j, p_{j+1}), s_j is the sinuosity of point p_j (defined in Equation 2). The greater the value of $\bar{C}$ or $\bar{S}$, the lower the degree of smoothness.

$$\bar{C} = \frac{1}{n} * \sum_{i=1}^{n} \frac{\sum_{j=1}^{m_i-1} c_j}{m_i - 2} \tag{7}$$

$$\bar{S} = \frac{1}{n} * \sum_{i=1}^{N} \frac{\sum_{j=1}^{m_i-1} s_j}{m_i - 2} \tag{8}$$

We evaluated $\bar{C}$ and $\bar{S}$ of MaritimeNET approach and Yang's approach in randomly selected five areas and five centerlines in Bohai Sea area. The experimental results are shown in Figure 15 and 16, we can see the average curvature and average sinuosity of centerlines extracted by our approach is less than Yang's approach, which means the centerlines extracted using the approach in this paper is smoother.

TABLE 3. Quantitative evaluation with different parameters.

no.	roughness	kurt	WinSize
1	0.447	99.15	–
2	0.088	290.82	3
3	0.056	624.54	5
4	0.047	1069.63	7
5	0.035	648.09	9
6	0.029	800.92	11
7	0.031	1065.29	13
8	0.0385	1750.35	15

Our approach requires a important input parameters *WinSize* introduced in Section VI-A. To determine the value of parameter *WinSize*, two metrics are introduced to assess the quality of smoothing effect, they are *roughness* introduced in Equation 5 and *kurt* introduced in Equation 6. As shown in

the table 3, we can see that the value of *kurt* before smoothing is 99.15, the optimum value of *WinSize* is 11, because the *kurt*(11) is 800.92 larger than the value before smoothing, and the roughness value is the smallest.

VIII. CONCLUSION

We proposed a novel maritime road network extraction approach for large scale area from AIS data. The approach can not only handle the large scale, high noise problem, but also adapt to very uneven density and quality of AIS data in different areas. In addition, our approach is capable of extracting continuous and smooth road network including road boundaries, segments and segment curves, junctions and their relationship. We evaluate the approach on real-world AIS datasets in various areas, achieving effectiveness beyond the most related work.

REFERENCES

[1] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, May 2015, Art. no. 29.

[2] J. Xu, K. Zheng, M.-M. Chi, Y.-Y. Zhu, X.-H. Yu, and X.-F. Zhou, "Trajectory big data: Data, applications and techniques," *J. Commun.*, vol. 36, no. 12, pp. 97–105, 2015.

[3] J. Biagioni and J. Eriksson, "Inferring road maps from global positioning system traces: Survey and comparative evaluation," *Transp. Res. Rec.*, vol. 2291, no. 1, pp. 61–71, 2012.

[4] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk, *Map Construction Algorithms*. Cham, Switzerland: Springer, 2015, pp. 1–14.

[5] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu, "Mining large-scale, sparse GPS traces for map inference: Comparison of approaches," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Aug. 2012, pp. 669–677.

[6] J. Krumm, *Trajectory Analysis for Driving*. New York, NY, USA: Springer, 2011, pp. 213–241.

[7] M. Tavakoli and A. Rosenfeld, "Building and road extraction from aerial photographs," *IEEE Trans. Syst., Man, Cybern.*, vol. 12, no. 1, pp. 84–91, Jan. 1982.

[8] V. Mnih and G. E. Hinton, "Learning to detect roads in high-resolution aerial images," in *Computer Vision*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Germany: Springer, 2010, pp. 210–223.

[9] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, Oct./Dec. 2008.

[10] W. Yang, T. Ai, and W. Lu, "A method for extracting road boundary information from crowdsourcing vehicle GPS trajectories," *Sensors*, vol. 18, no. 4, p. 1261, Apr. 2018.

[11] Z. Wang, T. Ye, M. Lu, X. Yuan, H. Qu, J. Yuan, and Q. Wu, "Visual exploration of sparse traffic trajectory data," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 12, pp. 1813–1822, Dec. 2014.

[12] C. Chen, C. Lu, Q. Huang, Q. Yang, D. Gunopulos, and L. Guibas, "City-scale map creation and updating using GPS collections," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Aug. 2016, pp. 1465–1474.

[13] G. Pallotta, M. Vespe, and K. Bryan, "Vessel pattern knowledge discovery from AIS data: A framework for anomaly detection and route prediction," *Entropy*, vol. 15, no. 6, pp. 2218–2245, 2013.

[14] N. L. Guillarme and X. Lerouvreur, "Unsupervised extraction of knowledge from S-AIS data for maritime situational awareness," in *Proc. 16th Int. Conf. Inf. Fusion*, Jul. 2013, pp. 2025–2032.

[15] V. F. Arguedas, G. Pallotta, and M. Vespe, "Automatic generation of geographical networks for maritime traffic surveillance," in *Proc. 17th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2014, pp. 1–8.

[16] V. F. Arguedas, G. Pallotta, and M. Vespe, "Maritime traffic networks: From historical positioning data to unsupervised maritime traffic monitoring," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 722–732, Mar. 2018.

[17] W. Yan, R. Wen, A. N. Zhang, and D. Yang, "Vessel movement analysis and pattern discovery using density-based clustering approach," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2016, pp. 3798–3806.

[18] A. Dobrkovic, M.-E. Iacob, and J. van Hillegersberg, "Maritime pattern extraction and route reconstruction from incomplete AIS data," *Int. J. Data Sci. Anal.*, vol. 5, nos. 2–3, pp. 111–136, Mar. 2018.

[19] G. Agamennoni, J. I. Nieto, and E. M. Nebot, "Robust inference of principal road paths for intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 1, pp. 298–308, Mar. 2011.

[20] C.-C. Hung, W.-C. Peng, W.-C. Lee, C. S. Jensen, and H. T. Shen, "Clustering and aggregating clues of trajectories for mining trajectory patterns and routes," *VLDB J.*, vol. 24, no. 2, pp. 169–192, 2015.

[21] G. Spiliopoulos, D. Zissis, and K. Chatzikokolakis, "A big data driven approach to extracting global trade patterns," in *Mobility Analytics for Spatio-Temporal and Social Data*, C. Doulkeridis, G. A. Vouros, Q. Qu, and S. Wang, Eds. Cham, Switzerland: Springer, 2018, pp. 109–121.

[22] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: A partition-and-group framework," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*. New York, NY, USA: ACM, Jun. 2007, pp. 593–604.

[23] E. Naserian, X. Wang, K. Dahal, Z. Wang, and Z. Wang, "Personalized location prediction for group travellers from spatial–temporal trajectories," *Future Gener. Comput. Syst.*, vol. 83, pp. 278–292, Jun. 2018.

[24] R. Wen, W. Yan, A. N. Zhang, N. Q. Chinh, and O. Akcan, "Spatio-temporal route mining and visualization for busy waterways," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 849–854.

[25] L. Etienne, T. Devogele, and A. Bouju, "Spatio-temporal trajectory analysis of mobile objects following the same itinerary," *Adv. Geo-Spatial Inf. Sci.*, vol. 10, pp. 47–57, Jul. 2012.

[26] L. Tang, C. Ren, Z. Liu, and Q. Li, "A road map refinement method using Delaunay triangulation for big trace data," *ISPRS Int. J. Geo-Inf.*, vol. 6, no. 2, p. 45, 2017.

[27] C. Chen and Y. Cheng, "Roads digital map generation with multi-track GPS data," in *Proc. Int. Workshop Educ. Technol. Training Int. Workshop Geosci. Remote Sens.*, Shanghai, China, vol. 1, Dec. 2008, pp. 508–511.

[28] W. Shi, S. Shen, and Y. Liu, "Automatic generation of road network map from massive GPS, vehicle trajectories," in *Proc. 12th Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2009, pp. 1–6.

[29] W. Yang and T. Ai, "The extraction of road boundary from crowdsourcing trajectory using constrained delaunay triangulation," *Acta Geodaetica et Cartographica Sinica*, vol. 46, no. 2, pp. 237–245, Feb. 2017.

[30] J. Li, W. Chen, M. Li, K. Zhang, and Y. Liu, "The algorithm of ship rule path extraction based on the grid heat value," *J. Comput. Res. Develop.*, vol. 55, no. 5, pp. 908–919, 2018.

[31] Y. Wei and A. Ting-Hua, "Road centerline extraction from crowdsourcing trajectory data," *Geography Geo-Inf. Sci.*, vol. 32, no. 3, pp. 1–7, 2016.

[32] Wikipedia. (2018). *Geohash*. Accessed: Dec. 9, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Geohash

[33] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.* Berkeley, CA, USA: USENIX, 2010, Art. no. 10.

[34] W. Niblack, *An Introduction to Digital Image Processing*. Birkerød, Denmark: Strandberg, 1985.

[35] Z. Li, G. Wang, J. Meng, and Y. Xu, "The parallel and precision adaptive method of marine lane extraction based on QuadTree," in *Collaborative Computing: Networking, Applications and Worksharing*, H. Gao, X. Wang, Y. Yin, and M. Iqbal, Eds. Cham, Switzerland: Springer, 2018, pp. 170–188.

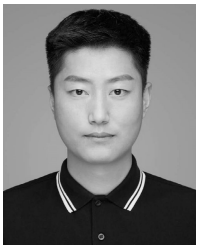
[36] K. Rong and P. Bailis, "ASAP: Prioritizing attention via time series smoothing," *Proc. VLDB Endowment*, vol. 10, no. 11, pp. 1358–1369, Aug. 2017.

[37] T. Ai and W. Yang, "The detection of transport land-use data using crowdsourcing taxi trajectory," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. XLI-B8, pp. 785–788, Jul. 2016.

[38] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.

GUILING WANG was born in Shandong, China, in 1978. She received the B.S. and M.S. degrees from the Nanjing University of Posts and Telecommunications, China, in 1999 and 2002, respectively, and the Ph.D. degree from Tsinghua University, China, in 2007, all in computer science.

From 2007 to 2009, she was a Postdoctoral with the Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS). From 2009 to 2012, she was an Assistant Professor with ICT, CAS. Since 2012, she has been an Associate Professor with the Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, North China University of Technology, Beijing. From 2017 to 2018, she was a Guest Researcher with Ocean Information Technology Company, China Electronics Technology Group Corporation. From 2018 to 2019, she was also a Visiting Research Scientist with Duisburg-Essen University, Germany. Her research interests include data integration, services composition and mashup technologies, trajectory computing, and large-scale streaming data integration and processing.



JINLONG MENG was born in Gansu, China, in 1994. He received the B.S. degree in information security from the North China University of Technology, China, in 2017, where he is currently pursuing the M.S. degree in computer technology, under the supervision of Dr. Wang. His main research interests include trajectory computing and large-scale streaming data processing.

YANBO HAN was born in Chengde, Hebei, China, in 1962. He received the Ph.D. degree in computer science from the Technical University of Berlin, Germany, in 1996.

Since 2000, he has been a Full Professor in computer science with the Institute of Computing Technology, Chinese Academy of Sciences, till 2012, and currently with the North China University of Technology, Beijing. He is also the Director of Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, North China University of Technology. He has authored or coauthored over 160 papers and five books. His team has acquired over 70 intellectual properties and 15 of them have been transferred to the industry. His current research interests include streaming data processing, cloud computing, dependable distributed systems, and business process collaboration and management. He has undertaken over 40 funded research projects and served as the PI of the China Grid Initiative. Dr. Han has organized over 30 academic events as General Chairs or Program Chairs, including 12 journal special issues.