LiDAR Based Navigable Region Detection for Unmanned Surface Vehicles

Xiangtong Yao[1], Yunxiao Shan[1], Jieling Li[1], Donghui Ma[1], Kai Huang[1,2,*]

Abstract— **Detection of the navigable regions for the unmanned surface vehicles (USVs) sailing on the narrow rivers is very important. Existing detection methods mostly depend on the cameras, which is sensitive to environments and cannot provide reliable navigable regions for sailing. In this paper, we propose a scheme to process 3D LiDAR data to achieve an accurate and robust navigable regions detection. We conduct field experiments in a narrow river in different scenarios to prove the performance of the proposed scheme, which reaches on average 93.8% precision and 92.7% recall.**

Keywords: water surface extraction, deep learning, navigable region detection USVs

I. Introduction

Unmanned surface vehicles (USVs) are widely deployed for various open water missions like mapping environment[1], military tasks[2], environment monitoring[3], etc. A certain degree of autonomy can help to complete the missions efficiently and intelligently. To enable autonomy, detecting the navigable area of the water surface is a crucial step as it provides necessary information not only for obstacle avoidance, but also for later-on path planning and automated exploration.

Navigable regions for different kinds of water surfaces have different requirements, for the case of open water, e.g., oceans and large rivers, the navigable regions are not too strict and the detection is relatively easy. For the case of narrow rivers or canals, the river banks in principle define the navigable areas. Beside the banks, irregular bridges with holes will adjust the navigable regions, which will further complicate the detection of the navigable regions.

Existed methods for navigable regions detection in narrow rivers use the cameras mounted on the unmanned flying[4] or surface vehicle[2] to detect the navigable water surface. The reasons for choosing stereo systems are their relatively cheap prices and rich semantic information. A major limitation of stereo systems is their weakness in handling changes in illumination. The reflection of water will further exacerbate the illumination variation. In addition, stereo cameras are sensitive to calibration errors.

In recent developments of autonomous automobiles, LiDAR emerges as the major sensor for environment perception. Compared to cameras, LiDAR is less sensitive to light conditions and can work under poor illumination conditions. Applying LiDAR for navigable regions detection of water surface is however not straightforward. First of all, rivers surface are not as reflective as roads surface and most of the LiDAR signals will be absorbed. Second, the unabsorbed signals may be reflected by the rough river-bed of shadow water, which results in uncertainties in the characteristics of the LiDAR signals on the water. Third, ripples and small floating objects on the river surface, e.g., leaves and dead fish bodies, may also reflect LiDAR signals.

Being aware of the aforementioned challenges, this paper presents a navigable regions detecting scheme using 3D LiDAR point clouds. Our scheme consists of CNN for semantic segmentation, polygon fitting for surface extraction, and particle filter for noise canceling. The point clouds are firstly segmented by a customized deep learning network into three classes, the bridges, river banks, and plants on the bank. With the segmented river banks, an improved polygon fitting method is employed to extract the water surface. Afterwards, the particle filter is applied to filter out the noises on the water surface to detect an accurate and robust navigable region. Experimental results show that our algorithm can reach on average 93.8% precision and 92.7% recall rate. The main contributions of this paper are as follows:

- A customized deep learning method for 3D LiDAR point clouds is proposed to achieve semantic segmentation, then classify the objects into bridges, river banks, and plants on the banks.
- An improved convex polygon fitting method is designed to extract the water surface roughly, and particle filter is used to filter out the noise point clouds on the water surface to obtain the fine navigable regions.
- We conduct experiments on a narrow river with different types of bridges and small floating objects and compare our results with manually-measured ground truth in different river bank scenes.

The organization of this paper is as follows: Section II reviews related work in the literature. Section III intro-

* Corresponding author.

[1] Xiangtong Yao, Jieling Li, Donghui Ma, Yunxiao Shan, Kai Huang are with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, School of Data and Computer Science Sun Yat-sen University, Guangzhou, China. (email:{yaoxt3,lijling7,madh5}@mail2.sysu.edu.cn, {shanyx,huangk36}@mail.sysu.edu.cn)

[2] Kai Huang is with Sun Yat-sen University, Shenzhen Institute.

duces the background knowledge. Section IV presents our approach. Experimental results are provided in Section V. Section VI concludes the paper.

II. Related Work

There are only a few researches targeting water surface. Most of them are camera based. Gong et al.[5] employed multiple features from images to build a two-stage algorithm to detect the shoreline. Nevertheless, the algorithm is only validated in an ideal water surface, without objects and ripples. Achar et al.[6] proposed a self-supervised method to detect the river surface with higher accuracy than supervised alternative. Moreover, it saved the retraining process when working under different conditions. This method cannot learn an appearance model from previous images so that the novel objects below the horizon cannot be detected. Jain et al.[4] fused the vision and LiDAR in a flying vehicle to map the river. In their mapping system, camera was employed to detect the water surface and LiDAR was used to construct an environment map. In their work, while LiDAR is applied, it is not used to further improve the detection quality of the navigable water surface.

For the case of open water detecting applications with LiDAR, Han et al.[7] used LiDAR to detect huge targets, e.g. bridges, waterside buildings, towers, and cranes. The Gaussian-means (G-means) algorithm was used to extract the feature, and then the inscribed angle variance (IAV) was employed for segmentation. Nevertheless, the prior known models of target objects are required to achieve accurate detections. Tompson et al.[8] proposed an object detection method for the ocean application with LiDAR. Support Vector Machine (SVM) was used to classify the objects on the sea surface, nonetheless, SVM can only be applied to the small-scale data. For the case of the narrow water, Terry et al.[2] segmented the water surface by the characteristics that the water surface absorbs the wave. This method works for static, deep and no-floats water regions, but fails when there are ripples, shadow water regions or small floating objects because of the possible random reflections. For the navigable regions detection in the dynamic and complex narrow rivers, it is not possible to know the prior models of the bank. Moreover, random noise point clouds are needed to be filtered out to guarantee an accurate detection. Therefore, a model-free and accurate detection method is eagerly required for safe autonomous sailing.

III. Background

A. Deep Learning in Semantic Segmentation

Semantic segmentation of image processing refers to assigning each pixel to the corresponding object class. The development of deep learning improves the accuracy and efficiency of image semantic segmentation, such as Fully Convolutional Networks (FCN), DeepLab, U-Net, etc. FCN performs multiple convolution and deconvolution operations on the input image, and finally predicts the object class each pixel belongs to.

B. Convex Hull and Graham Scan Algorithm

Given a point set P of n points, the convex hull problem is to use the smallest convex polygon to contain all the points in P. In Euclidean plane E^2, a famous algorithm known as *Graham Scan Algorithm(GSA)* can solve the convex hull problem efficiently[9]. Moreover, the GSA outputs a set of ordered points in either counterclockwise or clockwise order, which are connected in turn to obtain the smallest convex polygon.

C. Particle Filter

Particle filter is a good choice for solving multi-objective tracking problems with highly non-linear and non-Gaussian motion models. The vector x_t is used to indicate the state of a tracked object, and the vector Y_t denotes all the observations $\{y_1,y_2,...,y_{t-1},y_t\}$ up to time t. The aim of particle filter is to estimate the posterior density $p(x_t|Y_t)$ with given Y_t, by the two-step recursion[10]:

$$Predict : p(x_t|Y_{t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|Y_{t-1})dx_{t-1} \tag{1}$$

$$Update : p(x_t|Y_t) \propto p(y_t|x_t)p(x_t|Y_{t-1}) \tag{2}$$

It is quite difficult to solve the above equation when $p(x_t|Y_t)$ and $p(y_t|x_t)$ are non-Gaussian, therefore particle filter approximates the probability distribution by many weighted particles $s^{(i)}$ $(i = 1...N)$, each weighted particle represents one hypothetical state of the tracked object, the weight $w^{(i)}$ is calculated in terms of the observation vector Y_t, where $\sum_{n=1}^{N} w^{(n)} = 1$. The particle set $S = \{s^{(i)}|i = 1...N\}$ is re-sampled according to the weight of the particle, and the less weight particles are replaced by a particular particle with probability $w^{(i)} = p(y_t|x_t = s_t^i)$. After that the mean state of the tracked object is approximated by

$$x' = \sum_{n=1}^{N} w^{(n)} s^{(n)} \tag{3}$$

IV. Our Approach

Our proposed framework is composed of river bank segmentation, water surface extraction, and navigable regions detection, shown in Fig.2. In the framework, a customized deep learning method based on LiSeg[11] is applied to classify point clouds into three classes, i.e., brides, river banks, and plants. Secondly, a geometric model is proposed to extract the water surface, and then the particle filter is employed to filter out the noises on the extracted water surface to improve the detection precision of the navigable regions.

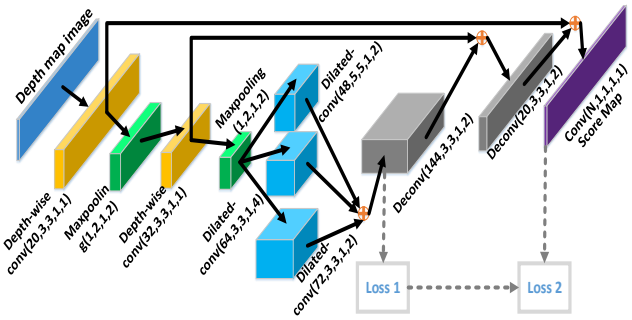


Fig. 1. LiSeg Structure. Depth-wise separable convolution layer is shown in orange. Max-pooling layer is in dark green. Dilated is shown in blue. Transpose convolution is shown in grey. Semantic segmentation result is shown in dark purple.

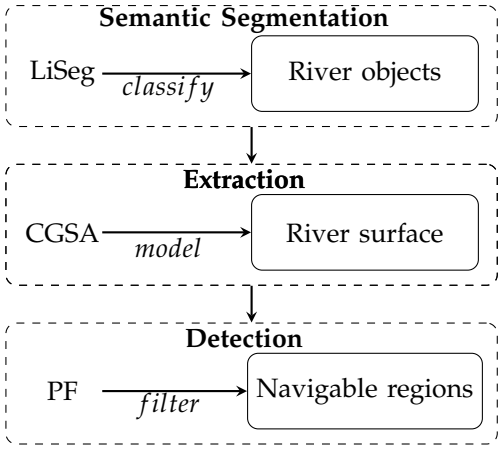


Fig. 2. Framework of river drivable area extraction

A. River bank segmentation

LiSeg is a lightweight FCN, the structure of LiSeg is shown in Fig.1. The depth-wise separable convolution layer is shown in orange. The max-pooling layer is shown in dark green. The dilated convolution layer is shown in blue. The transpose convolution layer is shown in grey. The convolution is shown in dark purple. The convolution operators are parameterized as $(T, \alpha 1, \alpha 2, \theta 1, \theta 2)$ and the max-pooling operators are parameterized as $(\alpha 1, \alpha 2, \theta 1, \theta 2)$, where T is the number of filters, $(\alpha 1, \alpha 2)$ is the kernel size respectively, and $(\theta 1, \theta 2)$ is the strides respectively, and the dilated convolution operators are parameterized as $(T, \alpha 1, \alpha 2, \delta 1, \delta 2)$, where $(\delta 1, \delta 2)$ stands for the dilated rate. Depth-wise, separable convolution is employed to decrease the number of convolution parameters in LiSeg. Moreover, two max-pooling layers are used in the early stage to reduce the number of parameters of feature maps to improve running time, and the dilated convolution is used to enhance the receptive field.

LiSeg is designed to segment the road-objects, whose characteristics are very different from the river scenarios as mentioned before. Therefore, the weight of the multi-layer loss function need to be adjusted in LiSeg to perform better on river scenarios.

The whole LiSeg network is trained via end-to-end back-propagation guided by the following weighted multi-layer loss function:

$$\mathcal{L} = \sum_{i=1}^{2} \lambda_i \mathcal{L}oss_i{}^{WCE}(\hat{\mathcal{X}}_i, \mathcal{X}_i) \tag{4}$$

The weighted multi-layer loss $\mathcal{L}$ shown in Eq.(4) is evaluated on two feature maps respectively, regarded as $\mathcal{L}oss_1$ and $\mathcal{L}oss_2$, as shown in Fig.1. The parameter λ_i is the corresponding regularization weights. $\hat{\mathcal{X}}$ and $\mathcal{X}$ are the predictions and ground truth respectively.

The number of points in the LiDAR point cloud is class-imbalance, which will decrease the accuracy of segmentation, more than 58% points belong to plants, and about 25% point clouds belong to the river bank. Thus the following multi-class Weighted Cross Entropy(WCE) loss is applied as regularization to alleviate the influence of class-imbalance problem. The corresponding weights of class-imbalance regularization are computed according to the training set statistics.

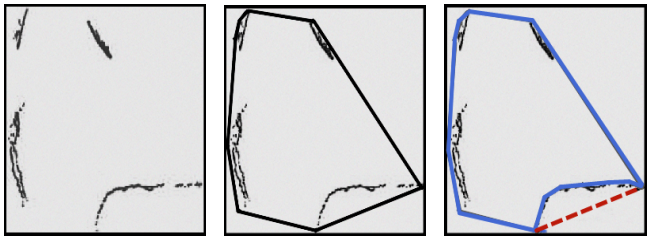
$$\mathcal{L}oss_i^{WCE} = -\sum_{j,k,l}^{H_i,W_i,N} \mathcal{W}(\mathcal{X}_{j,k}) \mathcal{ID}(\mathcal{X}_{j,k}) \log(\hat{\mathcal{X}}_{j,k,l}) \tag{5}$$

In Eq.(5), $\mathcal{ID}(\mathcal{X}_{j,k})$ is an index function to select the expected classes and $\mathcal{W}(\mathcal{X}_{j,k})$ corresponds to the class-imbalance weights.

B. Water surface extraction

The first step to detect the navigable regions is to model the segmented point clouds into geometric shapes. A naive method is to use the minimum convex polygon (MCP). While it can represent the regular banks accurately, the modeling errors at the curved banks are very large. In the curved river banks, the MCP may contain the regions outside the water surface. Therefore, the MCP must be modified to accommodate our application.

A modeling algorithm based on the MCP fitting method is proposed, named customized Graham Scan algorithm (CGSA), as shown in Alg.1. In the algorithm, the Graham Scan algorithm (GSA) method is employed to calculate MCP. With the *Polygon*, all the points are grouped in Q' according to the shortest distance from the point to the edge in *Polygon* and then attach them with the corresponding edge (line 3). A method of fitting error reductions is proposed to decrease the fitting errors caused by curved river bends, as described in *Function* **Cal_Poly**. For each edge e_i in *Polygon*, the deviation is decided by the fitting error $F\_error(e^i)$ of the attached points $Q'(e^i)$. If an over-threshold edge is found, a recursion algorithm will be triggered to decrease the fitting errors of each edge in the polygon (line 5 to 9). The basic idea of the recursion is that the



(a) River bank point clouds. (b) Apply MCP to get an initial polygon. (c) The final result of Alg.1

Fig. 3. Explanation of the Alg.1.

over-threshold edge will be replaced by MCP recursively until all the edges are smaller than T. In the newly calculated polygon, similar edges with the previous over-threshold edge will be deleted (line 7 to 8). Fig.3 explains the whole process, from calculating the initial basic polygon(Fig.3(a-b)) to deforming the polygon to a more accurate representation of the water surface model(Fig.3(b-c)). Fig.3(a) shows the river bank point clouds. Fig.3(b) represents MCP is applied to get an initial polygon with black edges and Fig.3(c) represents MCP is re-applied to the point clouds attached to the over-threshold edge(dashed red line) to get an more accurate polygon model with blue edges.

Algorithm 1 Water Surface Extraction

Input: The river banks 3D LiDAR point clouds Q
Output: The water surface model $G\_Polygon$

1: **function** $Cal\_Poly(Polygon)$
2: **for** Edge $e^i \in Polygon$ **do**
3: $Q'(e^i) \Leftarrow$ point clouds attached to e^i;
4: $F\_error(e^i) \Leftarrow Cal\_fitting\_error(e^i, Q'(e^i))$;
5: **if** $F\_error(e^i) > T$ **then**
6: $Polygon \Leftarrow GSA(Q'(e^i))$;
7: $new\_e^i \Leftarrow$ **search_e_i**$(Polygon)$;
8: **Delete** $new_e^i \in Polygon$;
9: $Cal\_Poly(Polygon, e^i)$;
10: **else**
11: $G\_Polygon \Leftarrow G\_Polygon \bigcup e^i$;
12: **end if**
13: **end for**
14: **end function**
15: Project Q onto the Euclidean plane $E^2 \rightarrow Q'$;
16: $Polygon \Leftarrow GSA(Q')$;
17: $Cal\_Poly(Polygon)$;

C. Navigable regions Detection

While the river surface is precisely separated from the banks by the proposed method above, it may still not be navigable due to the possible existence of fixed obstacles in the river, e.g., bridge piers, uncovered stones, and other static objects. Moreover, the appearances of fixed obstacles in point clouds are very similar to noises, e.g., ripples and floating objects. Therefore, it is a challenging task to discriminate the obstacles from the noises.

In the river application, the noise point clouds are so scattered that it is impossible to maintain continuous moves to keep a stable appearance in a series of consecutive frames. To cope with this problem, particle filter (PF) is applied on all the point clouds in $Polygon$ to extract the fixed obstacles and get rid of the noises. Specifically, N particles of the point clouds are initialized in the first frame at time t, and the observation density $p(y_{t'}|x_{t'})$ of these particles is evaluated in subsequent frames. With $p(y_{t'}|x_{t'})$, the posterior density $p(x_{t'}|Y_{t'})$ of the point is calculated by Eq.(2). In our application, a point will be removed from the $Polygon$ if the value of $p(x_{t'}|Y_{t'})$ is small enough.

In the proposed PF, the weight $w^{(i)}$ of the particle s^i will affect the final prediction results. According to Eq.(6), $w^{(i)}$ depends on the observation value $y_t^{(i)}$. Therefore, the method to calculate the observation vector Y_t is crucial for the prediction results. In our case, the observation space is set to a sphere with a radius of r centered on a particle. The observation value $R(x)$ is represented by the number of points in the sphere centered at x, and $w^{(i)}$ is calculated by Eq.(6).

$$w_t^{(i)} = p(y_t|x_t = s_t^{(i)}) = \begin{cases} \frac{R(x_{t-1})}{|R(x_{t-1}) - R(s_t^{(i)})| + \xi} & , R(s_t^{(i)}) > 0 \\ 0 & , R(s_t^{(i)}) = 0 \end{cases} \quad (6)$$

where $\xi = 0.5$. In Eq.(6), the weight of the ith particle ($w^{(i)}$) will be adjusted according to the difference between the observations of particle and point.

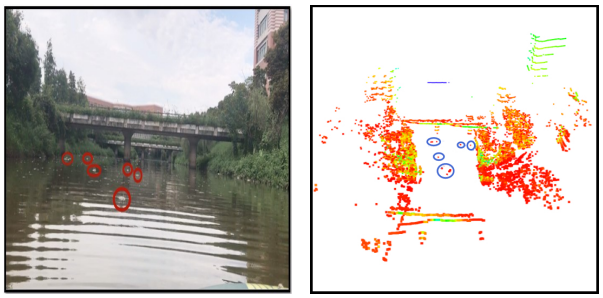
V. Experiments

A. Experiment setup

Currently, there are no open-source benchmarks of LiDAR data for sailing. Therefore, we collected LiDAR data in a narrow river using the Velodyne VLP-16 and camera. Fig.4 shows a sampling scenario, in which the dead fish bodies and other floating objects are scattered. For illustration purpose, we use the LOAM (Laser Odometry and Mapping) to process the raw LiDAR data to construct a LiDAR panorama of the whole river, shown in Fig.5(b). In the illustrated scene, plants, bridges, banks, and floats are all displayed. Besides, according to the curvature of river banks, we split the whole test scenario into two parts, AB in Fig.5 represents the river banks with smooth curvature (smooth banks) and BC in Fig.5 represents river banks with sharp curvature changes (sharp banks).

B. Setup of Semantic Segmentation Network

Parameters of Network: All the models were implemented based on Tensorflow Estimator API on Nvidia



(a) Floats in image (b) Floats in point clouds

Fig. 4. sample scenario

GTX1070. We set the max number of the training epoch to 70000. The model's learning rate, batch size, and fold for cross-validation are 0.001, 8, 5, respectively.

Point Cloud Data: We mainly annotated three kinds of common objects: river banks, plants, and bridges. The training data consists of 450 frames point clouds data, which contain 13.5 million points in total, annotated at point-wise level with the help of annotation tools. In our dataset, the point clouds of the river banks, plants and bridges accounted for 25.2%, 58.3%, and 16.5%, respectively.

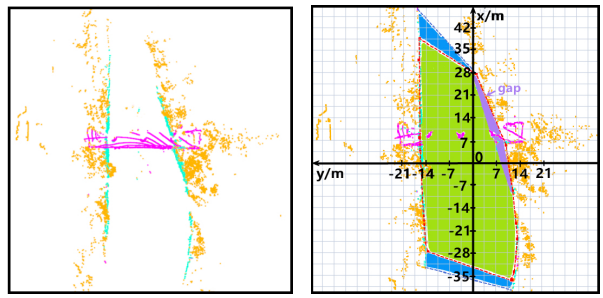
C. Evaluation of Semantic Segmentation Network

The accuracy of customized semantic segmentation model was evaluated on two common indexes: Mean Intersection over Union (mIoU) and Mean Pixel Accuracy (mPA). The formulas to calculate these indexes were shown in Eq.(7). The k denoted the number of classes, and p_{ij} indicated the number of the points (p) in class i but predicted to class j.

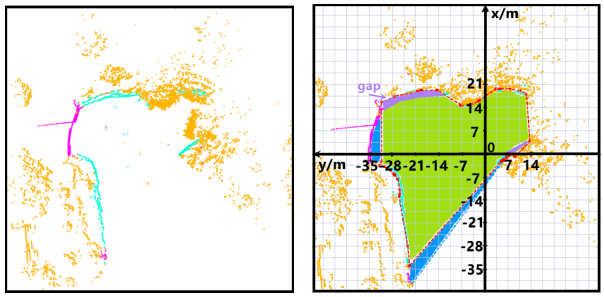
$$\begin{aligned} mIoU &= \frac{1}{k+1}\sum_{i=0}^{k}\frac{p_{ii}}{\sum_{j=0}^{k}p_{ij}+\sum_{j=0}^{k}p_{ji}-p_{ii}} \\ mPA &= \frac{1}{k+1}\sum_{i=0}^{k}\frac{p_{ii}}{\sum_{j=0}^{k}p_{ij}} \end{aligned} \tag{7}$$

D. Experiment results

In the river scene, the average mIoU and mPA of the customized LiSeg are 89.3% and 93.6%, respectively. The comparison results of precision and recall along the whole river in Fig.5 demonstrates the effectiveness of the proposed scheme. The customized LiSeg is applied to segment the scene, and then the detection performance of navigable regions is improved by applying the CGSA and PF. Firstly, we report overall precision and recall along the chosen river section. In Fig.5.(a), the first observation is that the precision and recall of section AB are 95.8% and 94.1%, respectively, and the precision and recall of section BC are 91.8% and 91.4%, respectively. The precision and recall of section BC are lower than section AB. The reason is that the curvature of river bank in section BC is larger than section AB,



(a) Semantic segmentation result of smooth bank (b) Water surface model of smooth bank



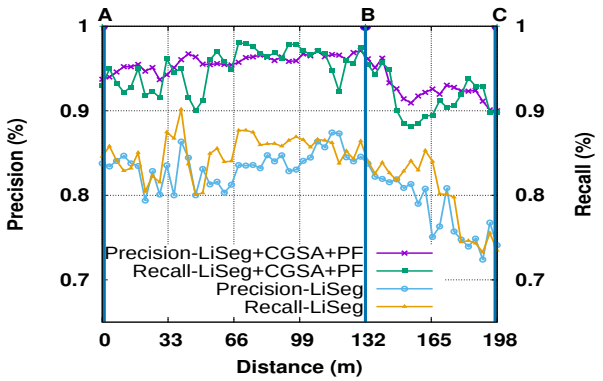
(c) Semantic segmentation result of sharp bank (d) Water surface model of sharp bank

Fig. 6. Navigable region detection results. In the segmented results, The bridge is shown in magenta, the river bank is shown in cyan and the plants is shown in yellow. In the detection results, the navigable region is marked with light green, the undetected region is blue, and the gap is light purple. The final result is placed in the coordinate system with grids. The origin is located at the location of LiDAR and the positive direction of X-axis is the front of USV.

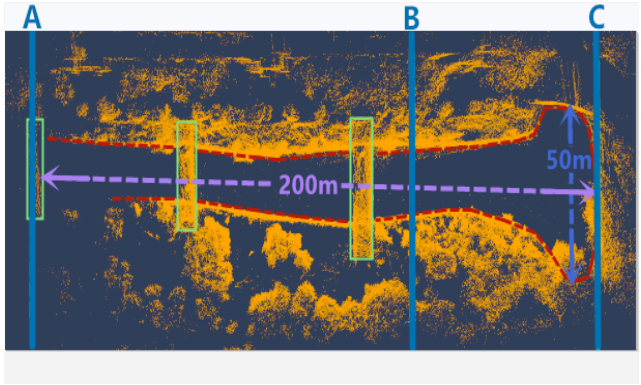
and thus the fitting error of section BC is bigger than section AB. The second observation is that the precision and recall of the algorithm without CGSA and PF are 10% lower than that of the algorithm with CGSA and PF. The reason is that the noises caused by the floats mistake the navigable regions as unnavigable areas.

We also analyze in details the result of the navigable region detection. Two specific scenes are chosen for the analysis. The detection results of a section of the smooth banks and sharp banks are shown in Fig.6. According to the segmentation results in Fig.6, both of the scenarios are all accurately segmented. The segmented point clouds of river banks are modeled by the proposed CGSA in Fig.6(b) and Fig.6(d). For the Fig.6(b) and Fig.6(d), we can find out that there are a large gap between the water surface and river banks in the sharp bank scenarios, especially in areas with the sharper banks. A reasonable explanation is that the gap depends on the selection of the parameter T in Alg.1. Larger T speeds up the extracting efficiency and smaller T may cost more time to refine the polygon. We make a trade-off to improve the overall performance of our scheme. Moreover, large gaps observed in the distance are due to the scarcity of laser points.

Finally, we show the effects of the customized PF and CGSA in Fig.7 separately. Fig.7(a) shows that CGSA has little effect on the detection of the smooth banks.

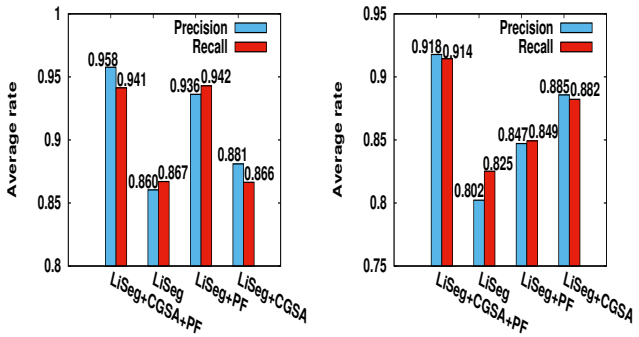


(a) Precision and recall along the river bank



(b) LiDAR panorama. Red dotted lines represent river bank, green rectangles represent bridges, and remaining yellow point clouds represent plants.

Fig. 5. LiDAR panorama and the corresponding accuracy of navigable regions detection



(a) Section *AB* (b) Section *BC*

Fig. 7. Performance of four different combinations of algorithms

In contrast, customized PF significantly improves the detection accuracy of the navigable area. The reason for that is the different curvatures of the river banks. The curvature of section *AB* is relatively small, and therefore the effects of CGSA are not obvious, but the noises have a huge influence on the precision and recall, and thus PF plays a more important role in section *AB*. Moreover, Fig.7(b) proves that the CGSA is an effective method to tackle the problem of large fitting errors in the curved banks. The polygon with large fitting-errors will be replaced by a more compact polygon with the acceptable fitting errors. Besides, the noises also need to be filtered out in the curved banks.

VI. Conclusion

In this paper, we propose a 3D LiDAR-based scheme to achieve an accurate and robust navigable regions detection. Our scheme mainly 1) uses a customized deep learning network to segment LiDAR point clouds into bridges, river banks, and plants; 2) designs an improved minimum convex polygon fitting method to extract the water surface; 3) and finally extracts the accurate and robust navigable regions by applying the particle filter to filter out the noise on the water surface. Experiments were carried out in a narrow river, and the experiment results demonstrate that our proposed scheme achieves a state-of-the-art performance.

VII. Acknowledgements

The work described in this paper was supported in part by the grant from No.JCYJ20180508152434975, in part by the grant from No.2018B010108004, and in part by Guandong "Climbing Program" special funds under Grant No.pdjh2018b0012.

References

[1] J. C. Leedekerken, M. F. Fallon, and J. J. Leonard, "Mapping complex marine environments with autonomous surface craft," in *Experimental Robotics*. Springer, 2014, pp. 525–539.

[2] T. Huntsberger, H. Aghazarian, A. Howard, and D. C. Trotz, "Stereo vision ndash;based navigation for autonomous surface vessels," *Journal of Field Robotics*, vol. 28, no. 1, pp. 3–18, 2011.

[3] M. Dunbabin, A. Grinham, and J. Udy, "An autonomous surface vehicle for water quality monitoring," 2009.

[4] S. Jain, S. Nuske, A. Chambers, L. Yoder, H. Cover, L. Chamberlain, S. Scherer, and S. Singh, "Autonomous river exploration," in *International Conference on Field and Service Robotics*, 2014, pp. 93–106.

[5] X. Gong, A. Subramanian, and C. L. Wyatt, "A two-stage algorithm for shoreline detection," in *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*, 2007, pp. 40–40.

[6] S. Achar, B. Sankaran, S. Nuske, and S. Scherer, "Self-supervised segmentation of river scenes," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 6227–6232.

[7] J. Han, J. Park, T. Kim, and J. Kim, "Precision navigation and mapping under bridges with an unmanned surface vehicle," *Autonomous Robots*, vol. 38, no. 4, pp. 349–362, 2015.

[8] D. J. Thompson, "Maritime object detection, tracking, and classification using lidar and vision-based sensor fusion," 2017.

[9] R. L. Graham, "An efficient algorith for determining the convex hull of a finite planar set," *Information Processing Letters*, vol. 1, no. 4, pp. 132–133, 1972.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[11] W. Zhang, C. Zhou, J. Yang, and K. Huang, "Liseg: Lightweight road-object semantic segmentation in 3d lidar scans for autonomous driving," in *IEEE Intelligent Vehicles Symposium*, 2018, pp. 40–40.