Proc. European Conf. on Mobile Robots 2021 (ECMR2021), Aug. 31-Sep. 3, 2021.

3D Road Boundary Estimation using 3D LiDAR with Scanline-wise 1D Deep Feature and Particle Filtering

Yuta Nakayama[1] and Jun Miura[1]

Abstract— **Recognizing road shape is one of the fundamental functions for outdoor navigation of mobile robots and vehicles. This function is crucial for safe control and used for autonomous navigation combined with global localization using maps or GNSS. This paper describes a method of estimating the 3D structure of road boundaries using a 3D LiDAR with a combination of scanline-wise 1D feature extraction and temporal filtering by particle filter. In outdoor environments, since the road shape changes not on a horizontal plane but three-dimensionally, we model the road boundary shape with a series of 3D segments and estimate its parameters repeatedly with the feature extraction and particle filter. The proposed method is tested in terms of the feature extraction performance and the applicability of autonomous navigation.**

I. Introduction

Autonomous navigation is one of the active research topics in robotics. There are many application areas from autonomous driving [1], [2] to delivery robots [3]. A common pipeline for realizing autonomous navigation is a combination of sensor-based road recognition and road following control [4], [5]. Although several attempts have been made for end-to-end driving [6], [7], the pipeline currently seems the most practical.

Road region recognition has been tackled for a long time. Vision-based systems [8], [9], [10] have been developed, recently with the advancement of deep learning technologies and large-sized datasets like KITTI [11]. The use of LiDAR (Light Detection and Ranging) is also popular for reducing the effect of lighting conditions [12], [13].

In these approaches, the recognition results, especially road boundary locations, tend to be noisy or sometimes wholly wrong. Therefore the direct application of the recognition results to robot control is occasionally risky. A temporal smoothing of control commands may address this issue, but it is more desirable to address it in the recognition phase.

To cope with such an occasional recognition failure, we have developed boundary estimation methods with a combination of image processing and temporal filtering [14], [15], [16]. In these works, we adopt a boundary continuity constraint through a predefined but flexible boundary model, thereby naturally handling occasional failures. This paper extends such an approach in the following two points: use of LiDAR and estimation of 3D shape of road boundaries. The contributions of the paper are as follows:

- We develop a lightweight 1D feature extraction for localizing road boundary points in each scanline.
- We formulate the 3D road boundary model and its updating process.
- We realized autonomous navigation on a non-planar path with 3D trajectory reconstruction.

[1]The authors are with Department of Computer Science and Engineering, Toyohashi University of Technology, Japan

II. Related Work

A. Visual road recognition

Borkar et al. [8] developed a lane detection method that converts input images into the bird's eye view ones and detects boundary lines using RANSAC. Danescu and Nedevschi [9] developed a method of detecting lane boundaries using stereo vision and particle filtering. Current visual systems frequently use semantic segmentation methods such as [17]. Teichmann et al. [10] conducted road region extraction using a multi-task learning approach.

B. LiDAR-based road recognition

LiDARs have also been popular sensors for autonomous driving [18]. Sun et al. [13] developed a 3D LiDAR-based road boundary detection method. It first excludes noise points from obstacles, then extracts boundary points using a predefined feature set, and finally fit a curve to those points. The method performs frame-wise processing. The application of deep learning methods has recently been popular. They realize semantic segmentation using point cloud to image conversion (e.g., [19], [20]) or point-based classification (e.g., [21]). They require a dataset for training such as SemanticKITTI [22].

C. LiDAR-camera integration for road recognition

Integration of different sensing modalities is an effective way of increasing robustness. Matsushita and Miura [14] developed a road boundary tracking method combining the color and edge information from a camera and the shape information from a 2D LiDAR. Caltagirone et al. [23] proposed a LiDAR-camera fusion method. It first projects point cloud on the camera image plane with up-sampling and then combines the LiDAR image and the camera image using FCN (fully-convolutional network)-based architectures.

III. Boundary feature extraction

A. Outline of feature extraction

Fig. 1 shows an example of LiDAR data for a road scene. This data was taken by a 32-line LiDAR (HDL-32E, Velodyne). We use the front-half data of the LiDAR for navigation, as shown in Fig. 2. We extract scanline-wise features from such LiDAR data by supposing that a scanline will form a specific shape near road boundary regions.

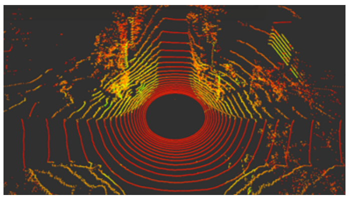


Fig. 1. A LiDAR data example.

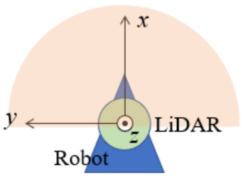


Fig. 2. Robot and LiDAR setting.

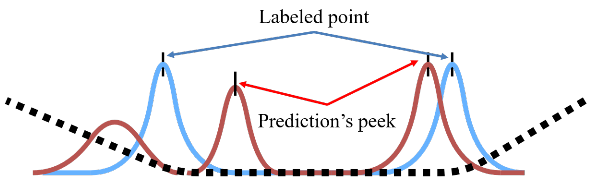


Fig. 6. Peak extraction and evaluation.

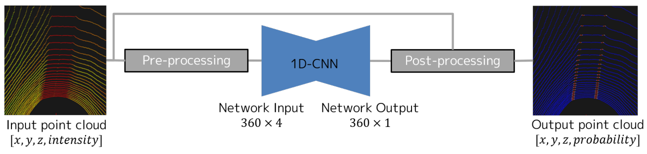


Fig. 3. Process of feature extraction.

Fig. 3 illustrates the feature extraction process. We first convert one scanline data to a fixed-size data sequence and supply it to a neural network to get likelihood values. We then calculate the likelihood for each (original) 3D point by a simple interpolation from those likelihood values.

B. Neural network and dataset

Fig. 4 shows the structure of the network. The network is an encoder-decoder type one for 1D data. The input is a vector of (x, y, z, ref), where (x, y, z) is the 3D location and ref is the reflection intensity.

The data for training were collected by carrying the LiDAR with a pack frame on cart paths in a nearby golf course. The height of the LiDAR is about $2.0\,[m]$. Among 4,000 data collected, we sampled 142 data with a regular interval. Fig. 5 shows an example data and annotation. We take the following steps for dataset generation:

1) Choose at most one boundary point at each side of the road for each scanline data (see Fig. 5(c)). We used an annotation tool[1] for choosing the points.
2) Apply a Gaussian smoothing with $N(0.0, 0.2^2)$ followed by normalizing the maximum value to be 1.0 and letting the values less than 0.1 be zero.
3) We repeat the steps above not only for the frontal data but also for backside ones to get more training data. We get at most 64 lines of scans from one LiDAR measurement.

We also do a data augmentation by a horizontal flip and scale conversion of intensity values so that the size of the final dataset becomes six times larger.

C. Training

We divide the dataset into the training data and the test data with 8:2 ratio. The loss function used is the binary cross-entropy:

$$Loss(y, \hat{y}) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y}),$$

and the hyperparameters used are: 100 epochs, batch size is 16, Adam optimizer, training rate is 0.01.

D. Evaluation

We evaluate the feature extraction in the following two criteria: detection rate and detected location accuracy. Fig. 6 illustrates the relationship between the labeled (i.e., ground truth) point and the predicted peak point. We consider that the boundary point is detected for the first criterion if the peak is within some distance threshold th. For the second one, we take the distance to the nearest boundary point as the measure.

1) Results for Golf park data: Fig. 7 shows the prediction results using the test dataset. In each subfigure, the left and the right images show the predicted and the annotated result, respectively. The color indicates the normalized confidence. Boundaries are mostly correctly extracted. Fig. 11(c) shows the case where the detection rate is lowest, where roadside trees occlude many points near the boundary. Fig. 8 shows the result on accuracy. We change the value of th and

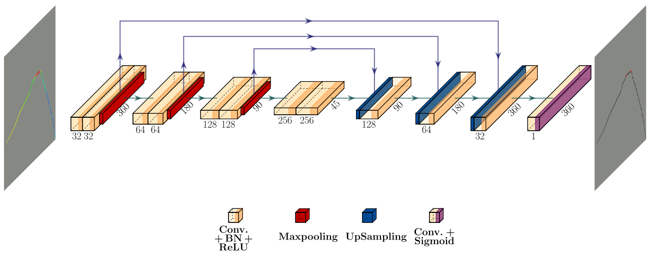
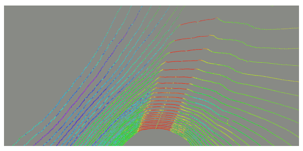
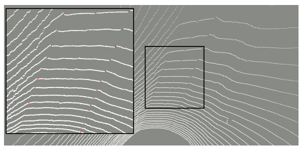


Fig. 4. Network architecture.

(a) Example scene


(b) LiDAR data


(c) Annotation data

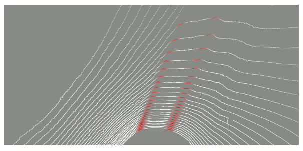

(d) Smoothed annotation data

Fig. 5. Process of training data generation.

[1]https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor

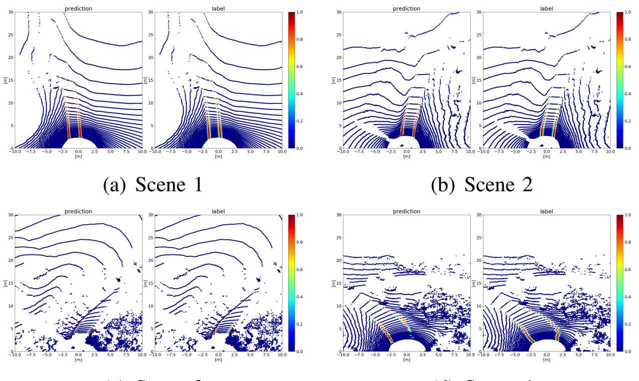


Fig. 7. Prediction results of the proposed method for the golf park data. Left: prediction, Right: Annotation.

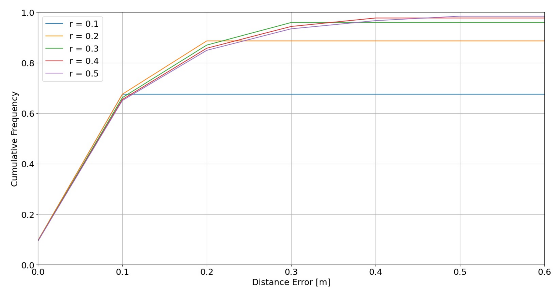


Fig. 8. Boundary localization accuracy of the proposed method for the golf park data.

calculated the cumulative frequency of correct detections. From this result, about 90% of boundary points are detected within $0.2\,[m]$ accuracy, which is reasonably well considering the averaged width of the road ($2.0\,[m]$).

2) Comparison with a heuristic method: For comparison purposes, we implemented a heuristic method for boundary point detection. Similarly to [15], we assess the differences in multiple features, that is, those in angle, height, and reflection intensity. Let p, p_l, and p_r be the point under consideration, the left reference point, and the right reference point, respectively (see Fig. 9). Reference points are set at $1.0\,[m]$ apart from p.

Let p^z and p^i be the z value and the intensity of a point.

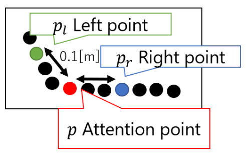


Fig. 9. Reference points.

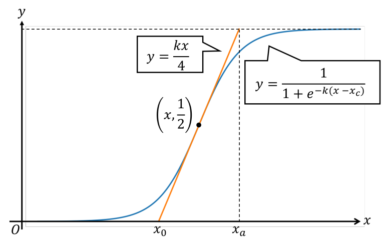

y
$y = \frac{kx}{4}$
$y = \frac{1}{1+e^{-k(x-x_c)}}$
$\left(x, \frac{1}{2}\right)$
O
x_0
x_a
x

Fig. 10. Sigmoid function.

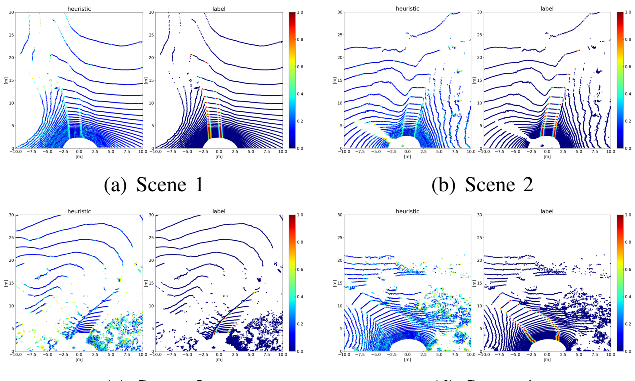


Fig. 11. Prediction results of the heuristic method for the golf park data. Left: prediction, Right: Annotation.

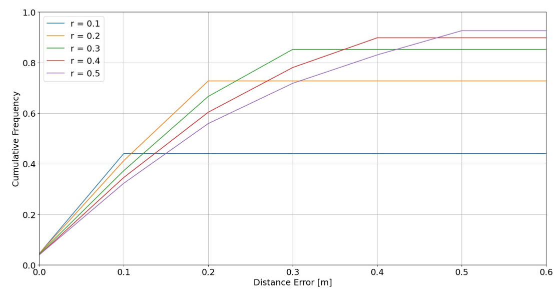


Fig. 12. Boundary localization accuracy of the heuristic method for the golf park data.

The likelihood $L(p)$ is defined as:

$$L(p) = (L_a(p) + L_h(p) + L_i(p))/3, \tag{1}$$

$$L_a(p) = \mathrm{Normal}(\theta_p; \mu_a, \sigma_a^2), \tag{2}$$

$$L_h(p) = \mathrm{Sigmoid}(|p_l^z - p_r^z|; k_h, x_h), \tag{3}$$

$$L_i(p) = \mathrm{Sigmoid}(|p_l^i - p_r^i|; k_i, x_i), \tag{4}$$

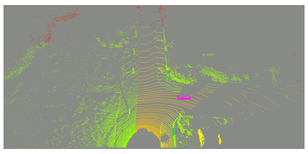
where Normal is the normal distribution with mean μ_a and variance σ_a^2, Sigmoid is the sigmoid function with parameters k and x (see Fig. 10), and θ_p is the curvature at point p, calculated by:

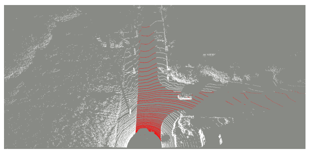
$$\theta_p = \arccos\left(\frac{(p-p_l)\cdot(p-p_r)}{|p-p_l||p-p_r|}\right). \tag{5}$$

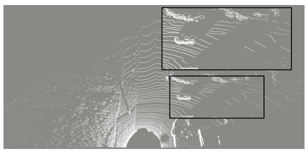
From Fig. 11, we can see that the likelihood values are large around boundary points, but we also see large-value regions in the roadside areas. Concerning the accuracy (see Fig. 12), only 70% of boundary points are detected within $0.2\,[m]$ accuracy. Therefore, the proposed method achieves a considerable improvement over the heuristic one.

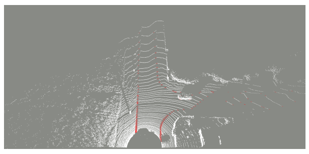
3) Evaluation with SemanticKITTI: SemanticKITTI [22] provides annotated point cloud data. We use the *road*-labeled regions to generate the dataset for boundary points; more specifically, extract a point sequence with the road label longer than or equal to three and mark the endpoints on both sides as boundary points. Although this simple procedure can extract false boundary points at occluding boundaries, we use the extraction result for training. For the testing, we

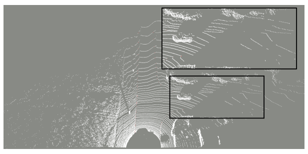
(a) Example scene


(b) Class labels


(c) Road label


(d) Extracted boundary points


(e) Gauss smoothed data


(f) Removal of false boundaries

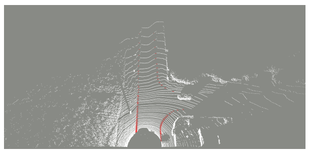
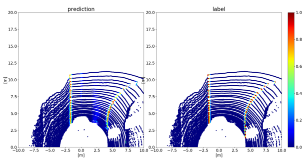
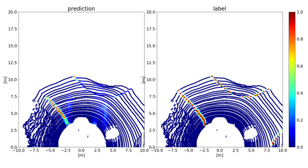

(g) Gauss smoothed data

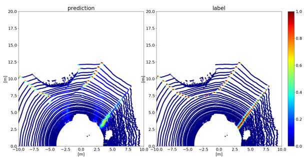
Fig. 13. Dataset generation from SemanticKITTI



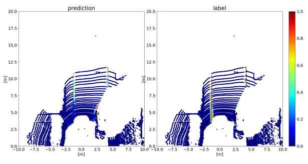
(a) Scene 1



(b) Scene 2



(c) Scene 3



(d) Scene 4

Fig. 14. Prediction results of the proposed method for SemanticKITTI. Left: prediction, Right: annotation.

manually remove such false boundary points. Fig. 13 shows an example step of data generation. For the scene shown in Fig. 13(a), Fig. 13(e) is the data for training (only Gaussian smoothing applied), and Fig. 13(g) is the one for testing (false boundary points removal and Gaussian smoothing).

For the evaluation using SemanticKITTI, we used only the 3D point location data due to a difference in reflection intensity representation between our system and SemanticKITTI. Fig. 14 shows the boundary point prediction results. Prediction is good for Fig. 14(a) and are not satisfactory for the others, especially in complex road shapes. Fig. 15 shows the result on the accuracy. about 75% of boundary points are detected within $0.2\,[m]$ accuracy.

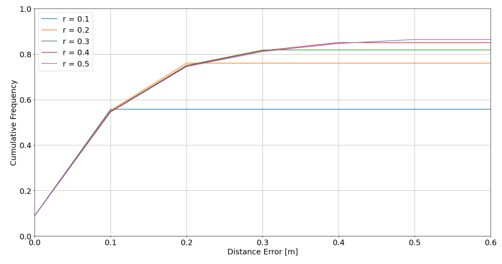


Fig. 15. Boundary localization accuracy of the proposed method for SemanticKITTI.

IV. 3D Road Boundary Estimation

A. Overview

We develop a particle filter-based 3D road boundary estimation method to cope with occasional boundary point extraction failures. The effectiveness of such temporal filtering has been shown in our previous work [15], [16]. Fig. 16 shows the architecture of the whole system. The left-hand side in the diagram is for the boundary point extraction part stated in the previous section, while the right-hand one is the filtering part; those parts communicate with each other via ROS topics.

B. Road boundary model and state vector

The road scenes we deal with in this work include 3D ones with upward and downward road segments. Fig. 17 shows the road model. The model is composed of quadrangular segments. Each segment is represented by the left and the right corner at the front segment: (x^l, y^l, z^l) and (x^r, y^r, z^r). Assuming that $z^l = z^r$, we can consider two inclination angles of the segment, c_{yaw} and c_{pitch}, as shown on the right of the figure. The segment length h is fixed to $0.7\,[m]$ and the number of segments $n = 6$. By adding the robot pose change from the previous pose, $\Delta x, \Delta y, \Delta z, \Delta roll, \Delta pitch, \Delta yaw$, to the segment parameters, we have the following state vector:

$$X = [U, S_1, S_2, \ldots, S_{n+1}]^T, \quad (6)$$

$$U = [\Delta x, \Delta y, \Delta z, \Delta roll, \Delta pitch, \Delta yaw]^T, \quad (7)$$

$$S_i = [x_i^l, y_i^l, z_i^l, x_i^r, y_i^r, z_i^r]^T (z_i^l = z_i^r). \quad (8)$$

In other words, the robot and the road boundary are represented in the previous robot frame. The road boundary paramters are updated after each estimation step of the particle filter.

C. State prediction

The state prediction step considers two factors. One is the robot motion, and the other is road shape change.

1) Prediction of robot pose change: We predict the robot pose change as follows. Since the robot is supposed to move on a locally-planar terrain, we use odometry data for predicting Δx, Δy, Δyaw. We set Δz to the height difference between the nearest two segments, S_1 and S_2, calculated by: $\Delta z = \sqrt{\Delta x^2 + \Delta y^2} \cdot \tan(c_{pitch})$. Δz is set to zero from the constraint $z^l = z^r$. The uncertainties for these pose change parameters

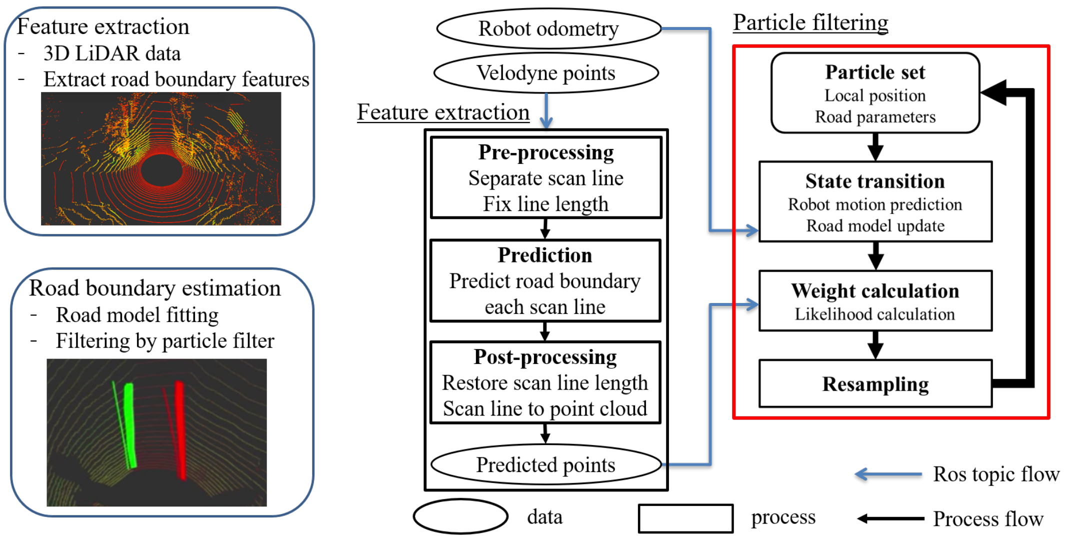


Fig. 16. System architecture.

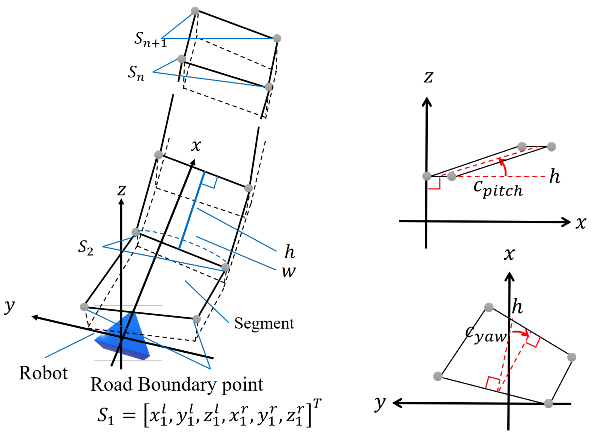
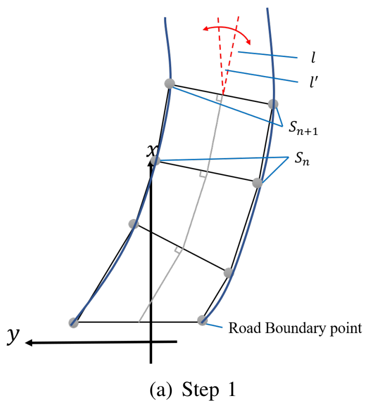
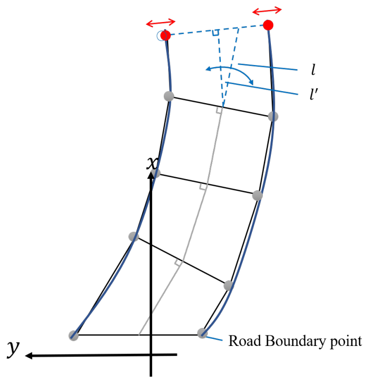


Fig. 17. Road boundary model



(a) Step 1



(b) Step 2

Fig. 18. Road model update steps.

are given by $N(0.0, 0.005^2\,[m^2])$ and $N(0.0, 0.0087^2\,[rad^2])$ for the translational and the rotational motions, respectively.

2) Prediction of road shape change: We predict the road shape change only when the robot passes the nearest segment in the model. In such a case, we delete that segment and add a new one at the farthest location. For keeping the continuity of the segment, we calculate the parameters of the new segment from the two farthest corner points S_n and S_{n+1} and their associated angles c_{pitch} and c_{yaw} just before the update. Since we do not know exactly what segment will come, we apply the following uncertainty models to distributing particles: $N(c_{yaw}, 0.0087^2\,[rad])$ for the horizontal orientation, $N(0.0, 0.002^2\,[rad])$ for the vertical orientation, and $N(\mathrm{w}, 0.1^2\,[m])$ for the width.

Fig. 18 illustrates the road model update procedure. The first step is to set line l with length h in parallel with the centerline of the farthest segment. We then rotate l with the sampled orientation to l' (see Fig. 18(a)). The second step is to set two corner points on the perpendicular line to l' such that the width will be the sampled width (see Fig. 18(b)).

3) Weight calculation: We calculate the weight for each particle by projecting the corresponding road boundary model onto the LiDAR point cloud. Since each point has the confidence value for being a boundary point as explained in the previous section, we collect the points which are within a certain distance r (currently, $r = 0.3\,[m]$) from the left

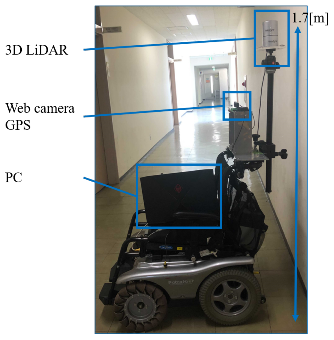


Fig. 19. Our robot.

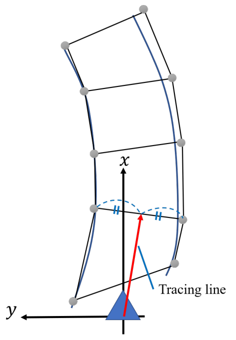


Fig. 20. Control strategy.

and right boundary lines and use their confidence values, weighted by their distances for calculating the weight for a particle. We use the following expression:

$$w = w^{left} \cdot w^{right}, \tag{9}$$

$$w^{left\ or\ right} = \sum_{i=1}^{n} \left[\frac{1}{M_i} \sum_{j=1}^{M_i} Conf(p_j) \left(1 - \frac{dist(p_j)}{r} \right) \right] \tag{10}$$

where n is the number of segments, M_i is the number of nearby points for the ith segment, $Conf(p)$ is the confidence value, $dist(p)$ is the distance to the corresponding segment boundary line. The calculated weights are normalized over all particles and used for resampling.

V. Navigation experiments

A. Robot and control strategy

Fig. 19 shows the robot used for navigation experiments. The mobile base is PatraFour (Toyota Motor East Japan Inc.), equipped with a LiDAR (HDL-32e), a camera, and a GPS receiver. We use RTK-GNSS for obtaining reference location information. The specification of the PC is Intel Core i7-7700HQ 2.8GHz, 16GB memory, and GeForce GTX 1070 Mobile GPU.

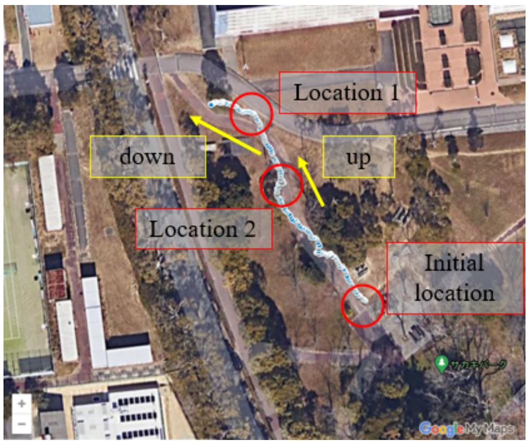
Fig. 20 illustrates the control strategy for road following. We use the second pair of corners (i.e., S_2) for guiding the robot. The corner points are calculated as the weighted averages of those points for all particles. The target tracing line is set to pass the midpoints of the corners and be perpendicular to the line connecting them. We then apply a simple feedback control, in which the translational velocity v is constant $0.25\,[m/s]$ and the rotational velocity w is calculated by:

$$w = k_d d + k_\theta \theta, \tag{11}$$

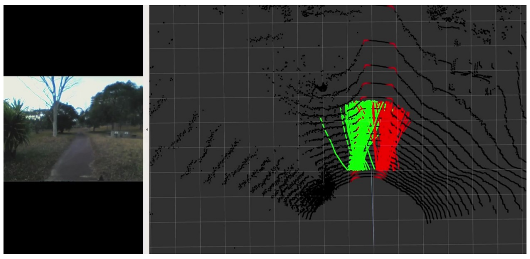
where d is the distance between the robot and the target line, θ is the angle between the robot orientation and the target line; parameters are set as $k_d = 0.2, k_\theta = 0.4$.

B. Navigation result

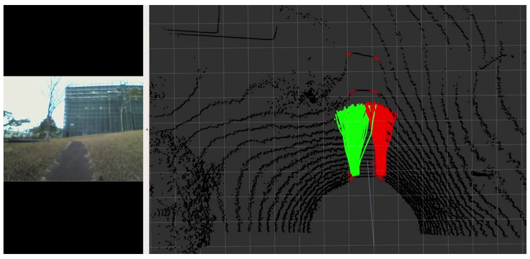
We here show the results of navigation experiments in a park on our campus. Fig. 21(a) shows an aerial image of the road used for the experiments. We set the number of particles to 400. The processing speed for the boundary point extraction was $10\,[Hz]$ and that for the whole system was about $3\,[Hz]$.



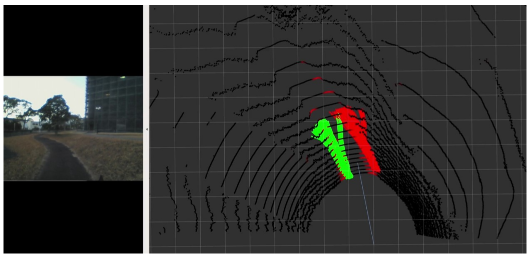
(a) Road segment for the experiments



(b) Initial particle distribution



(c) Particle distribution at location 1



(d) Particle distribution at location 2

Fig. 21. Boundary estimation results.

Figs. 21(b) to 21(d) show the particle distributions at the initial location, location 1, and location 2. The distributions are reasonable at all locations.

Since the method repeatedly estimates the robot pose change, we can recover the trajectory by accumulating the changes. Fig. 22 shows the trajectory recovery results. On the left, we compare the estimated trajectory with the one from GPS and the one by pure odometry in the horizontal plane. The trajectories by the proposed and the odometry-

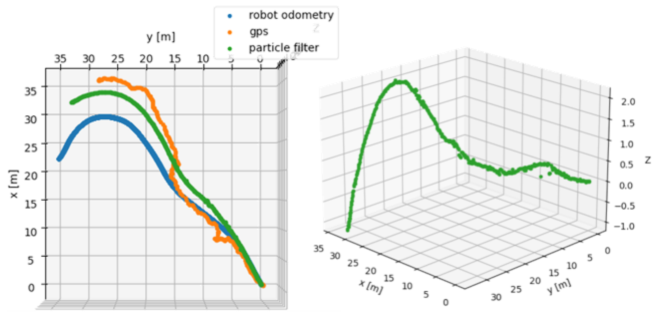


Fig. 22. Trajectory recovery results. Left: comparison among the proposed, the odometry-based, and the GPS-based method in the horizontal plane. Right: 3D trajectory by the proposed method.

based method match with the GPS data reasonably well. At the end part of the route (around location 2), the odometry-based method gives a relatively large error (compare with Fig. 21(a)), whereas the proposed method corrects it with boundary detection and particle filtering. The 3D plot of the trajectory is on the right. Since the GPS did not give sufficient accuracy for estimating the height change on the road, we cannot give a quantitative evaluation, but we can say that the recovered 3D trajectory matches well with the upward and the downward trend of the path (see Fig. 21(a)) qualitatively. This result shows the advantage of using the 3D road boundary model.

VI. Conclusions and Discussion

This paper has described a road boundary estimation method for 3D road navigation with 1D deep feature extraction and particle filter-based 3D road model estimation. We generated a dataset for feature model training and showed that the feature extraction is accurate enough. The feature extraction is also sufficiently fast because each scanline is processed separately. We then constructed a 3D road model with a series of segments and developed its update procedure. The proposed method was applied to actual robot navigation in our campus to show its feasibility in road boundary estimation and robot trajectory recovery.

The current feature extraction model cannot handle complex road shapes. Our 1D feature extraction trades the efficiency and the detection ability. It is future work to consider the relationship between features in adjacent scanlines in feature extraction. It is also future work to extend the current road model to more complex shapes such as branches and crossings, as in [15].

References

[1] C. Badue et al., "Self-driving cars: A survey," *Expert Systems With Applications*, 2020.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[3] Y. Du et al, "Group surfing: A pedestrian-based approach to sidewalk robot navigation," in *Proceedings of the 2019 Int. Conf. on Robotics and Automation*, 2019.

[4] E. Dickmanns, "The development of machine vision for road vehicles in the last decade," in *Proceedings of 2002 IEEE Intelligent Vehicle Symp.*, vol. 1, 2002, pp. 268–281.

[5] J. Leonard et al., "A perception-driven autonomous urban vehicle," *J. of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[6] M. Borjarski et al., "End to end learning for self-driving cars," *arXiv:1604.07316 [cs.CV]*, 2016.

[7] F. Codevilla et al., "End-to-end driving via conditional imitation learning," in *Proceedings of the 2018 Int. Conf. on Robotics and Automation*, 2018.

[8] A. Borkar, M. Hayes, and M. Smith, "Robust lane detection and tracking with ransac and kalman filter," in *the 16th IEEE Int. Conf. on Image Processing*, 2009, pp. 3261–3264.

[9] R. Danescu and S. Nedevschi, "Probabilistic lane tracking in difficult road scenarios using stereovision," *IEEE Trans. on Intelligent Transportation Systems*, vol. 10, pp. 272–282, 2009.

[10] M. Teichmann, M. Weber, M. Zollner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *Proceedings of 2018 IEEE Intelligent Vehicles Symp.*, 2018, pp. 1013–1020.

[11] A. Geiger et al., "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEE Conf. on Computer Vision and Pattern Recognition*, 2012.

[12] W. Wijesoma, K. Kodagoda, and A. Balasuriya, "Road boundary detection and tracking using ladar sensing," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 3, pp. 456–464, 2004.

[13] P. Sun, X. Zhao, Z. Xu, R. Wang, and H. Min, "A 3d lidar data-based dedicated road boundary detection algorithm for autonomous vehicles," *IEEE Access*, vol. 7, pp. 29 623–29 638, 2019.

[14] Y. Matsushita and J. Miura, "On-line road boundary modeling with multiple sensory features, flexible road model, and particle filter," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 274–284, 2011.

[15] T. Chiku and J. Miura, "On-line road boundary estimation by switching multiple road models using visual features from a stereo camera," in *Proceedings of 2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 4939–4944.

[16] K. Mano, H. Masuzawa, J. Miura, and I. Ardiyanto, "Road boundary estimation for mobile robot using deep learning and particle filter," in *Proceedings of 2018 IEEE Int. Conf. on Robotics and Biomimetics*, 2018.

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2017.

[18] S. Thrun and et al., "Stanley: The robot that won the darpa grand challenge," *J. of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[19] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1887–1893.

[20] A. Milioto, "Rangenet ++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4213–4220.

[21] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[22] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[23] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "Lidar-camera fusion for road detection using fully convolutional neural networks," *Robotics and Autonomous Systems*, vol. 111, 11 2018.