Hindawi
Journal of Advanced Transportation
Volume 2022, Article ID 9328398, 13 pages
https://doi.org/10.1155/2022/9328398

Research Article

Vision-Based Branch Road Detection for Intersection Navigation in Unstructured Environment Using Multi-Task Network

Joonwoo Ahn,[1] **Yangwoo Lee**,[1] **Minsoo Kim**,[1] **and Jaeheung Park**[1,2,3]

[1]*Dynamic Robotic Systems (DYROS) Lab., Graduate School of Convergence Science and Technology, Seoul National University, 1, Gwanak-ro, Seoul 08826, Republic of Korea*
[2]*ASRI, RICS, Seoul National University, Seoul, Republic of Korea*
[3]*Advanced Institutes of Convergence Technology, Suwon 443-270, Republic of Korea*

Correspondence should be addressed to Jaeheung Park; park73@snu.ac.kr

Received 21 March 2022; Revised 26 June 2022; Accepted 6 July 2022; Published 5 August 2022

Academic Editor: Carlos Guindel

Autonomous vehicles need a driving method to be less dependent on localization data to navigate intersections in unstructured environments because these data may not be accurate in such environments. Methods of distinguishing branch roads existing at intersections using vision and applying them to intersection navigation have been studied. Model-based detection methods recognize patterns of the branch roads, but are sensitive to sensor noise and difficult to apply to various complex situations. Therefore, this study proposes a method for detecting the branch roads at the intersection using deep learning. The proposed multi-task deep neural network distinguishes the branch road into a shape of rotated bounding boxes and also recognizes the drivable area to consider obstacles inside the box. Through the output of the network, an occupancy grid map consisting of one branch road at an intersection is obtained, which can be used as an input to the existing motion-planning algorithms that do not consider intersections. Experiments in real environments show that the proposed method detected the branch roads more accurately than the model-based detection method, and the vehicle drove safely at the intersection.

1. Introduction

Unstructured environments, such as parking lots or alleyways, contain intersections, and there is a need for autonomous vehicles to navigate intersections in these environments. The conventional autonomous navigation studies [1–3] use a global path with localization data. The global path consists of multiple waypoints that the vehicle can pass through intersections. The localization data are obtained through the global positioning system or by the simultaneous localization and mapping technique [4–6]. However, in an unstructured environment with narrow roads or complex obstacles, the localization data can be inaccurate, which increases the possibility of collision with obstacles.

Driving methods by detecting road and obstacles with local sensor data have been studied rather than tracking the global path with inaccurate localization data [7–10]. Among the local sensors, vision has been widely used due to its low price and low memory usage. For motion-planning at the intersection, it is necessary to recognize the intersection and detect the branch road existing at the intersection. Model and learning-based methods have been studied for branch road detection.

The model-based detection methods recognize and distinguish the branch road existing in an intersection according to a pattern of its shape, size, and direction. In [7–9], a road is segmented and contours of the road are recognized. The branch roads are distinguished according to the difference in position and direction between the vehicle and the road boundary, and entry points are recognized at the widest area of the branch road. Yi et al. [10] proposed a method of converting the road distance from the vehicle into a histogram

using the road segmentation image. Then, the branch road is classified according to the histogram distribution. However, model-based detection methods do not accurately detect branch roads, especially in environments with changes in the width or curvature of the branch road, or in the presence of obstacles [11]. These methods are also sensitive to sensor noise in nonaccurately distinguished road boundaries.

Deep learning-based methods have been studied to address challenges in the model-based method. Object detection methods using the bounding box are used for autonomous driving, and recent studies [12–14] propose improved methods to obtain robust results in a variety of weather and light conditions. Methods for improving detection accuracy [15] and computing efficiency [16] have been proposed. In addition, a method for recognizing roads in various environments (structured, unstructured, lane/line-based, and curb) has been proposed [17]. However, the mentioned studies are difficult to be applied to the environments with intersections.

Recognition methods related to intersections and branch roads using deep learning are as follows. In [18], vehicle kinematic information, point cloud acquired by 3D LiDAR, and OpenStreetMap (OSM) are used to find branch roads at an intersection using machine learning. However, this method [18] uses global data, OSM, and a high-cost 3D LiDAR sensor. The camera image is passed through a long-term recurrent convolutional network to recognize vehicles passing through an intersection [19]. Through the drivable area matching method and model generation method, intersections classified into 7 types are detected [20, 21]. However, these methods [19–21] do not distinguish branch roads at the intersection. A method in [22] distinguishes branch roads in a form of region of interests on a front view image to determine whether a vehicle is driving at an intersection. However, this method is applied only in a structured environment where the width and curvature of the branch road are constant, not in unstructured environments. Further, detected branch roads are not applied to the intersection driving method, and a separate intersection driving algorithm is used.

To address the limitations, this paper proposes a vision-based branch road detection method using deep learning that can learn various shapes of branch roads in unstructured environments and situations with obstacles in the branch road. Branch roads consist of straight, left turn, and right turn roads, which are detected and are represented as a shape of rotated bounding boxes. Compared to using the unrotated bounding box, the rotated version can detect the branch road more accurately even when the vehicle is turning at an intersection. The number of branch roads is recognized differently according to the nonintersection road, and three-way and four-way intersections. At the intersection, one branch road is selected according to a global plan, and the inside of the selected road is regarded as the drivable area of an occupancy grid map. Furthermore, for safe navigating, obstacles inside the selected road are considered in this map by using the segmented drivable area image. This occupancy grid map is used as an input to existing motion-planning algorithms to enable intersection navigation. In addition, drivable area segmentation and branch road detection consist of a single multi-task deep neural network, which improves the learning performance of each task and reduces overall network memory usage.

2. Method

This study proposes a method to detect branch roads existing at an intersection in unstructured environments using deep learning. In addition, a method to obtain inputs for the use of existing motion-planning algorithms at intersections is proposed by combining the data from detected branch road, global planning, and drivable area segmentation. The overall system architecture of the proposed method is shown in Figure 1. The method uses vision, road distance information, vehicle velocity, and navigation information. It does not use the global path and localization data. The navigation information gives the command to vehicles going straight, turning left, or turning right when a vehicle is passing through the intersection. This information is obtained through global planning using a topological map consisting of intersections and roads, before starting the navigation [23].

A multi-task network is proposed to perform two tasks: segmenting the drivable area and detecting the branch roads as rotated bounding boxes. The drivable area segmentation image is used to obtain an occupancy grid map (OGM). Through the detected boxes, it is determined whether a vehicle is driving at the intersection and after which one branch road is selected using the navigation information. A road occupancy grid map (OGM_{road}) is obtained that considers the inside of the selected road as the drivable area. By merging OGM and OGM_{road}, a merged occupancy grid map (OGM_{mer}) is acquired so that the vehicle can drive toward the selected road at an intersection while avoiding obstacles. Thus, OGM_{mer} can be used as an input to existing motion-planning algorithms.

2.1. Bird's Eye View Image Transform. A camera sensor consists of two lenses and is used to detect drivable/non-drivable areas as well as branch roads at an intersection. The front view camera image is transformed to the bird's eye view image which is depicted in Figure 2. The world coordinates ([A, B, C, D], as shown in Figure 2(d)), can be calculated to obtain an 11 m×11 m occupancy grid map by considering the vehicle position. The world-to-pixel coordinate relationship is obtained using the extrinsic parameters, polynomial coefficients, and the camera position and orientation from the vehicle. Through the relationship, a trapezoidal shape ROI, which is the coordinate of the pixel corresponding to the obtained world coordinate ([a, b, c, d], the green area shown in Figure 2(a)), is calculated. As shown in Figure 2(b), the front view camera image vertices ([a, b, c, d]) and vertices of the bird's eye view ([A, B, C, D], as shown in Figure 2(c)), are passed through a getPerspectiveTransform function which aids in the acquisition of a perspective transform matrix. The bird's eye view image is obtained by using a warpPerspective function with the obtained matrix. These functions are in the OpenCV library. The size of this image is 200×200 pixels, and its real world size is 11×11 m.

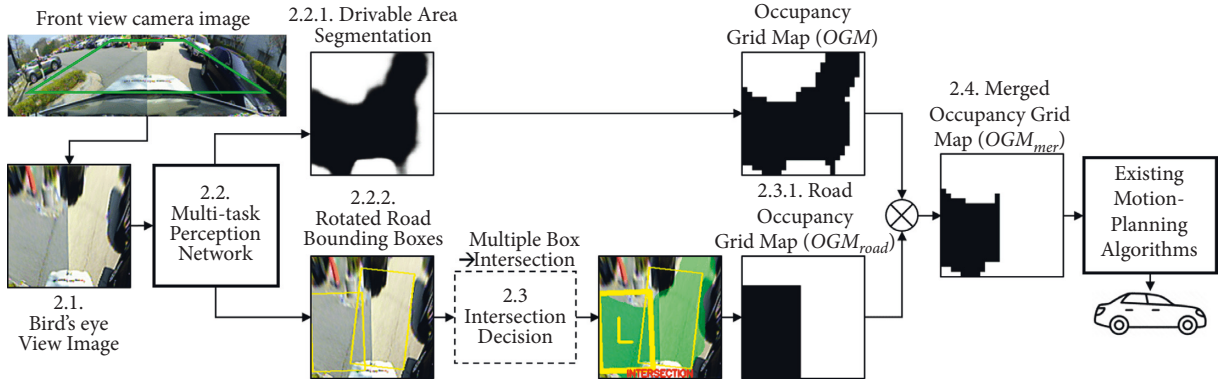


Figure 1: System architecture of vision-based branch road detection and output for navigation.

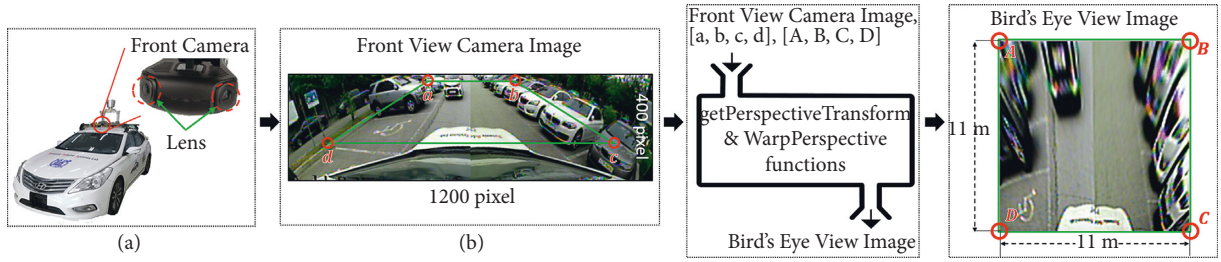


Figure 2: Process of transforming the front view camera image to bird's eye view image.

2.2. Multi-Task Perception Network for Intersection Navigation. The perception network receives the transformed image of the bird's eye view. As shown in Figure 3, the perception network shares one encoder, which is used in YOLOP [24], DLT-Net [25], and Multinet [26]. Then, the output of the encoder is passed on to two decoders. Each decoder performs the drivable area segmentation and rotated road bounding box detection tasks, which receives the same abstracted features for the drivable area from the one encoder. Therefore, the network size, computation time, and GPU usage can be reduced. It also decreases the possibility of overfitting by learning more generalized shared expressions to simultaneously fit multiple tasks.

The structure of the encoder and the segmentation decoder is the same as the Multinet [26]; however, the detection decoder is different. The encoder is based on the VGG16 network [27] which is widely used for training 2D data and shows high accuracy while ensuring real-time computation due to its simple structure. Each decoder and its output are described in detail in below subsections. The sizes of the encoder layers are set according to the 200×200 input. This is shown on the left side of Figure 3. The encoders' weights are initialized using pretrained weights from the ImageNet data [28]. Each decoder and its output are described in detail in below subsections.

2.2.1. Drivable Area Segmentation (Occupancy Grid Map (OGM)). The structure of the segmentation decoder is shown in the upper left side of Figure 3. Features abstracted by the encoder have a low resolution 7×7 with a 1×1 convolutional layer. These features are passed through a convolutional layer and upsampled by three transposed convolutional layers [28, 29]. The size of the segmentation decoder layers is related to the encoder and is the network size for outputting 200×200. In addition, each convolutional layer of the encoder is combined with the decoder's layers with skip connections [30] to extract high-resolution features from the lower layers (see Figure 3). The convolutional layers are initialized using the scheme in [26] which performs bilinear upsampling to segment two classes: drivable and nondrivable.

The output of the segmentation decoder is a probability of the drivable at each pixel in the input image, which is shown in Figure 4(b). The closer the pixel is to black, the more likely it is drivable. The Otsu algorithm [31] is used to calculate the threshold value (Figure 4(c)). This algorithm divides the pixels into two classes by randomly setting a boundary value and repeatedly obtains the intensity distribution of the two classes. Then, it selects the boundary value that makes the distribution of the values of the two classes most uniform. In other words, an optimal threshold value at which the ratio difference between binary-classified pixels can be smallest is obtained. The size of the segmented image is 200×200, and it is converted to the occupancy grid map (*OGM*). This map is a 2D map that divides the image into 25×25 grids as shown in Figure 4(d). The grid is considered occupied even with only a single nondrivable pixel in the grid cell. That is, only drivable pixels exist in the unoccupied grid.

The data labeling criteria for training the drivable area segmentation image are in the image as follows: roads, road marks, a stop line, and crosswalk are labeled as the drivable

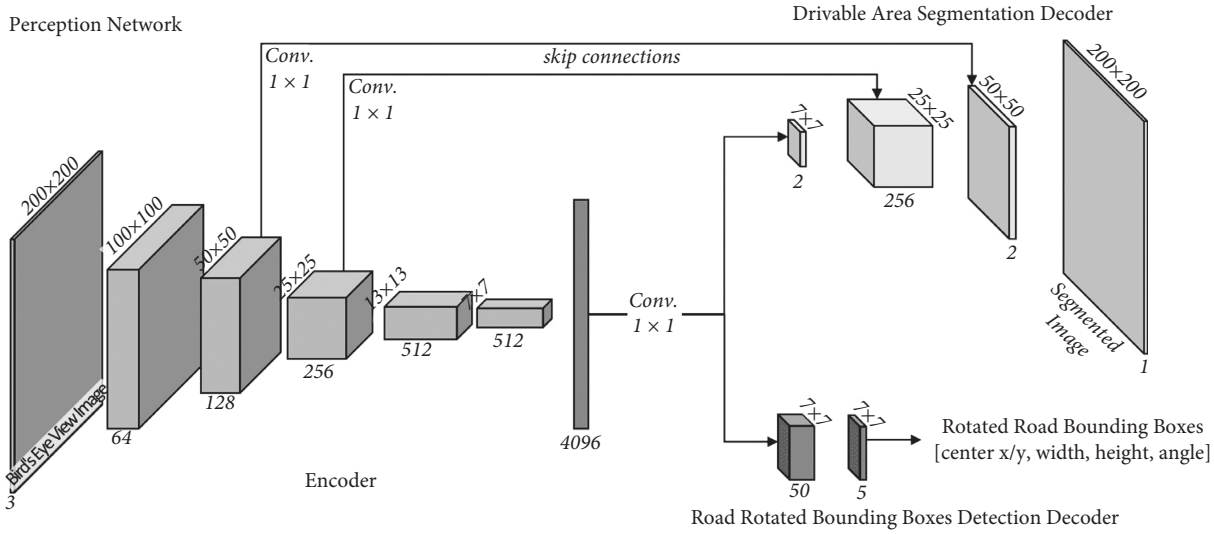


Figure 3: Multi-task perception deep neural network for intersection navigation.

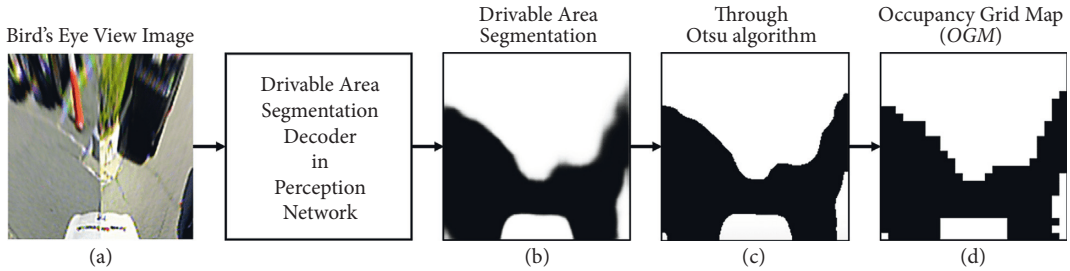


Figure 4: Process of drivable area segmentation to obtain occupancy grid map (*OGM*).

area. The other area except the drivable area is designated as the nondrivable area. Sidewalks, parking spaces (including parking line), road boundary lines, pedestrian walkways, and vehicles are regarded as the nondrivable area.

2.2.2. Rotated Road Bounding Box Detection. In this study, the rotated bounding box is used to detect and distinguish the branch roads. In general, the bounding box is recognized as an unrotated form in the image frame. In this study, the rotated bounding box is used because, when a vehicle is driving at an intersection, the direction of the road differs from that of the vehicle. In this case, if the unrotated bounding box is used, it is not possible to cover all the drivable area without including obstacles inside the box (see Figure 5(c)). Moreover, an area that is not an actual branch road can be mistaken as a branch road (see the third (center box), fourth (right box), fifth (center box) images in Figure 5(c)). The rotated bounding box can detect one branch road as one box and cover the drivable area as much as possible (see Figure 5(b)), and the detection network can accurately find the box with a feature of the branch road.

A structure of the detection decoder is illustrated at the bottom right side of Figure 3 and is like the YOLO network [32]. The abstracted features, and the output of the encoder, pass a 1×1 convolutional layer with 50 filters. These passed features are divided into 7×7 grids g, and a 7×7×50 shape tensor is obtained. Then, this tensor is passed through another tensor with 7×7×5 shape to obtain a box's detail information, where 5 in 7×7×5 means the number of the channels of the rotated bounding box, b ; the first to fourth channels represent the road bounding box coordinates, the x and y coordinates of the box's center, and the width and height ratio of the box to the image. The fifth channel is a rotated angle value of the rotated box.

Each grid g is assigned a bounding box b. The origin of the grid and box coordinates is the upper right corner, and the box labels are parametrized based on a grid's position.

$$\begin{aligned} g_x &= \frac{x_b - x_g}{w_g}, \\ g_y &= \frac{y_b - y_g}{h_g}, \\ g_w &= \frac{w_b}{w_g}, \\ g_h &= \frac{h_b}{h_g}, \end{aligned} \tag{1}$$

where x_g, y_g and x_b, and y_b are the center coordinates of g and b; w and h denote width and height; and w_g and h_g are

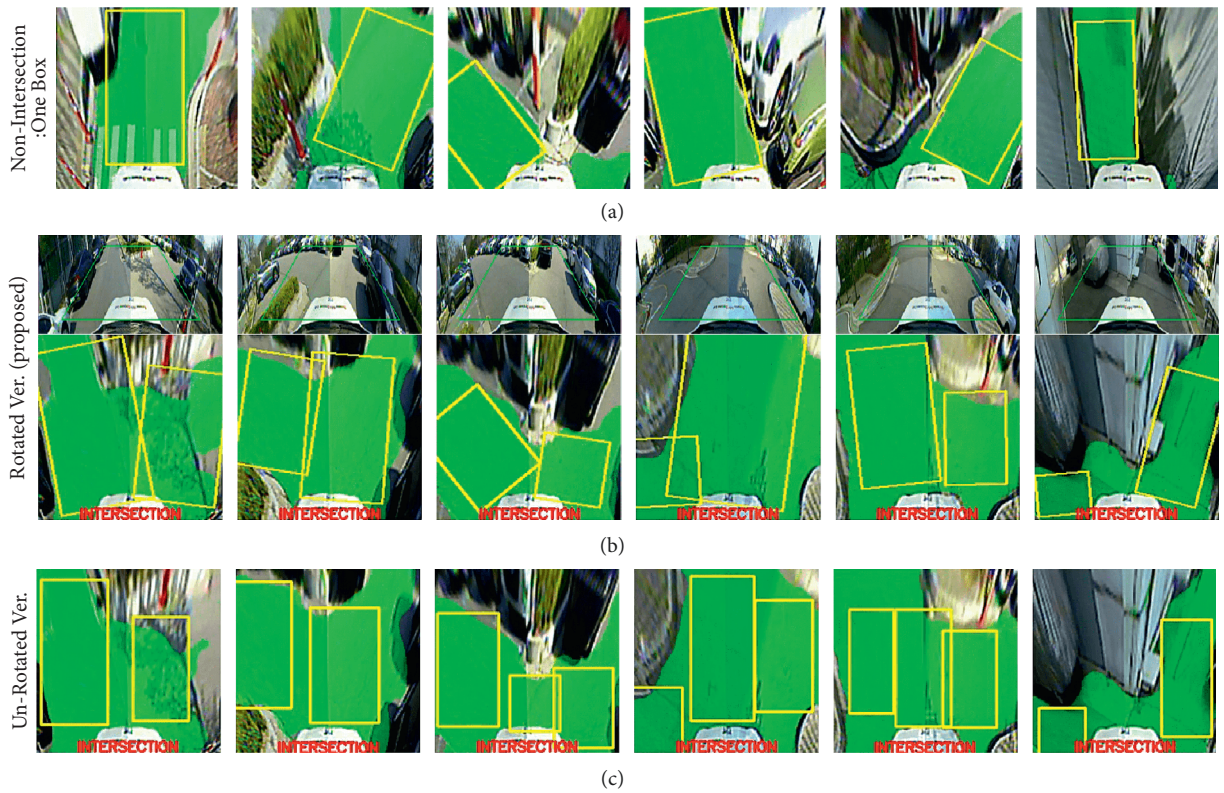


FIGURE 5: Rotated road bounding box detection results; (a) nonintersection case, (b) intersection case with rotated bounding box, and (c) unrotated bounding box.

the grid size, These units are pixels. The loss value in each grid is calculated by the following equation.

$$\mathscr{L}(g,\hat{g}) = g_p\left(\left|g_x-\hat{g}_x\right|+\left|g_y-\hat{g}_y\right|+\left|g_w-\widehat{g_w}\right|+\left|g_h-\hat{g}_h\right|+\left|g_a-\hat{g}_a\right|\right), \tag{2}$$

where g_p indicates whether a box exists in the grid, and if it exists, it is 1, otherwise 0. That is, a valid loss is reflected only when the box exists on the grid; g is the ground truth of the grid and $\hat{g}$ its prediction value; g_a is the angle of the rotated box in radians. The loss per image is the average over the losses of all grids.

Through the 7×7×50 shape tensor, 50 boxes are predicted, and boxes for the case where a confident value of the box is less than a threshold are filtered out (blue boxes in Figure 6). Here, the confidence value is calculated as the largest confidence value of the width and height channels. Through the nonmaximum suppression algorithm [33], a box with the highest confident is selected among other sets of boxes where the overlapping area between boxes exceeds 50% (yellow boxes in Figure 6).

The data labeling criteria for training the rotated road bounding box are as follows. (1) The drivable area between the front of the vehicle's bonnet and the front end of the image is labeled as a rectangle. In this case, the width of the rectangle should not exceed the vehicle's width. (2) Additional labeling is needed if there are unlabeled drivable areas on the side of the image. (3) The maximum number of boxes is three. (4) The size of the box should exceed $7m^2$. (5) Overlapping between two different boxes is possible but should be labeled to avoid overlapping area exceeding $5m^2$. (6) When labeling the rotated bounding box, one or two corners of the box can be outside the image to label it as large as possible.

2.3. Intersection Decision. When two or more boxes are recognized, it is regarded that the vehicle is driving at an intersection. As shown in Figures 7(a) and 8(d), multiple boxes can be detected at a nonintersection road, which is a false-positive case. To deal with this, a road distance between the center of intersections d_{inter_i} and an accumulated distance d_{veh_i} is defined. d_{inter_i} is defined as an edge distance (road) between two nodes (intersection) of the topological map, which is between the i'th and i+1′th intersections. Here, i is an index according to the order of the intersection, and a case of $i=0$ indicates the starting point (not an intersection). The order of visiting the intersection on this map is determined according to the navigation information. d_{veh_i} is the accumulated distance moved by a vehicle from the i'th intersection and is calculated using the vehicle's velocity.

If two or more boxes are recognized, and a difference between d_{inter_i} and d_{veh_i} is lower than d_{th}, it is determined that the vehicle is driving at an intersection:

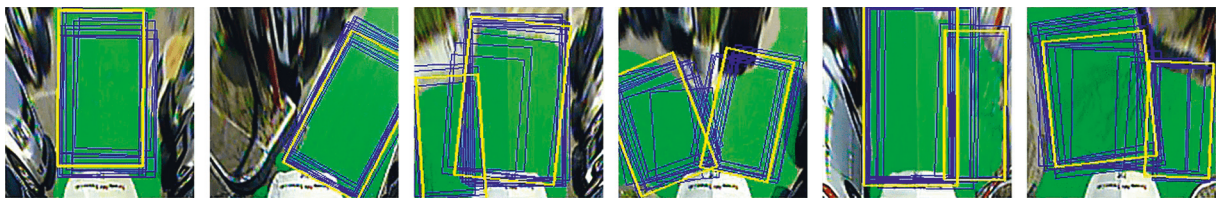


Figure 6: Nonmaximum suppression algorithm; blue box: raw output of detection decoder (not grouped); yellow box: finally detected (grouped) box through the blue boxes.

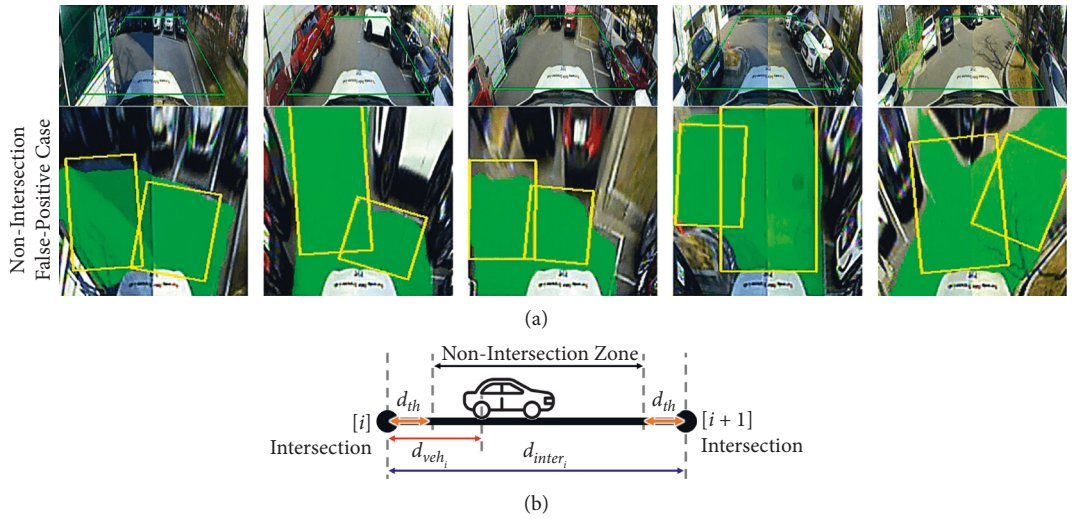


Figure 7: (a) Two or more road boxes are detected in a nonintersection road: false-positive case. (b) Intersection decision using a distance between intersections and the driving distance additionally.

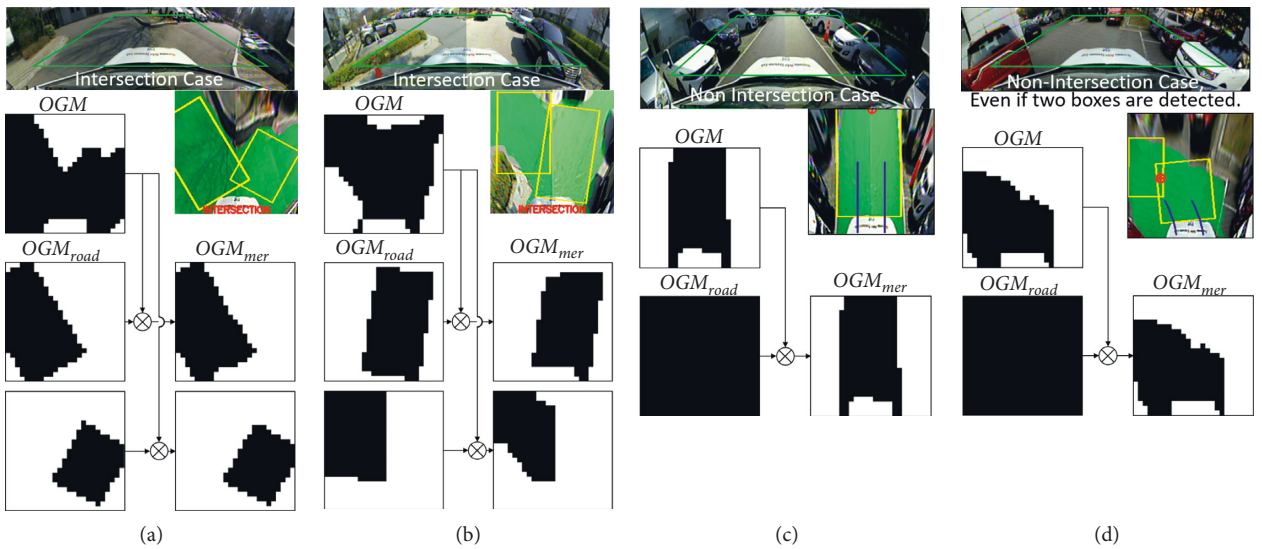


Figure 8: Processes of obtaining merged occupancy grid map (OGM_{mer}).

$$\text{Intersection Flag} = \begin{cases} \text{True}, & \left|d_{inter_i} - d_{veh_i}\right| \le d_{\text{th}} \,\&\, ''\text{Number of box} \ge 2'' \\ \text{False}, & \left|d_{inter_i} - d_{veh_i}\right| \le d_{\text{th}} \,\&\, ''\text{Number of box} < 2'', \\ \text{False}, & \left|d_{inter_i} - d_{veh_i}\right| > d_{\text{th}}. \end{cases} \tag{3}$$

Thus, if the vehicle is located at a distance of d_{th} between the intersection i'th and i+1′th as shown by the black arrow in Figure 7(b), it is regarded as not passing the intersection, even if more than two boxes are recognized. i is increased when intersection flag in (3) changes from true to false and $|d_{inter_i} - d_{veh_i}|$ is larger than d_{th} together.

2.3.1. Road Occupancy Grid Map (OGM_{road}). When it is determined that the vehicle is passing the intersection, one box is selected according to the navigation information, such

as going straight, turning left, and turning right. This information is obtained through a global plan to visit all the roads in the topological map [23]. This process is shown in Figure 1 and Figure 8. The criteria for selecting one box are as follows.

Going straight: the box closest to the middle of the image.

Turning left: the leftmost box in the image.

Turning right: the rightmost box in the image.

Through the selected box, a road occupancy grid map (OGM_{road}) is obtained, as shown in Figure 8. An inside region of the selected box is regarded as the drivable area, whereas its outside is considered as the nondrivable area. The image segmented according to the selected box is converted into OGM. If only one box is recognized, it is determined that the vehicle is not driving at the intersection, so all area of OGM_{road} is regarded as drivable area. This is shown in Figures 8(c) and 8(d).

2.4. Merged Occupancy Grid Map (OGM_{mer}). To navigate intersections, using existing motion-planning algorithms that do not take the intersection navigation into account, a merged occupancy grid map (OGM_{mer}) can be used as an input for these algorithms (see Figure 8). At intersections, several branch roads exist in the drivable area in OGM, and these algorithms cannot calculate the action of which branch drivable area to head to. To address this problem, the drivable area of OGM_{road} consists of the drivable area of OGM_{mer}. In addition, to consider obstacles existing inside OGM_{road}, only the common drivable between OGM_{road} and OGM is drivable in OGM_{mer}.

$$\text{Drivable Area of } OGM_{mer} = \text{Drivable Area of } OGM_{road} \cap \text{Drivable Area of } OGM. \tag{4}$$

The drivable area on each map is defined as the "true" value (the black area (grid) in Figure 8), and the nondrivable area as the "false" value (the white area (grid) in Figure 8). Each grid in OGM_{mer} is calculated by the ampersand operator (&) between OGM and OGM_{road}, and becomes a true value only when the grids of both maps are the "true" value. Thus, at an intersection, in OGM_{mer}, only one branch road among several branch roads is the drivable area, as if it were not the intersection. At nonintersection roads, OGM_{mer} is equivalent to OGM, since all grids of OGM_{road} are regarded as the drivable area (see Figures 8(c) and 8(d)). Therefore, by using OGM_{mer}, the motion-planning algorithm can calculate the vehicle's action as it heads to the drivable area of one branch road while avoiding obstacles.

3. Experimental Setup

3.1. Vehicle and Camera Setup. As shown in Figure 9, an autonomous vehicle, Hyundai HG 240, was used in experiments. One laptop computer was used to implement the proposed method. For training and executing the deep

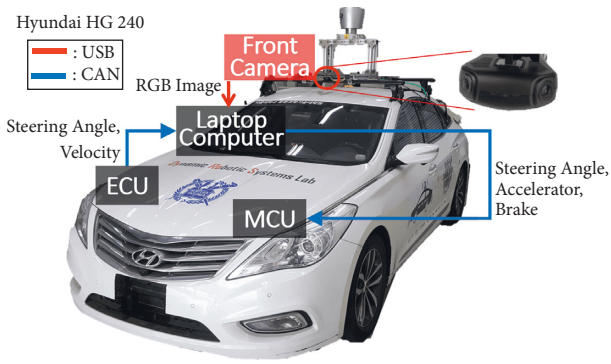


Figure 9: Autonomous vehicle hardware system architecture.

neural network, Nvidia GTX 1080 TI 8G GPU was used. The CPU was 3.9 GHz Intel i9-8950HK 2,9 GHz with a memory size of 16 gigabytes.

A camera was attached 1.55 m above the ground and 0.25 m away from the center of the vehicle. Moreover, it was rotated about 20° downward for the ground direction. This is to minimize the shaded area in the conversion of the image to the bird's eye view image. This camera consists of two lenses to widely recognize the environment, and the two images obtained are concatenated. The field of view of each lens is 120° so that the branch roads can be sufficiently recognized. Distortion of the image is corrected by obtaining the focal length, principal point, and distortion coefficients.

3.2. Multi-Task Perception Network Training Details. A method for training a multi-task-based perception network followed a classic fine-tuning pipeline [26]. The weights of each decoder were calculated and individually updated with the different loss values. To update the weights of the encoder, two-loss values were added with different weightings (segmentation: 25%; detection: 75%).

$$\mathscr{L}_{enc} = \frac{\alpha\mathscr{L}_{seg} + (1-\alpha)\mathscr{L}_{box}}{2}, \tag{5}$$

where $\mathscr{L}_{enc}$ is multi-task loss function of the encoder; $\mathscr{L}_{seg}$ and $\mathscr{L}_{box}$ are losses for the drivable area segmentation and the rotated bounding box detection; and α is a ratio that adjusts the training importance between $\mathscr{L}_{seg}$ and $\mathscr{L}_{box}$ and is set to 0.25. The weights of each decoder were calculated and individually updated with $\mathscr{L}_{seg}$ and $\mathscr{L}_{box}$. Through this process, different data and training hyperparameters could be applied to each decoder.

The loss function of the drivable area segmentation ($\mathscr{L}_{seg}$) decoder was softmax cross-entropy. It infers the drivable and nondrivable probability values for each pixel. The average value of all pixels becomes a loss value of segmentation. The road rotated bounding detection ($\mathscr{L}_{box}$) decoder was trained with the L1 (regression) losses for each five-channel value in the grid of 7×7 cells. The five channels are values for the box's position (x/y), width, height, and angle. These values are summed with equal weight. If the grid is nondrivable, the sum of the detection loss becomes zero. Additionally, the confidence values of the width and

height channels are trained with the cross-entropy loss function.

The Adam optimizer with a 10^{-5} learning rate was used to train the perception network. The weight decay of $5{\cdot}10^{-4}$ was applied to all layers, and a dropout with 50% probability was applied to all 1×1 convolutional layers of the detection decoder. Epochs were 10k, and the batch size was set to 128. Training time was about 3 hours when using a laptop.

The dataset was collected in three parking lots having 18 intersections (see Figure 10). One bird's eye view image per second was collected while driving, and 1,069 images were collected. In each task, 80% of the dataset was used for training, and the remaining portion of the dataset was used for validation. The entire image was used to segment the drivable and nondrivable areas by using a pixel annotation tool [35]. Among the collected images, half of the nonintersection data were excluded, and the road bounding box was trained using 772 images. The reason not to use the entire collected data is to balance the ratio between the nonintersection and intersection data by excluding a similar situation from the nonintersection data. Therefore, the detection accuracy could be higher at the intersection than when using the entire data.

3.3. Model-Based Branch Road Detection Method. The intersection scan model (*ISM*) method [10] is used to compare the performance of detecting branch roads with the proposed method. The *ISM* method recognizes branch roads at the intersection using the segmented image, without the global information. *ISM* [10] defines 21 traversable lines from a scan center point to the end of the image according to a traversable direction [13°, 21°, ..., 165°, and 173°]. The traversable directions are divided into the left turn (6 traversable lines), straight (9 traversable lines), and right turn (6 traversable lines) on the horizontal axis of the histogram.

If there are obstacles on the traversable line, an obstacle distance from the scan center point to the closest obstacle on the traversable line is calculated (blue lines in Figure 11). The traversable distance ratio of the traversable line's length to the obstacle distance is obtained. These ratios according to the traversable direction are used to obtain an *ISM* histogram (see the red dashed box in Figure 11). If more than half of the traversable distance ratio exceeding the threshold (0.7) exists in the traversable direction section, it can be determined that the branch road exists. Here, the distance ratio is the vertical axis of the *ISM* histogram.

4. Experimental Results

4.1. Quantitative Analysis of Multi-Task Network. The perception network was tested by validation dataset. The pixel accuracy metric was used to evaluate the performance of the drivable area segmentation:

$$\text{Pixel Accuracy} = \frac{\text{number of correctly classified pixels}}{\text{number of total pixels}}, \quad (6)$$

where the numerator is the number of cases matching the output of the network and the label in the data, in each pixel. The denominator which was 200×200 is the number of pixels in the bird's eye view image.

Detecting result of the rotated road bounding boxes was used over union, the intersection over union (IoU) metric:

$$\text{IoU} = \frac{\text{area of overlap}}{\text{area of union}}, \quad (7)$$

where the numerator represents the size of the overlap between the label box and the prediction box. The denominator represents the region for the union of the label box and the prediction box. High IoU indicates that the performance of the proposed network is high. Table 1 shows the performance of the network for each task in Figure 10 parking lots using the validation images.

As shown in Figures 12 and 13, the resulting images of the segmentation task are indicated as the green area in the bird's eye view image or the black area in *OGM*. The detection task result is indicated by the yellow squares. The inference speed of the perception network was an average of 23.7 fps.

4.2. Comparison of Branch Road Detection Methods. To evaluate the intersection recognition performance, the proposed rotated road bounding box detection algorithm and *ISM* [10] algorithm were tested at the same image. A branch road detection accuracy was defined as follows:

$$\text{Intersection Accuracy} = \frac{N_{match}}{N_{inter}}, \quad (8)$$

where N_{match} represents the number of matched cases between a label branch box for the validation dataset and an output branch box by each algorithm. N_{inter} represents the number of images containing the intersection. For example, if there were turning left and straight boxes in the data, and the output of the proposed algorithm was the turning left and straight boxes, it is regarded as the correctly recognized case. In the same situation, if *ISM* recognizes only the straight navigation information, it is not considered as a correctly recognized case. The results are shown in Table 2.

"#n" in Figure 12(a) and "#n'" in Figure 12(b) are results in scenes for slightly different spots at the same intersection. Figure 12(a) represents situations where two algorithms correctly detected the branch roads. Figure 12(b) shows that the proposed algorithm was recognized accurately; however *ISM* did not. There were cases in which a specific branch road at an intersection was not recognized. #1' and #2' in Figure 12(b) are examples where some drivable area was recognized as nondrivable (perception noise), so the traversable distance ratio in *ISM* was calculated shorter (indicated as the red lines in #1' and #2' of Figure 12(b)) than a real ground truth traversable distance ratio. Even if the inside of the rotated bounding box is recognized as not all drivable, the proposed detection algorithm had high accuracy, because it inferred the overall shape and distribution of the drivable area inside the box.

Cases #3', #4', and #5' of Figure 12(b) are cases where the branch road was not detected in *ISM* since the area behind the obstacle was recognized as the nondrivable area due to

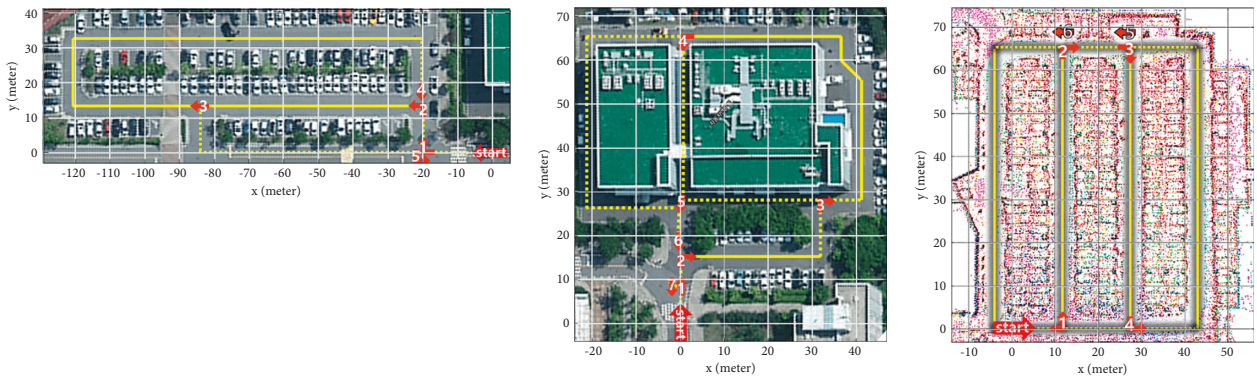


Figure 10: Parking lot environments for collecting perception data, (a, (b) outdoor parking lot and (c) indoor parking lot; the map image was built by LiDAR-based SLAM [34].

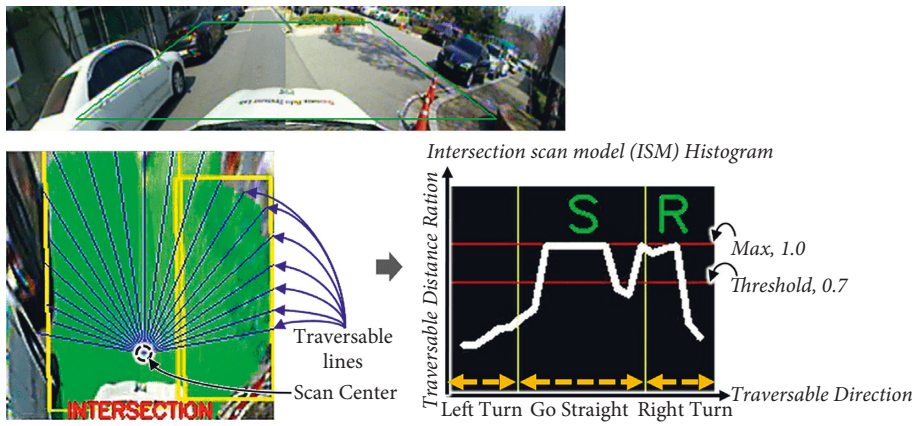


Figure 11: Results for intersection scan model (ISM) method.

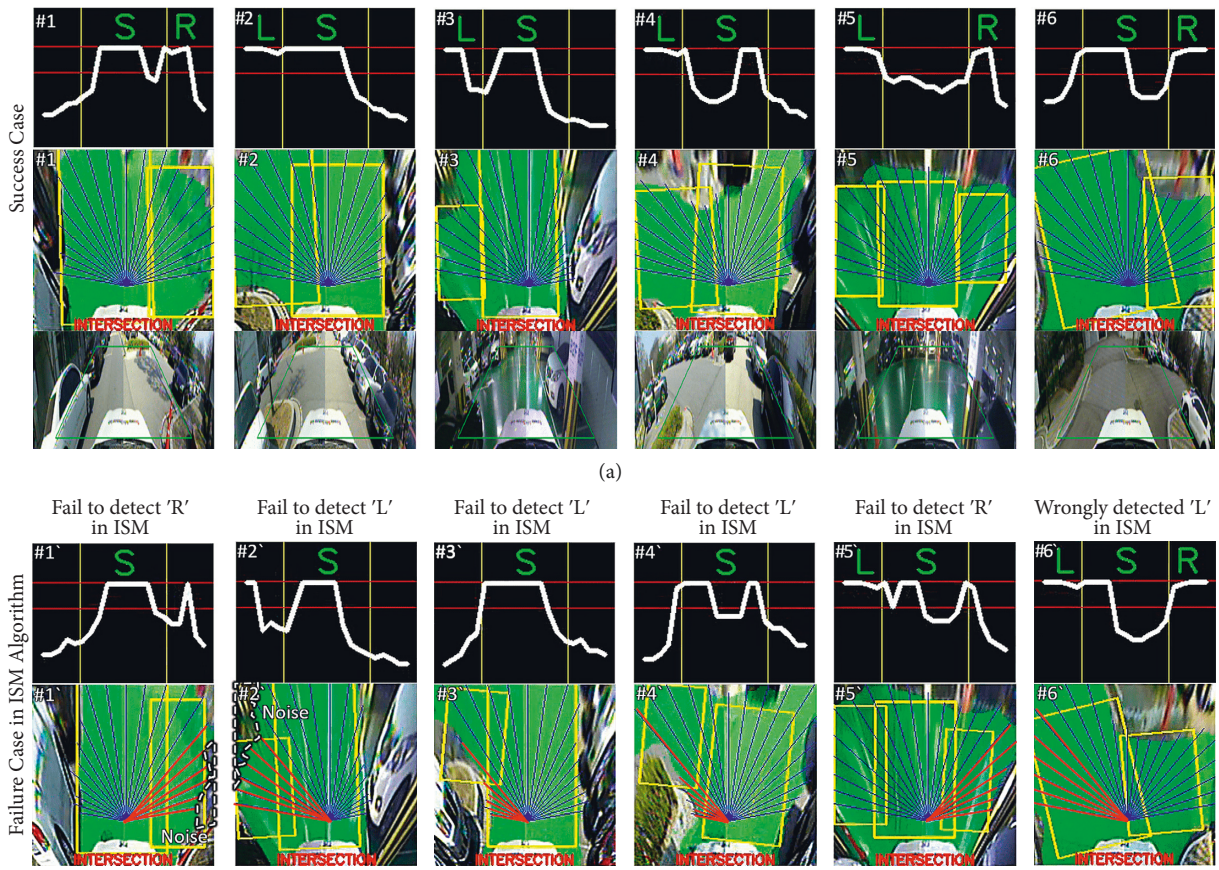
Table 1: Results of the perception network.

		Parking lots		
		Figure 10(a)	Figure 10(b)	Figure 10(c)
Drivable area segmentation (pixel accuracy, (6))		95.81%	94.15%	93.73%
Branch road detection (IoU, (7))	Rotated box (Proposed)	98.58%	97.31%	97.10%
	Unrotated box (Figure 5(c))	97.73%	95.17%	94.79%

the shadow of the camera. The traversable distance ratios of the left (#3' and #4') and right (#5') turn sections were short, and each branch road was not detected. In #6', the straight branch road was too wide, and the drivable distance ratio of the left turn was recognized as too long and was mistaken as the left branch road. Situations #3', #4', #5', and #6' may be addressed by properly tuning the model parameters of *ISM*, although it is unclear whether these parameters can be applied to other situations. However, the proposed learning-based method was able to accurately find the branch roads without finding the model parameters.

In addition, the computation time between the two methods was additionally compared. The proposed method, multi-task network, showed 27.3 fps, and *ISM* showed 29.5 fps. Although *ISM* showed a higher calculation speed, the difference between the two methods is not much different. These methods are faster than the driving control period, 20 fps.

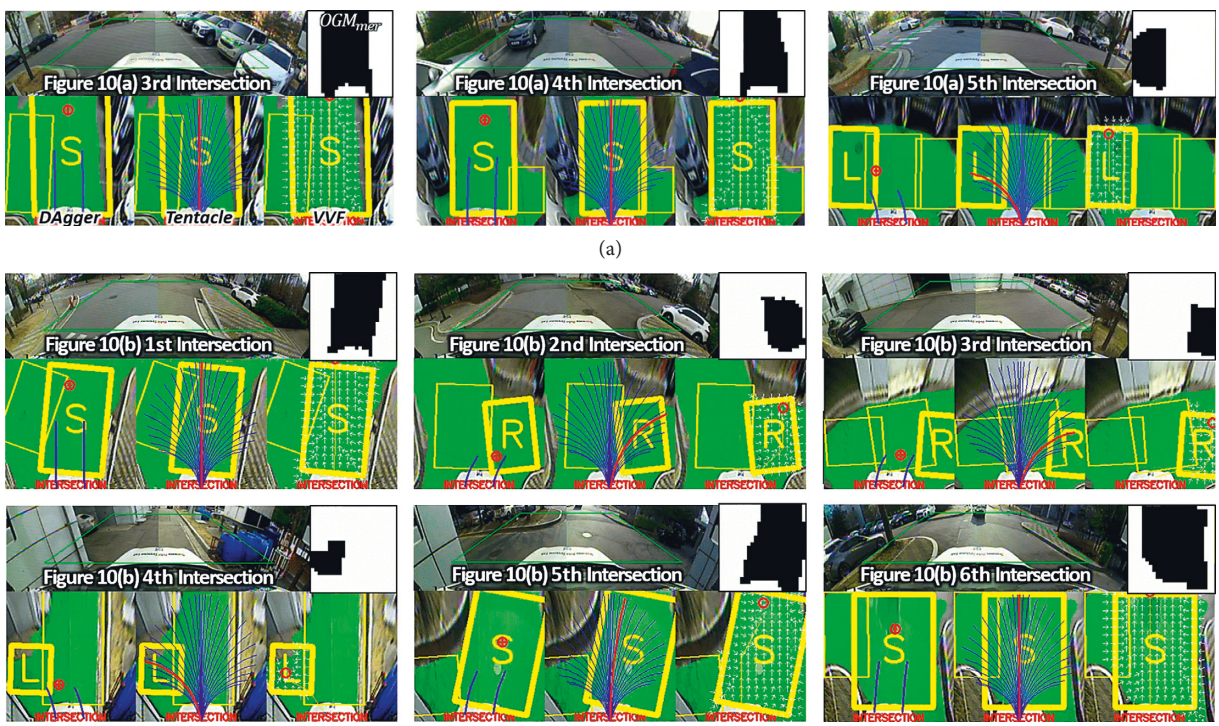
4.3. Results of Applying Existing Motion-Planning Algorithms. The model-based motion-planning algorithms (Tentacle [36] and *VVF* (velocity vector field) [37]) and learning-based motion-planning algorithm (DAgger (data aggregation) [38]) were used for the intersection driving test. These algorithms utilized OGM_{mer} as input and did not require the localization data, global path, and goal point. Tentacle creates candidate paths around the vehicle and selects one candidate path with the fewest obstacles and smallest steering angle change. *VVF* generates a repulsive field for obstacles and a velocity field that the vehicle can drive



(a)

(b)

Figure 12: Branch road detection results at intersection; (a) and (b) are results of different *ISM* performances at different spots at the same intersection; (a) detection success case, (b) detection failure case in *ISM* algorithm [10].



(a)

(b)

Figure 13: Continued.

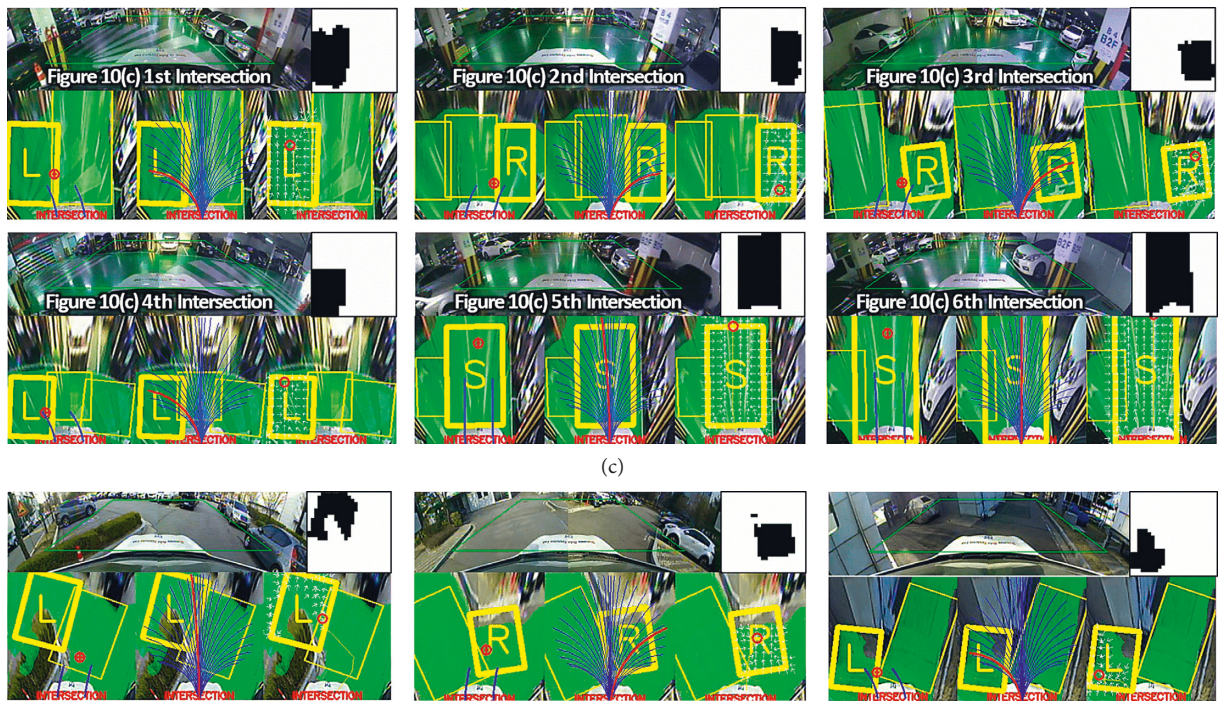


Figure 13: Branch road detection results and intersection navigating results using motion-planning algorithms; (a) map1 (Figure 10(a)), (b) map2 (Figure 10(b)), and (c) map3 (Figure 10(c)) and (d) case with obstacles existing in branch road box.

Table 2: Intersection detection results.

	Rotated box (protected)	Unrotated box (%)	*ISM* algorithm [10]
Intersection accuracy (8)	97.2	95.4%	82.7%

forward. The look-ahead point is searched by descending along the field's direction. DAgger is an algorithm that repeatedly performs imitation learning trained to imitate an expert driving. The expert collects data by selecting the look-ahead point at which the vehicle can drive toward the drivable area while avoiding obstacles. Here, the look-ahead point is a point where the vehicle should reach (red point in Figure 13) [39].

Three motion-planning algorithms were tested in three parking lots (see Figure 10), and the driving results are shown in Figure 13. They were able to calculate an action toward the branch road and could avoid obstacles existing at the branch road because obstacles were regarded as the nondrivable area in OGM_{mer} (see Figure 13(d)). Among the total 18 intersections, the vehicle using Tentacle and *VVF* drove through intersections successfully except for two intersection situations where two intersections were close together (Figure 10(a) 1st and 2nd intersections) and the situation in which the branch road was narrowed by an obstacle (Figure 10(b) 4th intersection). In contrast, the learning-based algorithm, DAgger, did not encounter any problems in the abovementioned two intersections and was able to navigate the entire intersection in the three parking lots. The driving result in the indoor parking lot (Figure 10(c)) was recorded as video1. Therefore, intersection navigation is possible by applying the proposed branch road detection method to the existing motion-planning algorithms, except for complex situations that the model-based motion-planning algorithms cannot handle well.

5. Conclusions

This study proposed a method that detects branch roads at an intersection using vision and deep learning, which can be used alongside the existing motion-planning algorithm for navigating in an unstructured environment. The proposed multi-task network distinguished the branch roads at an intersection as the rotated bounding box. At the intersection, the inner area of a box selected through the navigation information was regarded as the drivable area while considering obstacles. Testing in the parking lot, the proposed method detected the branch roads more robustly than the model-based method using the distance and direction histogram of the branch road in the cases where branch roads varied in size and shape, or the drivable area was detected noisy. In addition, the vehicle successfully navigated the intersection by applying the proposed perception method to the existing motion-planning algorithms such as the tentacle, field, imitation learning algorithms without using

global information. In the future, the experiment will be conducted in more diverse environments.

Data Availability

The datasets are available from https://fairsharing.org/4298.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Hentschel and B. Wagner, "Autonomous robot navigation based on openstreetmap geodata," in *Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems*, pp. 1645–1650, IEEE, Funchal, Portugal, 19-22 September 2010.

[2] B. B. K. Ayawli, R. Chellali, A. Y. Appiah, and F. Kyeremeh, "An overview of nature-inspired, conventional, and hybrid methods of autonomous vehicle path planning," *Journal of Advanced Transportation*, vol. 2018, pp. 1–27, 2018.

[3] T. Ort, K. Murthy, R. Banerjee et al., "Maplite: autonomous intersection navigation without a detailed prior map," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 556–563, 2020.

[4] M. Á. de Miguel, F. García, and J. M. Armingol, "Improved lidar probabilistic localization for autonomous vehicles using gnss," *Sensors*, vol. 20, no. 11, p. 3145, 2020.

[5] Z. Wang and A. Lambert, "A low-cost consistent vehicle localization based on interval constraint propagation," *Journal of Advanced Transportation*, vol. 2018, pp. 1–15, 2018.

[6] M. Osman, A. Hussein, and A. Al-Kaff, "Intelligent vehicles localization approaches between estimation and information: a review," in *Proceedings of the 2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 1–8, IEEE, Cairo, Egypt, 04-06 September 2019.

[7] E. C. Pereira, D. A. Lima, and A. C. Victorino, "Autonomous vehicle global navigation approach associating sensor based control and digital maps," in *Proceedings of the 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, pp. 2404–2409, IEEE, Bali, Indonesia, 05-10 December 2014.

[8] D. A. De Lima and A. C. Victorino, "Sensor-based control with digital maps association for global navigation: a real application for autonomous vehicles," in *Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 1791–1796, IEEE, Gran Canaria, Spain, 15-18 September 2015.

[9] Q. Li, L. Chen, Q. Zhu, M. Li, Q. Zhang, and S. S. Ge, "Intersection detection and recognition for autonomous urban driving using a virtual cylindrical scanner," *IET Intelligent Transport Systems*, vol. 8, no. 3, pp. 244–254, 2014.

[10] Y. Yi, L. Hao, Z. Hao, S. Songtian, L. Ningyi, and S. Wenjie, "Intersection scan model and probability inference for vision based small-scale urban intersection detection," in *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1393–1398, IEEE, Los Angeles, CA, USA, 11-14 June 2017.

[11] A. L. Ballardini, H. Saz, S. Carrasco Limeros et al., "Urban intersection classification: a comparative analysis," *Sensors*, vol. 21, no. 18, p. 6269, 2021.

[12] M. Hnewa and H. Radha, "Object detection under rainy conditions for autonomous vehicles: a review of state-of-the-art and emerging techniques," *IEEE Signal Processing Magazine*, vol. 38, no. 1, pp. 53–67, 2021.

[13] Z. Liu, Y. Cai, H. Wang et al., "Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6640–6653, 2021.

[14] A. Bell, T. Mantecón, C. Díaz, C. R. del Blanco, F. Jaureguizar, and N. García, "A novel system for nighttime vehicle detection based on foveal classifiers with real-time performance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, p. 5421, 2022.

[15] Y. Cai, T. Luan, H. Gao et al., "Yolov4-5d: an effective and efficient object detector for autonomous driving," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–13, 2021.

[16] Z. Yao and X. Chen, "Efficient lane detection technique based on lightweight attention deep neural network," *Journal of Advanced Transportation*, vol. 2022, pp. 1–13.

[17] T. Almeida, B. Lourenço, and V. Santos, "Road detection based on simultaneous deep learning approaches," *Robotics and Autonomous Systems*, vol. 133, p. 103605, 2020.

[18] L. Wang, J. Wang, X. Wang, and Y. Zhang, "3d-lidar based branch estimation and intersection location for autonomous vehicles," in *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1440–1445, IEEE, Los Angeles, CA, USA, 11-14 June 2017.

[19] D. Bhatt, D. Sodhi, A. Pal, V. Balasubramanian, and M. Krishna, "Have i reached the intersection: a deep learning-based approach for intersection detection from monocular cameras," in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4495–4500, IEEE, Vancouver, BC, Canada, 24-28 September 2017.

[20] A. L. Ballardini, A. H. Saz, and M. Á. Sotelo, "Model guided road intersection classification," in *Proceedings of the 2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 703–709, IEEE, Nagoya, Japan, 11-17 July 2021.

[21] F. Yan, K. Wang, B. Zou, L. Tang, W. Li, and C. Lv, "Lidar-based multi-task road perception network for autonomous vehicles," *IEEE Access*, vol. 8, pp. 86753–86764, 2020.

[22] J. B. Chipka and M. Campbell, "Estimation and navigation methods with limited information for autonomous urban driving," *Engineering Reports*, vol. 1, no. 4, p. e12054, 2019.

[23] M. Kim, J. Ahn, and J. Park, "Global planning method for visiting roads with parking spaces in priority using rural postman problem," in *Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 4184–4189, IEEE, Auckland, New Zealand, 27-30 October 2019.

[24] D. Wu, M. Liao, W. Zhang, and X. Wang, "Yolop: you only look once for panoptic driving perception," 2021, http://arxiv.org/abs/2108/11250.

[25] Y. Qian, J. M. Dolan, and M. Yang, "Dlt-net: joint detection of drivable areas, lane lines, and traffic objects," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4670–4679, 2020.

[26] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: real-time joint semantic reasoning for autonomous driving," in *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1013–1020, IEEE, Changshu, China, 26-30 June 2018.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, http://arxiv.org/abs/1409.1556.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: a large-scale hierarchical image database," in *Proceedings of the 2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, IEEE, Miami, FL, USA, 20-25 June 2009.

[29] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, San Juan, PR, USA, 17-19 June 1997.

[30] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *Proceedings of the International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, September 18-22, 2022.

[31] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[32] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, Honolulu, HI, USA, 21-26 July 2017.

[33] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression,"vol. 3, pp. 850–855, in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, IEEE, Hong Kong, China, 20-24 August 2006.

[34] J. Zhang and S. Singh, "Loam: lidar odometry and mapping in real-time robotics: science and systems," *Robotics*, vol. 2, no. 9, 2014.

[35] A. Bréhéret, "Pixel annotation tool," August 2017, https://github.com/abreheret/PixelAnnotationTool.

[36] H. Mouhagir, R. Talj, V. Cherfaoui, F. Aioun, and F. Guillemard, "Evidential-based approach for trajectory planning with tentacles, for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3485–3496, 2020.

[37] V. Olunloyo and M. Ayomoh, "Autonomous mobile robot navigation using hybrid virtual force field concept," *European Journal of Scientific Research*, vol. 31, no. 2, pp. 204–228, 2009.

[38] J. Ahn, M. Kim, and J. Park, "Vision-based autonomous driving for unstructured environments using imitation learning," 2022, http://arxiv.org/abs/2202.10002.

[39] J. Ahn, S. Shin, M. Kim, and J. Park, "Accurate path tracking by adjusting look-ahead point in pure pursuit method," *International Journal of Automotive Technology*, vol. 22, no. 1, pp. 119–129, 2021.