Vehicle Path Planning with Multicloud Computation Services

Po-Tong Wang[1], Shao-Yu Lin[2], Jia-Shing Sheu[2,*]

[1]Department of Electrical Engineering, Lunghwa University of Science and Technology, Taoyuan, Taiwan

[2]Department of Computer Science, National Taipei University of Education, Taipei, Taiwan

Received 21 February 2021; received in revised form 09 August 2021; accepted 10 August 2021

DOI: https://doi.org/10.46604/aiti.2021.7192

Abstract

With the development of artificial intelligence, public cloud service platforms have begun to provide common pretrained object recognition models for public use. In this study, a dynamic vehicle path-planning system is developed, which uses several general pretrained cloud models to detect obstacles and calculate the navigation area. The Euclidean distance and the inequality based on the detected marker box data are used for vehicle path planning. Experimental results show that the proposed method can effectively identify the driving area and plan a safe route. The proposed method integrates the bounding box information provided by multiple cloud object detection services to detect navigable areas and plan routes. The time required for cloud-based obstacle identification is 2 s per frame, and the time required for feasible area detection and action planning is 0.001 s per frame. In the experiments, the robot that uses the proposed navigation method can plan routes successfully.

Keywords: computer vision, scene recognition, cloud computing, object detection

1. Introduction

Although humans can walk small distances, walking long distances is exhausting and time-consuming. Therefore, bicycles, motorcycles, and cars were invented over time, and their continued evolution has made movement convenient and safe. The focus of vehicle development has now shifted to the production of low-pollution electric vehicles, such as electric bicycles, balance bikes, and skateboards. Autonomous driving can decrease the burden on human drivers, reduce road congestion, and improve transportation safety [1]. Planning a safe pathway is the focus of autonomous driving. The information regarding the surrounding environment is collected to plan the next action. To prevent collisions, the environment and moving objects are monitored in real time for determining appropriate responses. However, the implementation of autonomous driving technologies is difficult. Current autonomous vehicles use multiple ultrasonic or optical radars for detecting surrounding objects to create high-quality three-dimensional (3D) models at night and during the day; however, these sensors are expensive. Furthermore, inclement weather severely affects the performance of the aforementioned sensors.

Machine vision technology is used in daily life applications, such as the smart face unlock feature in smartphones, instant text translation, and automatic checkout in stores. The capabilities of hardware equipment, such as high-image-quality cameras, vision processors, and 5G networks, are continually improving. Commercially available visual models can accurately perform face, object, and text recognition. This study realizes the recognition of different objects in real-time images by using application programming interfaces (APIs), i.e., Google Cloud Vision, Amazon Rekognition, and Azure Computer Vision. The obtained object information is regarded as the current environmental conditions when determining the walking area. This study uses available resources and machine vision to develop a low-cost and fast method for dynamic path discrimination.

* Corresponding author. E-mail address: jiashing@tea.ntue.edu.tw
Tel.: +886-9-38397255; Fax: +886-2-27375457

The remainder of this study is organized as follows. Section 2 describes the relevant literature. Section 3 describes the system architecture. Section 4 presents a description of the experiments and the experiment results, and section 5 provides the conclusions of this study.

2. Literature Review and Methodology

Object detection is a technique for classifying objects in an image, and object detection technology has evolved considerably. LeCun et al. [2] proposed the LeNet image recognition network, which was subsequently modified by Krizhevsky et al. [3] into AlexNet. The addition of the rectified linear unit and DropOut nonlinear activation functions to LeNet considerably improved its image recognition rate. Thus, machine vision has evolved rapidly. He et al. [4] proposed residual network architecture to solve the problem of overfitting. Huang et al. [5] and Wang et al. [6] made subtle changes and proposed DenseNet and CSPNet, respectively. Similar concepts of CSPNet were used to design novel architectures that effectively enhance network identification capabilities [5-6]. Howard et al. [7] proposed lightweight network architecture to increase the processor calculation speed and solve the problem of slow network operations in mobile devices and embedded learning.

The aforementioned network architectures can be implemented to obtain a backbone network for the rapid learning of image features. Many object recognition networks use such backbone networks for image feature extraction. Various calculation methods are then incorporated into the network to learn the category and location of images [8]. Ren et al. [9] developed a two-stage object recognition, i.e., faster region based convolutional neural networks (Faster RCNN), with a high recognition rate by using visual geometry group (VGG) as the backbone network and the region proposal network. The aforementioned architecture was also used in an object recognition network [10]; however, in contrast to Faster RCNN, the single-shot detector (SSD) object recognition network performs one-stage identification in real time.

Numerous sophisticated, fast, and automated object detection networks have been proposed. The initially proposed object detection networks such as "you only look once (YOLO)" do not use artificial anchor frames [11]. Law et al. [12] proposed CornerNet, which is a novel anchorless frame network, for self-learning object detection. Duan et al. [13] and Tian et al. [14] subsequently improved CornerNet and achieved the same results as Law et al. [12] without relying on anchor frames. Tan et al. [15] developed EfficientDet and achieved up to 53.7 average precision (AP) with current one-stage object detectors.

Achieving simultaneous localization and mapping is critical for developing automated machinery. Light detection and ranging technology can be used to sense the surrounding environment and dynamically avoid moving objects, such as crowds and vehicles on the road [16-17]. Furthermore, RGBD-based visual sensors were used to establish real-time environmental images and motion paths for augmented reality (AR), virtual reality (VR), and unmanned aerial vehicle (UAV) positioning [18]. A vision-based deep learning network can be used to detect the environment for conducting motion detection, walkable area detection, and motion planning [19]. Obstacle detection can also performed by object detection, and the technique algorithm will be used to mark walkable areas and relative coordinates [20-21].

Avoiding obstacles is a crucial ability for autonomous mobile robots, which must plan suitable movement routes. When moving from the current location to a target location, the most efficient movement route is the shortest path without any restrictions. Dijkstra's algorithm and the A* algorithm are common methods for determining the shortest path. Dijkstra's algorithm is similar to the breadth-first search method; it searches for the shortest distance node outward from the current coordinate point and continues the search until the target point is found. The A* algorithm combines the speed priority and Dijkstra methods. The costs from the starting point to the node and from the node to the target point are added and used as the node's search cost, and the path with the lowest cost is searched for outward from the starting point. Using the method, the A* algorithm can plan the optimal path while avoiding obstacles. The obstacle avoidance problem is also called the velocity obstacle (VO) problem. If a robot collides with another robot when maintaining its current speed, the set of all collision events comprises the VO [22].

3. System Structure

This study proposes a real-time path-planning system based on neural networks for unmanned vehicles. The image obtained by an embedded camera is used as the input, and multiple public pretrained neural networks are used to increase the amount of information available on obstacle positions. The drivable area is determined from the blank areas between multiple objects, and safe paths are planned in drivable areas. Fig. 1 displays the architecture of the proposed system. Three steps are used in this study to plan the path of a self-propelled vehicle: object detection (Fig. 2), path planning (Fig. 3), and movement control.

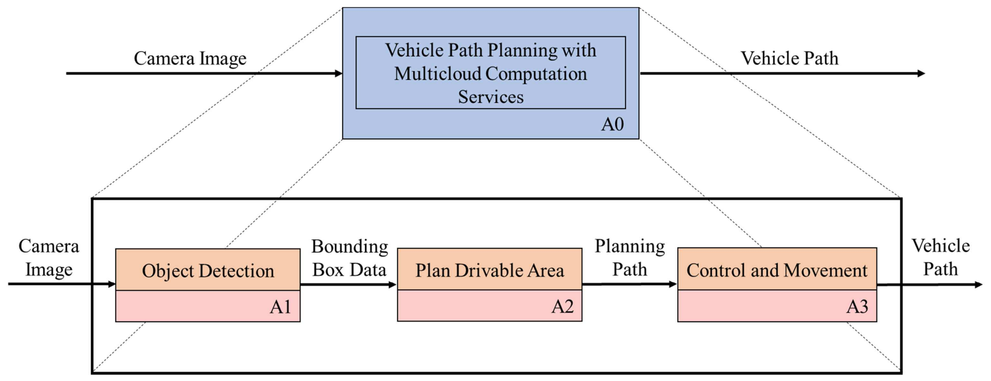


Fig. 1 Architecture of the proposed system

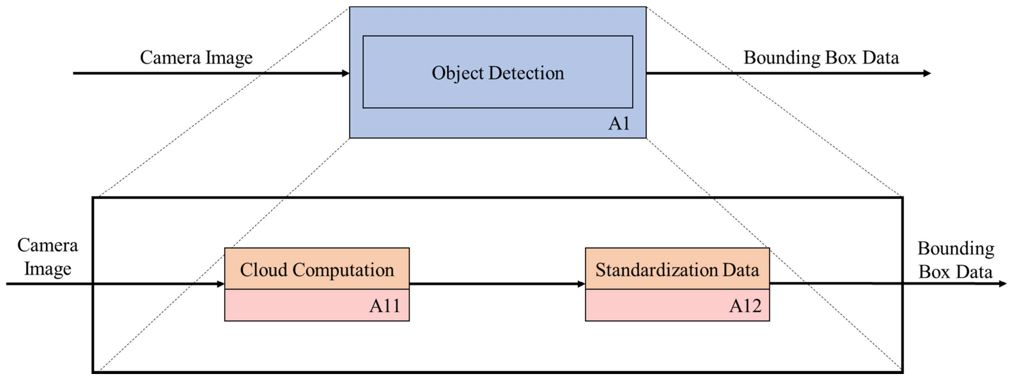


Fig. 2 Framework of cloud object detection

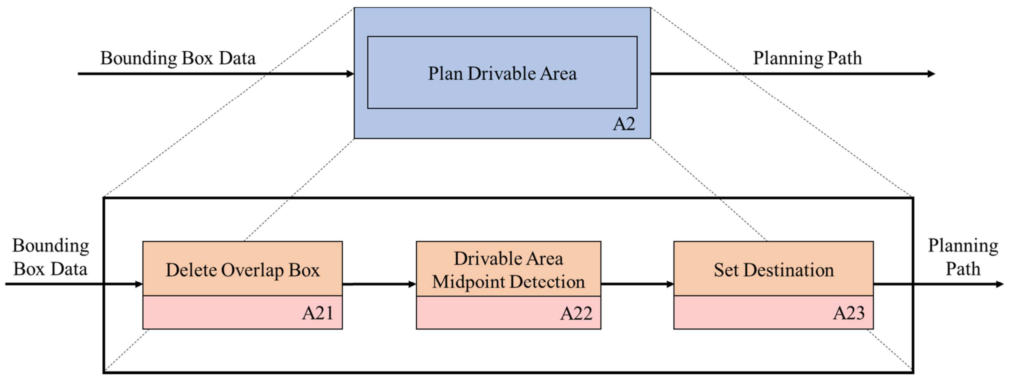


Fig. 3 Framework of path planning

3.1. Cloud object detection

The purpose of object detection is to detect the locations of obstacles and avoid them. An object detector is composed of a backbone, neck, and head. The backbone is a network for obtaining image features, the neck fuses the feature maps from various layers, and the head is used for classification and localization. The model architecture of an object detector is displayed in Fig. 4.

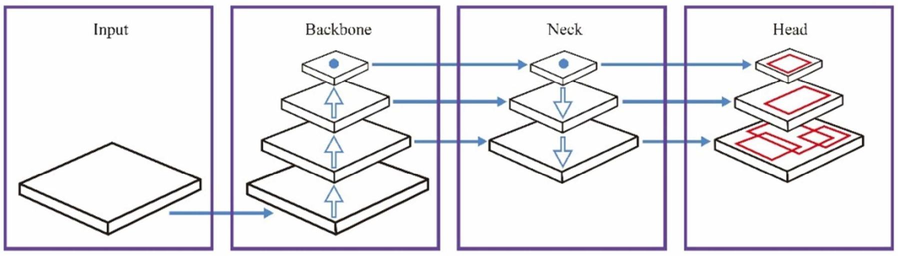


Fig. 4 Model architecture of an object detector

The pretrained networks provided by Google Cloud Platform (GCP), Microsoft Azure, and Amazon Web Service (AWS) are used to detect objects in real-time images and store the data from each platform in the following order: object class name, confidence score, bounding box top boundary value, bounding box bottom boundary value, bounding box left boundary value, and bounding box right boundary value.

3.2. Path planning for a vehicle

3.2.1. Deletion of the overlapped box

First, the coordinates of the object bounding box are determined. Next, whether two object bounding boxes overlap is determined. If an overlap is discovered, the bounding box with the higher bottom boundary value is removed. The left and right boundary values of objects A and B are set as (A_L, A_R) and (B_L, B_R), respectively. If the bounding boxes of A and B do not overlap, they must satisfy the following Eqs. (1) and (2).

$$B_L < B_R < A_L < A_R \tag{1}$$

$$A_L < A_R < B_L < B_R \tag{2}$$

3.2.2. Drivable area detection

After removing the overlapping bounding box, the blank area between any two nonoverlapping objects is considered the movement area. The distance between all objects can be calculated from their bounding box coordinates. According to an axiom in the Euclidean geometry system, the distance D between any two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ can be calculated using Eq. (3). The shortest distance between the bottom endpoints of two bounding boxes, that is, the bottom line of the drivable area, is calculated as follows:

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3}$$

3.2.3. Set destination

The drivable area should be clear and not blocked by intermediate objects. A and B are set as nonadjacent objects, and A is to the left of B. The four endpoint coordinates of the bottom boundaries of the two objects are denoted as $AL(x_1, y_1)$, $AR(x_2, y_2)$, $BL(x_1, y_1)$, and $BR(x_2, y_2)$. When N objects exist between A and B, the bottom boundary endpoint coordinates of any object can be denoted as $OL(x_1, y_1)$ or $OR(x_2, y_2)$. If the area is not blocked, Eqs. (4) and (5) are satisfied. When the denominator in Eqs. (4) and (5) is 0, the bounding boxes of the two objects overlap. This overlap is eliminated using Eq. (1) or (2).

$$\frac{|Ax_2 - Bx_1|}{|Ay_2 - By_1|} < \frac{|Ax_2 - Ox_1|}{|Ay_2 - Oy_1|} \tag{4}$$

$$\frac{|Ax_2 - Bx_1|}{|Ay_2 - By_1|} < \frac{|Ax_2 - Ox_2|}{|Ay_2 - Oy_2|} \tag{5}$$

The center point of a robotic car is set as the midpoint of the bottom boundary of the image. The center point of each drivable area can be calculated from the endpoint coordinates of the bottom line of the drivable area. Assuming that the two endpoints at the bottom of the drivable area are $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$, the midpoint coordinates of the drivable area, namely $P_m(x_m, y_m)$, can be calculated using Eq. (6). Eq. (3) is used to calculate the distance between the midpoint of all drivable areas and the center point of the robotic car. The midpoint with the shortest distance to the center point is set as the waypoint, and the straight line from the center point to the waypoint represents the planned route.

$$P_m(x_m, y_m) = (\frac{|x_1 - x_2|}{2}, \frac{|y_1 - y_2|}{2}) \tag{6}$$

3.3. Movement control

The coordinates of the center point of the robotic car and the waypoint can be used to calculate the offset angle between the travel direction of the robotic car and the waypoint. First, Eq. (3) is used to calculate the distance between the car and waypoint. Then, the cosine value of the angle between the planned path and the vertical line is calculated. The angle converted by the cosine value is the offset angle. If the offset angle does not exceed the preset threshold, a forward command is issued to allow the car to move forward. However, if the offset angle exceeds the preset threshold, a turn-left or turn-right command is issued for appropriately controlling the direction of car movement.

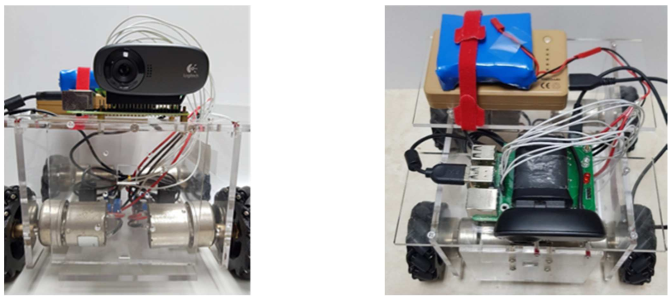
4. Experimental Results

Table 1 lists the experimental equipment used in this study. The Raspberry Pi is a single-chip computer developed by the Raspberry Pi foundation for improving students' understanding of computing science. The experiment uses Docker to construct an image file with the official Raspberry Pi operating system and the Python 3 language for research.

Table 1 Experimental equipment used in this study

Components	Specification
Operating system	Raspbian
Central processing unit	ARM Cortex-A72
Random access memory	4 GB (LPDDR4)
Camera	Logitech C310

Fig. 5 displays a self-propelled vehicle equipped with a Raspberry Pi 4 computer and complementary metal oxide semiconductor (CMOS) lens assembly for capturing environmental images. The Raspberry Pi 4 has a 40-pin universal input and output that can be used for external screens and sensors. The Raspberry Pi 4 is connected to a power source motor for controlling the movement of a robotic car. A Logitech C310 fixed-focus lens is used to obtain a 5-million-pixel image with a size of 1280 × 960. This lens is connected to the Raspberry Pi 4 through a universal serial bus (USB) interface. Raspberry Pi 4 has a rated power of 5 V/3 A. A mobile power bank is used as a power source through USB-C to provide a maximum output of 5 V/2.1 A to the Raspberry Pi 4.



(a) Front view of the vehicle (b) Top view of the vehicle

Fig. 5 The robotic car equipped with Raspberry Pi 4 and CMOS camera

4.1. Drivable area detection

The Raspberry Pi 4 and a streaming service are started. The streaming service is based on real time streaming protocol (RTSP). Streaming images are sent to the cloud through cloud platforms, and APIs are used for object identification. The object detection services used in this study are provided by Google Cloud Vision, Amazon Rekognition, and Azure Computer Vision cloud platforms to obtain the position coordinates, category names, and confidence scores for the objects in an image. Obstacles are identified by using multiple clouds to increase the amount of information because insufficient information would have negatively affected the suitability of the planned path. Moreover, accurate object class names and confidence scores are not required because the aim is to obtain information on surrounding obstacles. Fig. 6 displays the results of object marking when using multiple cloud platforms, and the marked boxes of the same color represent the detection results obtained from the same platform. Here, the GCP vision is presented by red color and the Azure vision is presented by blue color.

The program for drivable area detection is written in the Python 3 language. First, the overlapping object bounding boxes are removed. Then, the drivable area is detected and the waypoint is set. Fig. 7 displays the result obtained after subjecting Fig. 6 to drivable area detection. The green lines represent the bottom lines of multiple drivable areas.

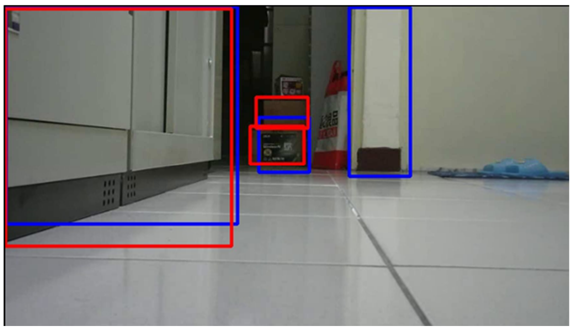


Fig. 6 Results of the comprehensive mark bounding box of cloud object detection

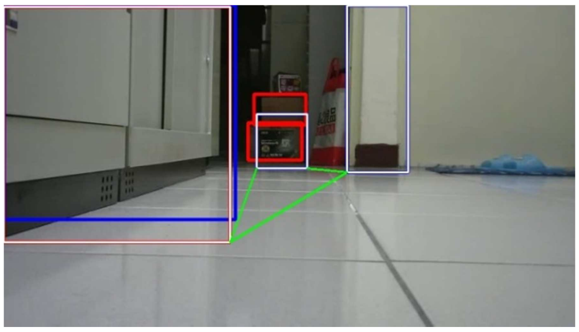


Fig. 7 Schematic of drivable area detection

4.2. Results and analysis

Fig. 8 depicts the two areas used in the experiment. Fig. 8(a) displays a narrow walking passage in a laboratory. This image indicates that the exit of the aisle is on the left; therefore, the robotic car should turn to the left at the end of this aisle to avoid collision with the stacked boxes. Fig. 8(b) displays a wide laboratory area with two exits. Fig. 9 displays the legend for the different bounding box colors in Figs. 10 and 11, which depict the continuous images obtained when the robot car moves according to the system instructions in the narrow and wide laboratory areas, respectively. In these images, different colors are used to represent the obstacles identified using different platforms. The yellow lines in Figs. 10 and 11 indicate the real-time planned path for each image. The images in the aforementioned figures prove that visual recognition could be used to obtain a bounding box for path planning. Thus, cloud computing facilitates real-time dynamic path planning.

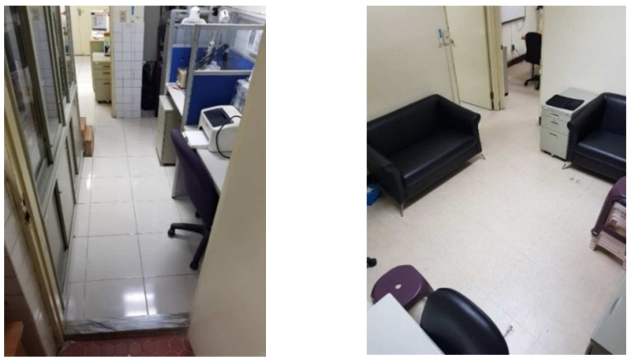

(a) Narrow area (b) Wide area

Fig. 8 Panoramic images of the experimental areas

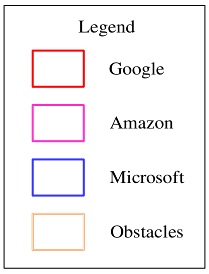
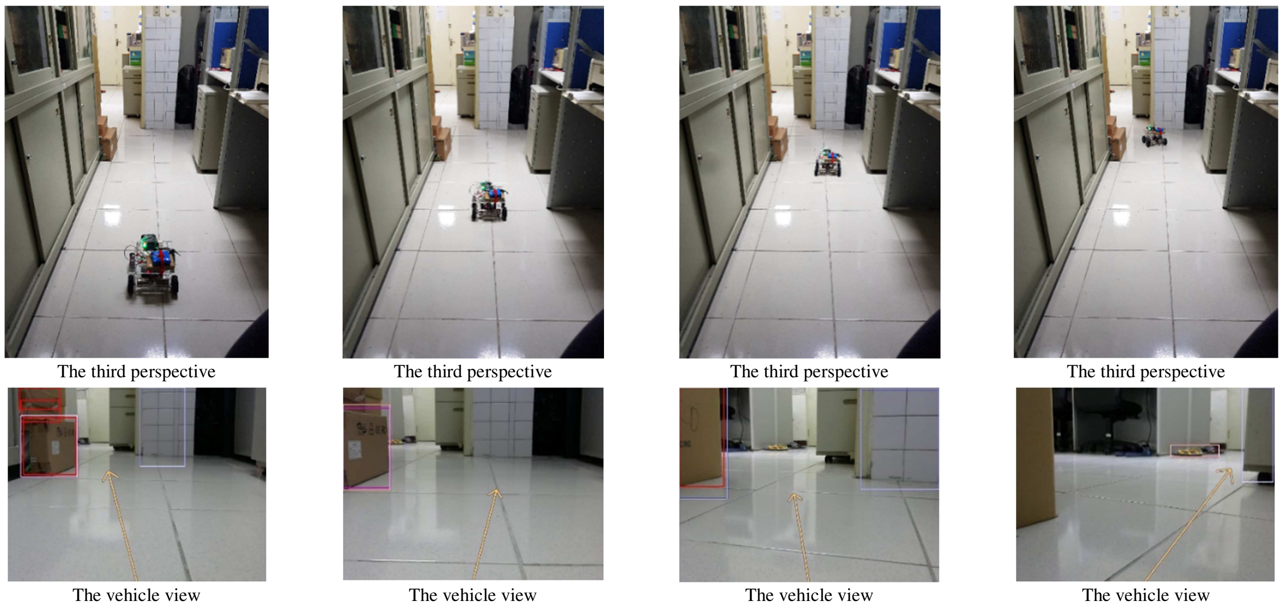
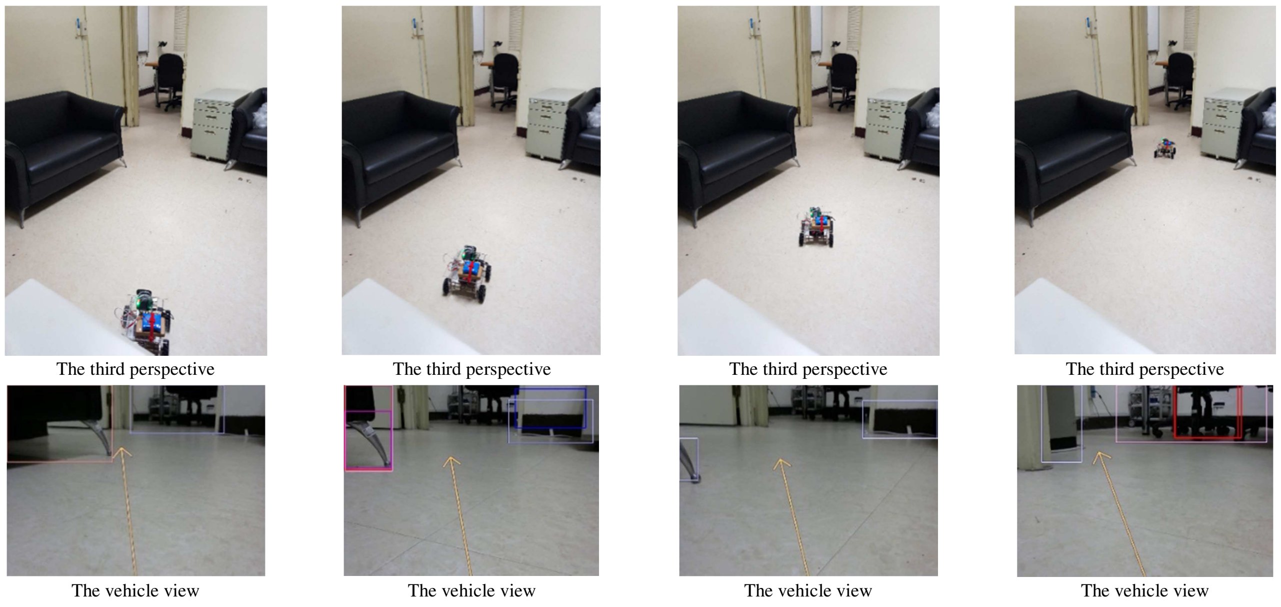


Fig. 9 Legend of the bounding box colors



(a) At 5 s (b) At 15 s (c) At 25 s (d) At 35 s

Fig. 10 Real-time images of the path planning and movement of the robotic car in the narrow laboratory area



(a) At 5 s (b) At 15 s (c) At 25 s (d) At 35 s

Fig. 11 Real-time images of the path planning and movement of the robotic car in the wide laboratory area

All the obstacles on the ground are marked with white bounding boxes because the correct labeling of the object category is not crucial. Moreover, the threshold to limit the object discrimination rate is obtained. Multiple travel zones are detected by using a self-developed algorithm, and path planning is completed rapidly by using the obtained obstacle information. Detailed experimental data are listed in Tables 2 and 3.

Table 2 Detailed experimental results for cloud object detection

Figure	Path planning time (s)	Cloud platform	Class name	Detection confidence score	Left boundary value	Right boundary value	Top boundary value	Bottom boundary value
10(a)	0.001	GCP	Packaged goods	0.76	60	259	0	109
		AWS	Box	0.95	66	345	139	444
		GCP	Packaged goods	0.69	67	261	64	171
		GCP	Shipping box	0.61	79	340	150	432
		Azure	Wall	0.35	656	878	0	391
10(b)	0.02	GCP	Shipping box	0.76	0	230	85	511
		AWS	Box	0.56	0	232	103	500
10(c)	0.001	Azure	Box	0.68	0	238	0	549
		GCP	Shipping box	0.71	2	221	0	489
		Azure	Wall	0.56	896	1280	10	503
10(d)	0.001	GCP	Shoe	0.62	765	1004	285	344
		Azure	Cabinet	0.36	1118	1272	0	468
11(a)	0.001	GCP	Furniture	0.59	3	538	0	386
		Azure	Chair	0.38	628	1073	0	249
11(b)	0.001	GCP	Person	0.73	2	243	2	427
		AWS	High heel	0.55	5	240	133	418
		GCP	Person	0.71	10	254	1	434
		Azure	Chair	0.51	829	1253	74	292
		Azure	Wall	0.36	862	1216	20	216
11(c)	0.001	Azure	Chair	0.45	2	94	274	481
		Azure	Chair	0.44	916	1280	78	267
11(d)	0.001	Azure	Chair	0.38	124	323	0	387
		AWS	Airplane	0.82	497	1248	0	288
		GCP	Person	0.69	787	1105	0	264
		GCP	Home appliance	0.54	791	1121	0	266

Table 3 Average time required for cloud service object identification

Google Cloud Vision API	0.356 (s)
Microsoft Azure Computer Vision API	1.042 (s)
Amazon Web Services Rekognition API	2.098 (s)

5. Conclusions

This study developed a method in which the bounding box information returned by multiple cloud object detection services is integrated to detect the drivable area and plan a movement route. The experimental results reveal that robotic cars can perform appropriate path planning by using the developed method. The obtained images indicate that the planned route using the developed method is safe. The goals of this research were to reduce the cost of developing a collision-avoidance system, the cost of computing, and the dependence on an object sensor. Cloud vision models are used to reduce the amount of computing in the designed system. The designed system is simple, has low cost, and contains only one lens. Routes are planned through simple mathematical operations that do not burden the developed real-time path-planning system. Latency is a critical real-time problem that should be considered when using cloud computing local area network signals because it affects system operation. The delay problem can be solved using next-generation high-speed networks that provide high transmission speeds with novel visual recognition models.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] P. H. Lin, C. Y. Lin, C. T. Hung, J. J. Chen, and J. M. Liang, “The Autonomous Shopping-Guide Robot in Cashier-Less Convenience Stores,” Proceedings of Engineering and Technology Innovation, vol. 14, pp. 9-15, January 2020.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, November 1998.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Communications of the ACM, vol. 60, no. 6, pp. 84-90, June 2017.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” IEEE Conference on Computer Vision and Pattern Recognition, June 2016, pp. 770-778.

[5] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” IEEE Conference on Computer Vision and Pattern Recognition, July 2017, pp. 2261-2269.

[6] C. Y. Wang, H. Y. M. Liao, Y. H. Wu, P. Y. Chen, J. W. Hsieh, and I. H. Yeh, “CSPNet: A New Backbone That Can Enhance Learning Capability of CNN,” IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, June 2020, pp. 1571-1580.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, et al., “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” https://arxiv.org/pdf/1704.04861.pdf, April 17, 2017.

[8] K. H. Shih, C. T. Chiu, J. A. Lin, and Y. Y. Bu, “Real-Time Object Detection with Reduced Region Proposal Network via Multi-Feature Concatenation,” IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 6, pp. 2164-2173, June 2020.

[9] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, June 2017.

[10] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, et al., “Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges,” IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 3, pp. 1341-1360, March 2021.

[11] W. He, Z. Huang, Z. Wei, C. Li, and B. Guo, “TF-YOLO: An Improved Incremental Network for Real-Time Object Detection,” Applied Sciences, vol. 9, no. 16, 3225, August 2019.

[12] H. Law and J. Deng, “CornerNet: Detecting Objects as Paired Keypoints,” European Conference on Computer Vision, September 2018, pp. 734-750.

[13] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, “CenterNet: Keypoint Triplets for Object Detection,” IEEE/CVF International Conference on Computer Vision, October 2019, pp. 6569-6578.

[14] Z. Tian, C. Shen, H. Chen, and T. He, “FCOS: Fully Convolutional One-Stage Object Detection,” IEEE/CVF International Conference on Computer Vision, October 2019, pp. 9626-9635.

[15] M. Tan, R. Pang, and Q. V. Le “EfficientDet: Scalable and Efficient Object Detection,” IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2020, pp. 10778-10787.

[16] J. Chen, W. Zhan, and M. Tomizuka, “Autonomous Driving Motion Planning with Constrained Iterative LQR,” IEEE Transactions on Intelligent Vehicles, vol. 4, no. 2, pp. 244-254, June 2019.

[17] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee, and D. Manocha, “PORCA: Modeling and Planning for Autonomous Driving Among Many Pedestrians,” IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3418-3425, October 2018.

[18] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System,” IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, May 2015.

[19] D. K. Dewangan and S. P. Sahu, “Deep Learning-Based Speed Bump Detection Model for Intelligent Vehicle System Using Raspberry Pi,” IEEE Sensors Journal, vol. 21, no. 3, pp. 3570-3578, February 2021.

[20] C. Shao, C. Zhang, Z. Fang, and G. Yang, “A Deep Learning-Based Semantic Filter for RANSAC-Based Fundamental Matrix Calculation and the ORB-SLAM System,” IEEE Access, vol. 8, pp. 3212-3223, 2019.

[21] J. S. Sheu and C. Y. Han, “Combining Cloud Computing and Artificial Intelligence Scene Recognition in Real-time Environment Image Planning Walkable Area,” Advances in Technology Innovation, vol. 5, no. 1, pp. 10-17, December 2019.

[22] T. Xu, S. Zhang, Z. Jiang, Z. Liu, and H. Cheng, “Collision Avoidance of High-Speed Obstacles for Mobile Robots via Maximum-Speed Aware Velocity Obstacle Method,” IEEE Access, vol. 8, pp.138493-138507, 2020.