Development of Teaching Materials on Autonomous Driving of robots using Deep Learning

Norito NIKI , Justin Lee , Koichi MURO, Yuta FUKUSHIMA and Yuichi HASUDA*
Department of Information and Electronic Engineering,
Teikyo University Tochigi prefecture Utsunomiya city
toyosatodai 1-1, Japan
* hasuda@ics.teikyo-u.ac.jp

Abstract

Automatic driving technology, where systems control cognitive processes, judgments, and operations performed by humans during driving, is rapidly becoming more prevalent and familiar to our world. The system drives automatically by analyzing the surrounding information collected by cameras and sensors. Deep learning is essential for collision avoidance and safe autonomous driving. In order for university students to experience these technologies in their lectures, it is necessary to introduce teaching materials on autonomous using simple models. In this study, we developed teaching materials on autonomous driving of robots using Deep Learning with AI, then verified its functionality in lectures and contests.

In this study, we tried to develop teaching materials on autonomous driving of robots using Deep Learning. Using Raspberry Pi 4 as the controller and implementing Deep Learning programming with Python, the prototype teaching material was able to recognize and avoid the two types of obstacles used while driving through the obstacle course. The prototype was able to accurately recognize the randomly placed obstacles on the 3m x 3m obstacle course and avoid them by turning left or right without making any contact with the obstacles.

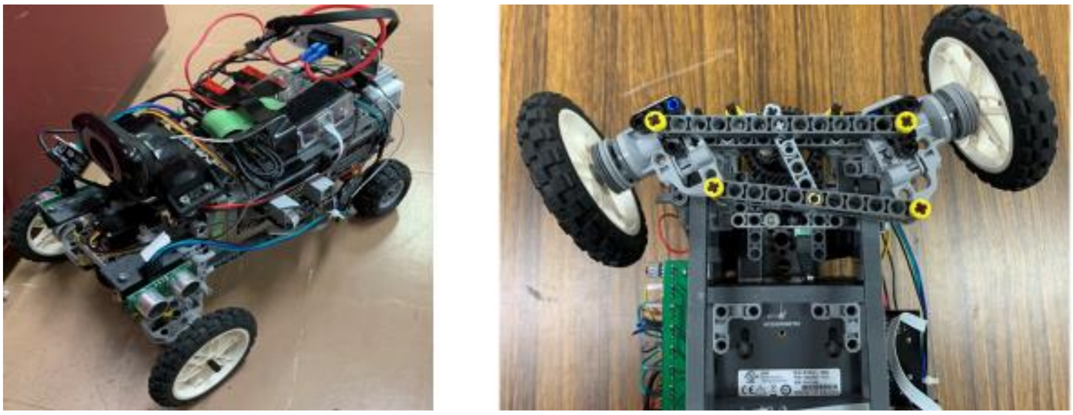
Keywords: Deep Learning, Autonomous Driving ,Teaching Material, AI, robots

1. Introduction

Automatic driving technology, where systems control cognitive processes, judgments, and operations performed by humans during driving, is rapidly becoming more prevalent and familiar to our world[1.2]. The system drives automatically by analyzing the surrounding information collected by cameras and sensors. [3-5] · In Japan, under the guidance of the Ministry of Economy, Trade, and Industry, the three major car manufacturers, Toyota, Nissan, and Honda conducted public road demonstrations and experiments showcasing their autonomous driving technologies back in 2013. Deep learning is essential for collision avoidance and safe autonomous driving. In order for university students to experience these technologies in their lectures, it is necessary to introduce teaching materials on autonomous using simple models. Therefore, in this study, we developed teaching materials on autonomous driving of robots using Deep Learning with AI, then verified its functionality in lectures and contests.

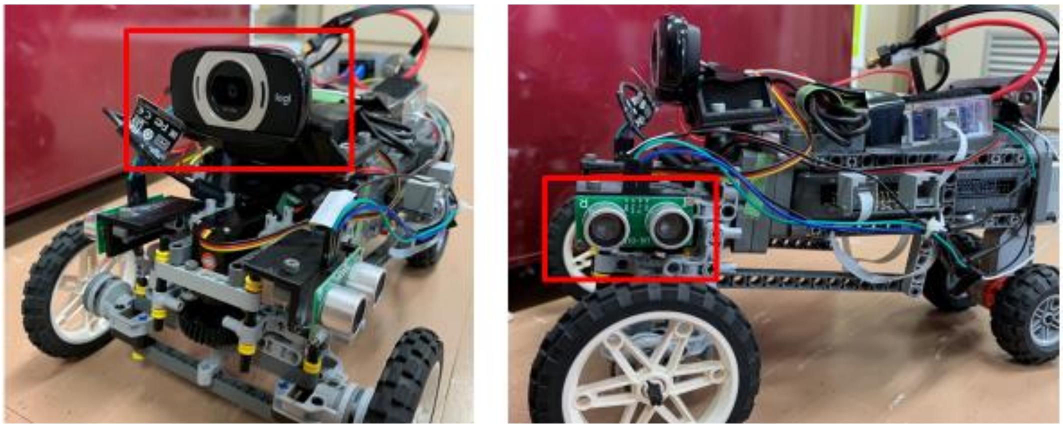
2. Prototype of Autonomous Driving Model teaching material using myRIO

Figure 1 shows the prototype of the robot used as teaching material. The design incorporates a steering mechanism on the front wheels and involves analyzing images sent from the camera located at the front of the robot to the myRIO controller. The ultimate objective is for the robot to identify the shapes and colors of the obstacles, enabling the robot to avoid them autonomously. [6] (Refer to Figure 2). Ultrasonic sensors are installed on both sides of the vehicle's main frame to prevent collisions with walls.



(a) Overview of the model car (b) Steering of the model car

Fig.1: Overview of the model car and the steering installed on the front wheels



(a) Camera in front of the model car (b) Ultrasonic sensor on the side

Fig.2: Camera and Ultrasonic sensor installed to the model car

As shown in Figure 3, the obstacle models are square columns with dimensions of 50mm width, 50mm length, and 100mm height, in green and red. Images of these obstacles were pre-trained and programmed using LabVIEW to avoids obstacles by turning left when encountering a green obstacle and turning right when encountering a red obstacle[7] (Refer to Figure 4).

The prototype teaching materials were able to recognize and avoid two types of obstacles using Deep Learning while driving through the obstacle course. The robot was able to recognize obstacles and avoid collisions using both myRIO and LabVIEW, albeit at a low speed[8]. However, the production cost of a single unit amounted to approximately 200,000yen, which is relatively high. The vehicle weight also exceeded 2.5kg. We concluded that it is necessary to supply the teaching materials inexpensively.

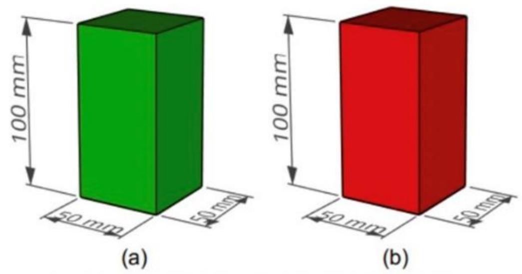


Fig.3: The two types of obstacle models used on the course

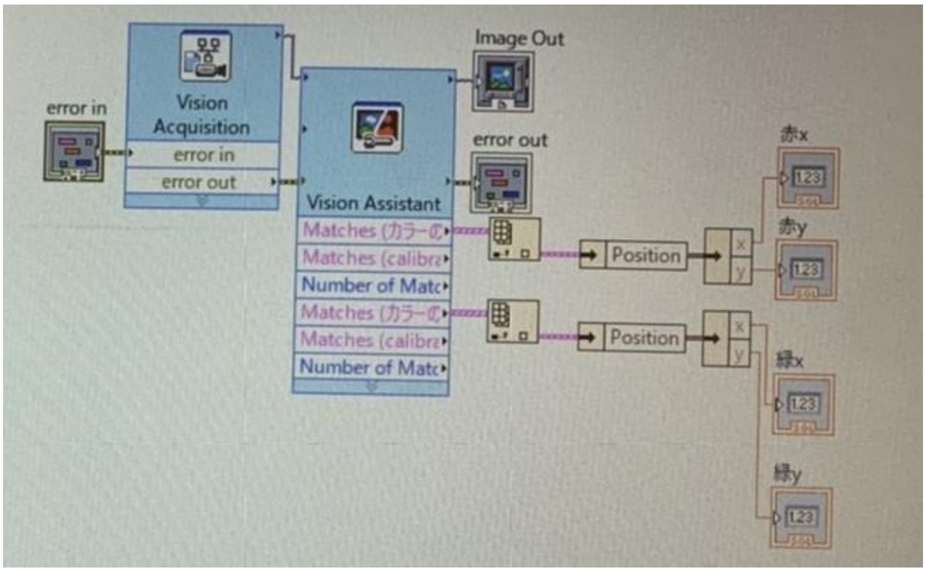


Fig.4: LabVIEW used in programs

3. Development of Teaching materials on Autonomous Driving Robot using RaspberyyPi4

3.1 Overview of the Autonomous Driving robot

In order to reduce the overall production cost of the teaching material, Raspberry Pi 4 was used as the controller (Refer to Figure 5). The Raspberry Pi 4 analyzes the images sent from the camera at the front of the robot. Based on the analysis, the Raspberry Pi 4 determines the positions and colors of the obstacles and enables the robot to avoid them according to their characteristics. This approach allows for cost reduction while maintaining the functionality required for obstacle avoidance.

By using Raspberry Pi 4 as the controller and optimizing the design, it was possible to reduce the production cost for one unit to approximately 30,000 yen, which is a cost reduction of about six times compared to the original prototype. The motor and steering are also reduced to two. Furthermore, instead of attaching ultrasonic sensors to both sides of the car’s body, only one is attached at the left side of the car to prevent collisions with walls. The weight of the robot has been reduced from myRIO-type′ s 2.5kg to 0.8kg, which is less than one-third of the original weight.

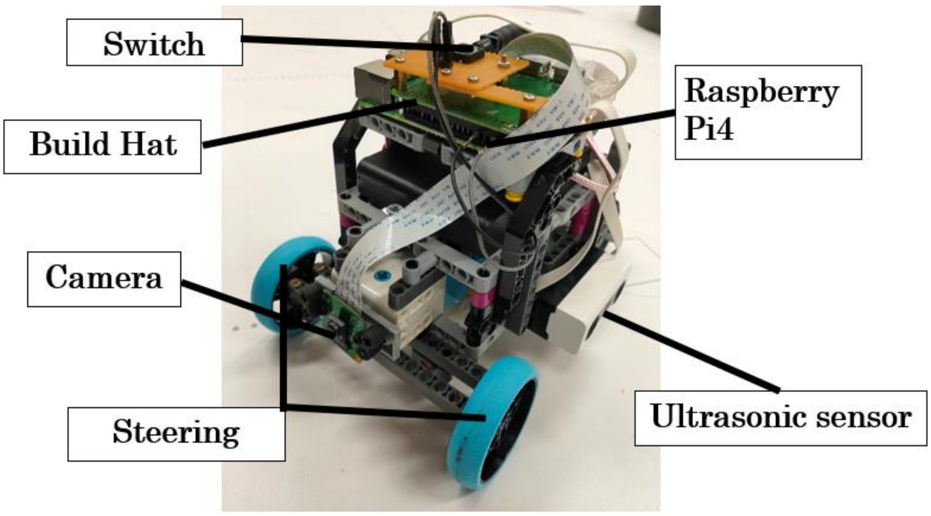


Fig.5: Autonomous driving robot teaching material

3.2 Formula and Equation

By using OpenCV, the robot was pre-trained using the obstacle model shown in Figure 3. The robot is programmed to avoid obstacles by turning left when encountering a green obstacle and turning right when encountering a red obstacle. The image recognition program was created using Python's OpenCV to process the images sent from the camera. The robot was able to obtain information about the obstacles by performing binary thresholding on the image with red and green colors. Figure 6 shows the program for binary thresholding.

```
def binary_threshold(frame, color='red'):
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    if color == 'red':
        lower_red = np.array(lower_R)
        upper_red = np.array(upper_R)
        lower_red2 = np.array(lower_R2)
        upper_red2 = np.array(upper_R2)

        mask1 = cv2.inRange(hsv, lower_red, upper_red)
        mask2 = cv2.inRange(hsv, lower_red2, upper_red2)

        mask = mask1 + mask2
    else:  # green
        lower_green = np.array(lower_G)
        upper_green = np.array(upper_G)
        mask = cv2.inRange(hsv, lower_green, upper_green)

    return mask
```

Fig.6: Program for binary thresholding

When binary thresholding is being performed, noise due to light reflection, which could hinder the process. Therefore, additional processing to remove objects within a certain area is added (Figure 7).

```
def remove_small_objects(mask, min_size):
    num_labels, labels, stats, _ = cv2.connectedComponentsWithStats(mask, connectivity=8)
    for i in range(1, num_labels):
        if stats[i][-1] < min_size:
            mask[labels == i] = 0
    return mask
```

Fig.7: Program to remove small area

The object's center coordinates, area, and minimum value of Y-axis is calculated from the binary threshold image. It is possible for multiple contours to exist within the image. Therefore, information is obtained from the object with the largest area from the binary threshold image (Figure 8).

```python
def get_largest_object_center(mask):
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
                                   cv2.CHAIN_APPROX_SIMPLE)
    bottom_most = (0, 0)
    if contours:
        l_contour = max(contours, key=cv2.contourArea)
        moments = cv2.moments(l_contour)
        if moments['m00'] != 0:
            center = int(moments['m10'] / moments['m00']),
             int(moments['m01'] / moments['m00'])
        else:
            center = (160,0)
        max_area = cv2.contourArea(l_contour)
        # get the bottom most point
        bottom_most = tuple(l_contour[l_contour[:,:,1].argmax()][0])
    else:
        center = (160, 0)
        max_area = 0
    return center, max_area, bottom_most
```

Fig.8: Program to obtain the object's information

4. Deep Learning program

GoogleColaboratory, a cloud-based platform was used for the AI training process. Images and data of the objects are sent to the PC from Raspberry Pi 4 to create the training model. Figure 9 shows the system diagram for image analysis and autonomous driving.

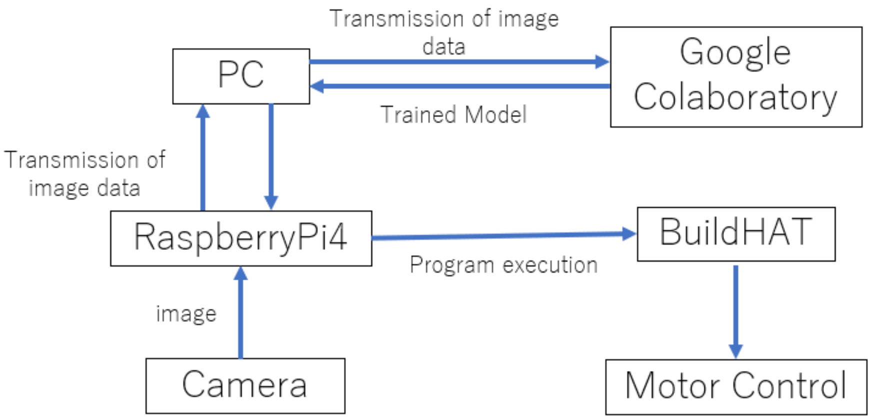
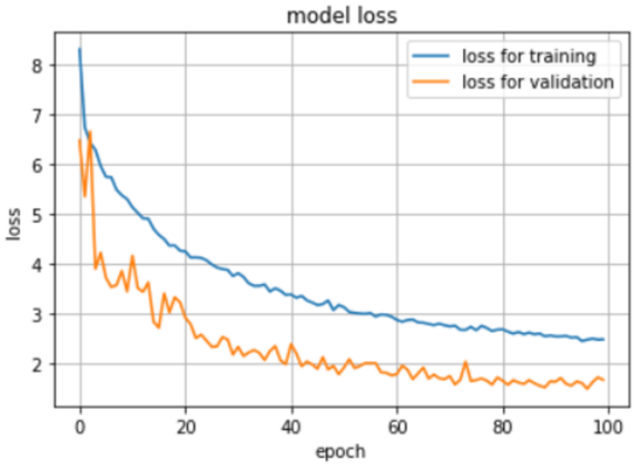


Fig.9: System diagram of image analysis and autonomous driving

In Deep Learning, it is necessary to adjust the parameters such as dropout rate, learning rate, weight decay, mini-batch size, and number of epochs before starting the training process. Figure 10 shows the configuration values used for the training process and results obtained from the process.

Dropout rate	0.5
Learning rate	0.01
Weight decay	0.000001
Mini-batch size	64
Number of epochs	100

(a) Configuration values



(b) Leaning result

Fig.10: Deep Learning results

The horizontal axis of the graph represents the number of training iteration, while the vertical axis represents the error between the predicted values generated by the training model and the actual values. As the number of training iterations increases, the data error decreases, allowing for the development of a more accurate training model.

5. Conclusion

In this study, we tried to develop teaching materials on autonomous driving of robots using Deep Learning. Using Raspberry Pi 4 as the controller and implementing Deep Learning programming with Python, the prototype teaching material was able to recognize and avoid the two types of obstacles used while driving through the obstacle course. The prototype was able to accurately recognize the randomly placed obstacles on the 3m x 3m obstacle course and avoid them by turning left or right without making any contact with the obstacles.

In the future, we aim to enhance the capabilities of the robot to include features such as parking assistance, detecting potential dangers and slowing down or stopping accordingly and implementing automatic deceleration before collisions. The goal is to improve the robot to a level where it can be practically introduced and used in educational settings in lectures.

6. References

[1] Daisuke FUJIMOTO (2019) ,A security on the Sensing process, Institute of Electronics, Information and Communication Engineers (IEICE), Fundamental reviews Vol.13,No.2,pp.142-150.

[2] Adrian Zlocki, Julian Bock, Lutz Eckstein (2017), Database of relevant traffic scenarios for highly automated vehicles, Autonomous vehicle Test & Development symposium,

2017

[3] Kishikawa, Arai (2019), Ensuring the safely of autonomous driving through inverse Deep Learning. The 33rd Annual Conference of the Japanese Society for Artificial Intelligence 2019, pp.1-4.

[4] Sugimoto, Uchida, Kurashige (2021), Evaluation of Indoor Mobile Robot Path Planning Algorithm Using Deep Reinforcement Learning, Journal of Industrial Application Engineering Society Vol. 9, No. 2, pp. 118-124.

[5] Hiroaki Agatsuma, Current Status and Issues of Driver Assistance and Autonomous Driving Technology Using Artificial Intelligence, Instrumentation and Control, Vol.54, No.11, November 2015, pp.808-815.

[6] Yukiko HATTORI, Daichi TAKAHASHI, Yuki ARAI, and Yuichi HASUDA (2019),esign and Production of the Robot for WRO International Competition, International Conference on Education, Economics, Psychology and Social Studies (ICEEPS2019) 209-217.

[7] Yuichi Hasuda, Keita Hirayama (2021), Development and Usability Test of Pesticide Spraying Robot for Greenhouse,International Journal of Engineering Research & Technology (IJERT) 10(5) 116-121.

[8] Yuichi Hasuda, Keigo Ozaki(2022), The Pesticide Spraying Robot Moving in A Greenhouse with A Line-Trace, International Journal of Engineering Research & Technology (IJERT) 11(1) 304-307.