Study on Platooning, Traffic Light Recognition, and PID-Based Path Control for Autonomous Robots

Hyun Seo Jeong, Eun Ju Jeong, Seo Yeon Kim, and Eunkyung Kim

Department of Artificial Intelligence Software
Hanbat National University
Daejeon, Korea
{hyunseo, eunju, syeon}@edu.hanbat.ac.kr, ekim@hanbat.ac.kr

***Abstract*—In this paper, we introduce the implementation of a path tracking for platooning autonomous driving robots. The system employs PID control, a feedback control mechanism, to ensure the robots remain on track. Ultrasonic sensors are also utilized to measure the inter-robot distance, enabling adaptive speed adjustments for the following robot. Specifically, the following robot decelerates to maintain a safe distance if the inter-robot distance decreases and comes to a complete stop if the distance falls below a predefined threshold. Conversely, when the inter-distance increases, the following robot accelerates to sustain the platooning configuration while preserving safe distance. Experimental results confirmed that implemented path tracking system and platooning algorithm work effectively.**

Keywords—PID, autonomous driving robots, path tracking, platooning driving, traffic light recognition

I. INTRODUCTION

In the era of the 4th industrial revolution, autonomous driving technology is emerging as one of the major innovative technologies [1]. Accordingly, research and development on data-based control methods of autonomous driving technology and rule-based control techniques using dynamics and model control techniques are actively underway. One of the rule-based control algorithms, Proportional-Integral-Derivative (PID) control, enables autonomous robots (or vehicles) to precisely adjust their position and speed, allowing for stable path tracking. In particular, platooning where two or more vehicles cooperate to form a cluster and drive requires interaction, communication, and maintenance of appropriate inter-vehicle distances. Platooning is an important application of autonomous driving technology, through which vehicles can cooperate to save energy and alleviate traffic congestion, among other benefits. In this paper, we study path tracking and platooning for autonomous driving robots using PID control. We propose an algorithm that enhances the path tracking performance of robots using PID control while enabling two or more robots to maintain consistent distance between themselves. Additionally, we implement actual autonomous driving robots to verify the path tracking using PID control.

The rest of this paper is organized as follows. We begin with research methods in Section II. We evaluate the path tracking ad platooning of autonomous driving robots via implementation in Section III. Finally, we conclude this paper in Section IV.

II. RESEARCH METHODS

In this study, we design the system with the aim of precise path tracking, safe driving, and platooning between two or more robots. To this end, PID control is applied to enable the robots to accurately follow a given path, and through the traffic light recognition system is implemented real time responses to traffic signals. In addition, through the platooning algorithm, two or more autonomous driving robots can drive in cooperation while maintaining inter-distance between themselves.

2.1 SYSTEM DESIGN

2.1.1 PATH TRACKING USING PID CONTROL

Autonomous robots require steering control to follow a target path. Steering control is implemented using PID (Proportional-Integral-Derivative) control, a type of feedback control. The PID controller utilizes P control (Proportional Control), I control (Integral Control), and D control (Derivative Control) as follows:

- "P control" is a proportional controller that uses the proportional gain (K_p) to generate a control output proportional to the error, ensuring the system's output approaches the target value. Here, the error refers to the difference between the target value and the actual measured value. However, in this process, the target value is not reached, and residual deviation remains, resulting in a steady-state error.
- "I control" is an integral controller that adjusts the control output based on the accumulated error, enabling the system to approach the target value more closely. This is primarily used to eliminate the steady-state error in the system. The integral gain

(K_i) increases the rate at which the error accumulates, thereby strengthening the control output. This eliminates the steady state error, which is difficult to solve only with "P control," and makes it possible to reach the target value closer.

- "D control" is a differential controller that uses the D gain (K_d) value in proportion to the rate of change of the error to control the response to rapid changes, improve the response speed of the system, and reduce overshoot to stably reach the target value.

The control equation of the PID controller is as follows:

$$Manipulated\ Variable = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de}{dt}, (1)$$

where $K_p, K_i,$ and K_d denote the value of proportional gain, the value of integral gain, and the value of D gain, respectively. In addition, $e(t)$ denotes a variable representing an error, i.e., the difference between the target value and the current value over time t.

We adopt the optimized values for each PID control gain through a manual tuning process, which required considerable time and trial and error [2]. In this paper, we implement the manual tuning method as follows: First, the value of K_p is adjusted to set it at a level that allows the system to approach the target value. However, during this process, the system fails to reach the target value, leaving a residual deviation that results in a steady-state error. To resolve this steady-state error, the residual deviation is gradually reduced using the value of K_i to be closer to the target value. Finally, the value of K_d is adjusted to decrease the system's response time.

Figure 1 shows method for calculating error values with 8 LSA sensors arranged in a line for path tracking, where the measured illuminance values from each sensor are stored and utilized in a list format. These values enable the autonomous robot to follow the white line on the track.

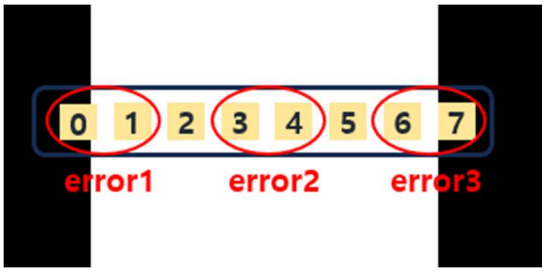


Figure 1. Method for Calculating Error Values

In particular, to enable precise control of the robot, error values (i.e., error1, error2, and error3) are calculated based on the data obtained from each sensor, respectively, as follows:

$$error1 = th - \frac{sensorvalue[0] + sensorvalue[1]}{2}. (2)$$

$$error2 = th - \frac{sensorvalue[3] + sensorvalue[4]}{2}. (3)$$

$$error3 = th - \frac{sensorvalue[6] + sensorvalue[7]}{2}. (4)$$

Here, th and $sensorvalue[i]$ denote the target value and the illuminance value measured by the LSA sensor i, respectively. The target value th indicates the value of the white line on the track measured by the robot's sensors. The error value using the average of the leftmost two sensor values is represented by Equation (2), the error value using the average of the center two sensor values is represented by Equation (3), and the error value using the average of the rightmost two sensor values is represented by Equation (4). These error values are used to determine the control adjustment so that the robot does not deviate from the track.

The manipulation amount is calculated according to the PID control formula in Equation (1). Through this formula, three manipulation amounts for each error value are generated, and the movement of the robot is delicately controlled so that the track may be accurately driven.

Algorithm 1 Steering Control

```
correction1 - Control input calculated based on the values of the two leftmost sensors
correction2 - Control input calculated based on the two central sensor values
correction3 - Control input calculated based on the two rightmost sensor values

procedure SteeringControl(correction1, correction2, correction3)
    if (sensorvalues[0] ≤ a) and ((sensorvalues[3] + sensorvalues[4])≥ b) and (sensorvalues[7] ])≤ c) then
        Strong right turn using correction1
    else if (sensorvalues[0] ≤ d) and ((sensorvalues[3] + sensorvalues[4])≤ e) and (sensorvalues[7] ≤ f) then
        Weak right turn using correction2
    else if (sensorvalues[0] ≥ g) and ((sensorvalues[3] + sensorvalues[4])≥ h) and (sensorvalues[7] ≥ i) then
        Weak left turn using correction2
    else if (sensorvalues[0] ≥ j) and ((sensorvalues[3] + sensorvalues[4])≤ k) and (sensorvalues[7] ≥ l) then
        Strong left turn using correction3
    else if (sensorvalues[0] ≤ x) and ((sensorvalues[3] + sensorvalues[4])≥ y) and (sensorvalues[7] ≤ z) then
        No steering control. Moving straight
    end if
end procedure
```

Algorithm 1 describes a control algorithm designed to ensure the stable driving of an autonomous robot along a track. This algorithm dynamically controls the robot's direction and adjusts the turning intensity for left and right turns based on data collected from LSA sensors mounted on the robot. Each sensor detects the relative position between the robot and the track's white line, enabling the robot to maintain its driving path and achieve stable navigation. Located at the far-left end of the robot, *sensorvalue*[0] detects

the relative position of the robot to the track's white line. Located at the center of the robot, *sensorvalue*[3] and *sensorvalue*[4] detect the central position of the white line relative to the robot. Located at the far-right end of the robot, *sensorvalue*[7] detects the relative position of the robot to the track's white line. The robot determines the required turning intensity using three control outputs and parameters (a, b, c, d, e, f, j, k, l, x, y, z). Each sensor provides distinct data about the robot's environment, and appropriate control criteria must be established based on this data.

2.1.2 Traffic Light Recognition

To enable autonomous driving robots to recognize traffic lights and respond accordingly, we make an image classification model which is constructed using the camera mounted on the robot. This model trained traffic light images using Convolutional Neural Network (CNN) [3]. The number of classes is two in total. In addition, 1,500 Red and 1,500 Green images were used as training data, respectively. The ratio was designated as 8:2 by dividing it into training data and verification data within the training data. The test data consisted of a total of 446 images, including 224 Red images and 222 Green images.

2.1.3 Platooning

In platooning, maintaining a safe distance between robots is essential for collision avoidance and smooth driving, requiring the following robot to respond appropriately to sudden speed changes of the leading robot.

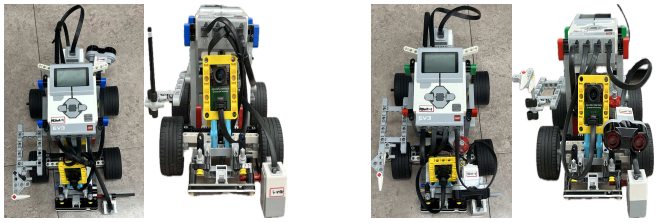
Algorithm 2 Maintaining inter-robot distance in Platooning

```
1:  Input: Distance between robots measured by ultrasonic sensors
2:  Output: Speed control commands for the robot
3:  procedure ControlDistance(distance)
4:      if distance ≤ 35 then
5:          if distance ≤ 27 then
6:              Stop the robot
7:          else if 27 <distance ≤ 35 then
8:              Reduce speed
9:          end if
10:     else if distance>35 then
11:         Increase speed
12:     end if
13: end procedure
```

Algorithm 2 describes an algorithm for maintaining a distance between robots using an ultrasonic sensor. The core of this algorithm is to measure the distance between robots and speed adjustment based on this distance. First, the distance between the following robot and the leading robot is measured in real-time using the ultrasonic sensor (line 1). If the distance becomes less than the minimum safe distance of 35 cm, the speed is reduced to about 0.12 m/s to maintain the distance, and if the distance becomes less than 27 cm, the robot is designed to stop (lines 4-8). When the distance between the robots exceeds 35 cm, the speed of the following robot is increased to about 0.17 m/s to maintain the 35 cm distance, enabling swarm driving (lines 10-11).

2.2 Experimental Methods

2.2.1 Autonomous Driving Robots



(a) Leading Robot (b) Following Robot

Figure 2. Autonomous Driving Robots: (a) Leading Robot and (b) Following Robot

Figure 2 shows (a) the leading robot and (b) the following robot, which are autonomous driving robots used in this study. In particular, the following robot is equipped with an ultrasonic sensor at the front to measure the distance from the leading robot. An LSA sensor used to track the line while driving is also mounted at the front, which serves to detect the position of the line so that the robot can accurately follow the path.

2.2.2 Experimental Track And Road Markings

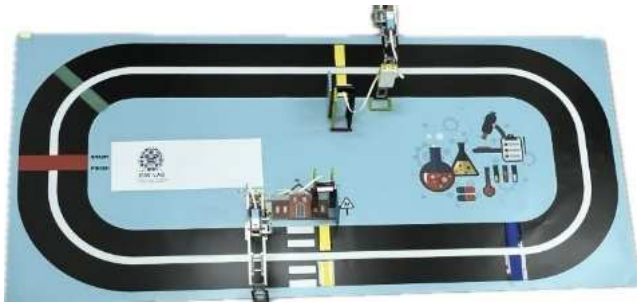


Figure 3. Experimental Track

Figure 3 shows the experimental track for confirming the path tracking and platooning of the autonomous driving robot proposed in this study. The green line and the red line indicate the starting point of the leading robot and the starting point of the following robot, respectively. The yellow line indicates the point at which to stop to recognize the traffic light and the blue line indicates the slow-speed zone.

2.2.3 Traffic Lights

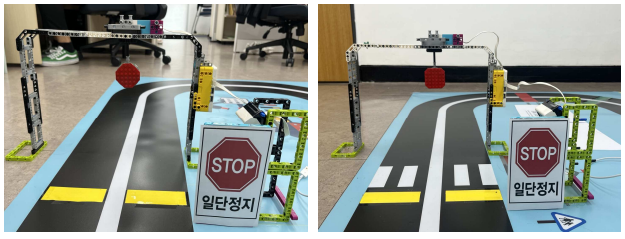


Figure 4. Two types of Traffic Lights

Figure 4 shows a traffic light recognized by the camera after the robot stops at the yellow line. The red part of the traffic light is divided into two types, circular and rectangular, and is operated by the detection sensor.

III. Experimental Results

3.1 Path Tracking Using PID Control

Table 1 summarizes values for the manual tuning process described in 2.1.1. The optimized control gain values for the leading and following robots obtained through the manual tuning process. These control gain values in Table 1 are adjusted according to the characteristics of each robot and the driving environment and are set for stable driving.

Table 1 The values for manual tuning process

	P gain	I gain	D gain
Leading Robot	$K_p = 5$	$K_i = 0.002$	$K_d = 0.13$
Following Robot	$K_p = 5.5$	$K_i = 0.02$	$K_d = 0.06$

3.2 Traffic Light Recognition

To evaluate the performance of the model, we measured the accuracy (ACCURACY) for training data and test data, and the results are as follows.

Table 2. Performance evaluation of the image classification model

	ACCURACY (%)	LOSS
Validation data	99.7	0.02
Test data	95.96	

Table 2 summarizes the performance of the image classification. In particular, the accuracy of this model is 95.96%. This confirms that the CNN-based image classification model can accurately identify traffic lights in real-time and appropriately reflect this information in the behavior of the autonomous robot.

3.3 Platooning

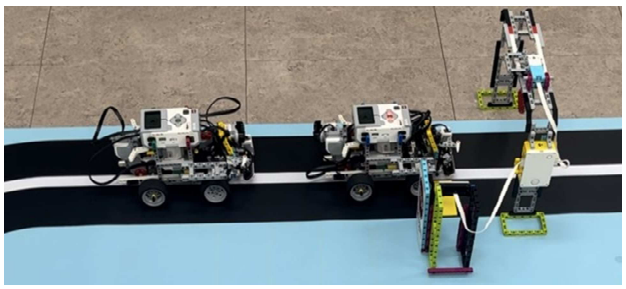


Figure 5. Maintaining distance between robots

Figure 5 shows the robots maintaining their distance when Algorithm 2 is applied in practice. When the leading robot stops to wait for a traffic light, the following robot, which was trailing, stops while maintaining a safe distance. This visually demonstrates the process of platooning while maintaining the distance between the robots.

3.4 Comprehensive Results

Both leading and following robots were driven on experimental tracks to validate our previously designed autonomous and platooning systems by integrating them.

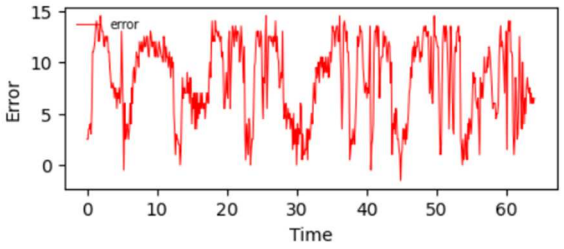


Figure 6. PID errors before PID control

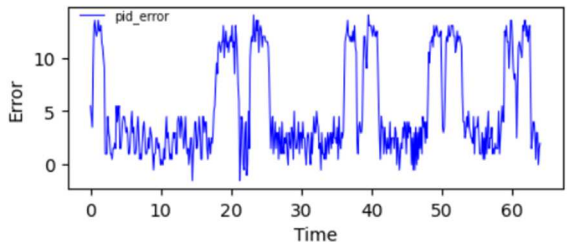


Figure 7. PID errors after PID control

Figures 6 and 7 show data comparing the leading robot's route tracking before and after the application of PID control, respectively. The x-axis represents the time taken by the robot to complete two laps of the track, and the y-axis shows error2 in the path tracking and swarm driving system design using PID control. As seen in the graph, in the case without PID control, the error value fluctuates sharply without a consistent pattern, indicating unstable driving. This shows that the robot is unable to accurately track the white line in both the straight and corner sections. Also, there was an issue where the robot would deviate from the track, requiring the robot to be moved back onto the track. Therefore, it can be seen that the driving without PID control is more time-consuming and inefficient. When PID control is applied, the error value remained relatively constant, and it is found that the robot followed the white line well without any sudden change in the straight line section. In the corner section, the robot briefly deviated from the white line but quickly returned to it, demonstrating stable driving. This indicates that the PID control is functioning effectively.

Figure 8 shows the two robots starting simultaneously in the experiment, maintaining a safe distance while driving and stopping. In particular, it is confirmed that the following robot stops without collision even when the leading robot stops in front of the traffic light. This demonstrates that the system's autonomous driving and platooning functions work

successfully in the experimental environment (see [4] for additional results).

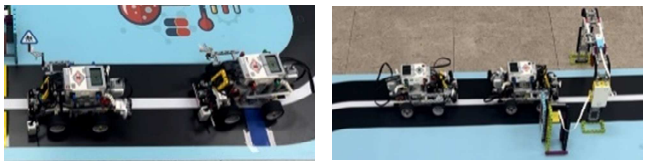


Figure 8. Maintaining distance during driving (left) and ensuring a safe distance when stopping (right)

IV. CONCLUSION

In this paper, we implement a path tracking and swarm driving system for autonomous robots. To ensure smooth driving without deviating from the track, we apply PID control, a type of feedback control. Additionally, we propose a platooning algorithm that measures the distance between the leading and following robots and maintains a certain distance using ultrasonic sensors. Our implements experiments show that PID-based path tracking and platooning for autonomous robots performs effectively. For future research, we will focus on improving autonomous driving systems by introducing additional sensor technologies such as LiDAR, RADAR, and GPS to improve robots' environmental awareness and platooning capabilities.

ACKNOWLEDGMENT

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2025-RS-2024-00437886) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation)

REFERENCES

[1] Novak, Libor. "Vehicle detection and pose estimation for autonomous driving." *Ph. D. dissertation, PhD thesis, Masters thesis* (2017).

[2] Pohjola, Mikael. *PID controller design in networked control systems*. MS thesis. 2006.

[3] Alexander Shustanov, Pavel Yakimov. "CNN Design for Real-Time Traffic Sign Recognition", Procedia Engineering, Volume 201, 2017, Pages 718-725, ISSN 1877-7058.

[4] "Study on Platooning, Traffic Light Recognition, and PID-Based Path Control for Autonomous Robots," URL: https://github.com/iSWLab/Study-on-Platooning-Traffic-Light-Recognition-and-PID-Based-Path-Control-for-Autonomous-Robots/blob/main/README.md