Genetic Algorithm based Obstacle Avoidance for 4-Wheeled Robot

Junmyeong Kim[1†], Kwanho Kim[1] and Kanghyun Jo[1]

[1]Department of Electrical, Electronic and Computer Engineering, University of Ulsan, South Korea
(Tel: +82 052-259-1664; E-mail: junmyeong7029@gmail.com, aarony12@naver.com, acejo@ulsan.ac.kr)

Abstract: This paper proposes an algorithm that uses for decision-making and control in Autonomous Mobile Robots (AMR) and performs in a virtual environment for simulation. In the decision-making stage, the algorithm uses the Velocity Obstacle (VO) and Artificial Potential Field (APF) methods to select the optimal velocity for avoiding collision and reaching the goal point in real-time. It additionally computes fitness that helps eliminates rapid changes in velocity. In control, the PID controller moves the robot along the velocity determined in the decision-making stage. The PID gain is updated by the gradient descent method to find the optimal gains. Proper brake torque is applied for the robot moves smoothly according to velocity previously determined. This parameter, α affects chance of avoidance obstacles, however it increase the travel time in experiment. The travel time averagely increase 9.54% at every α 0.1. When comparing the smallest and largest periods, the larger period resulted in a 29.9% decrease in control error. The results also showed that using the proposed PID controller resulted in a 13.6% decrease in control error compared to using a fixed PID controller.

Keywords: Robotic and Automation Systems; Intelligent Control; Adaptive and Optimal Control

1. INTRODUCTION

The use of indoor mobile robots, such as Autonomous Mobile Robots (AMR), is increasing.[1] AMRs have an important feature of being able to update the optimal path for moving obstacles like humans. The process for autonomous driving of AMR mainly consists of perception, decision-making, and control stages. In the perception stage, it detects the surrounding objects and their velocities, as well as its own position. Many studies have been performed in relation to this stage. Usually object detection method such as YOLO[2], Faster-RCNN[3] was used to detect obstacle's position. Moreover, to perform self-localization using particle filter.[4] In the decision-making stage, it outputs the velocity and path that the vehicle should drive based on the perceived information. When outputting the velocity and path, it should explore a path that avoids obstacles while getting closer to the destination using the location and velocity information of the obstacles. In the control stage, it performs motor torque and steering control to follow the velocity and path determined in decision-making stage.This paper focuses on the decision-making and control stages of a four wheeled robot that can be used as an AMR, assuming that the perception stage is well performed.

2. PROPOSED METHODS

This paper implements the decision-making and control stages of autonomous driving robots using genetic algorithms and PID controller. In the decision stage, the positions and velocities of obstacles are received from perception stage, and the reference velocity that the robot should move is output. To move the robot according to the reference velocity, the appropriate torque must be applied and the wheels must be steered, which is done in the control stage.

† Junmyeong Kim is the presenter of this paper.

2.1. Decision-Making

In the decision-making stage, the robot determines the direction and speed. It should move based on the position and velocity of obstacles through a genetic algorithm.

2.1.1. Fitness

In the paper that uses the genetic algorithm as a real-time obstacle avoidance method [5], safety and destination-seeking terms were used when calculating fitness to enhance safety. In this paper, the destination-seeking term is used as it is, and safety term is implemented using APF information. In addition, a term that prevents the movement of the robot from changing significantly by using the information of the previous frame is additionally used.

First the term based on APF is calculated using Eqs. (1) to (2).

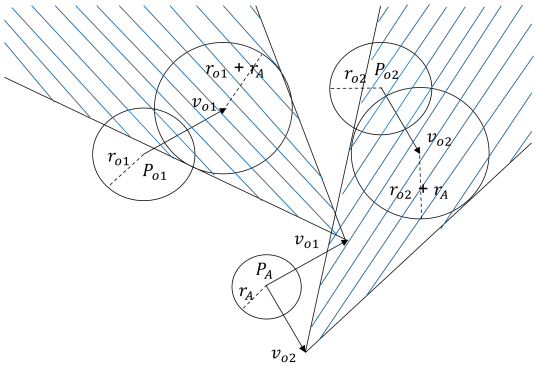

Fig. 1: Velocity Obstacle : Shaded area represents VO where robot should avoid to have velocities. If the robot moves at velocity in VO area, from the perspective of the obstacle, the robot has a relative velocity directed towards the obstacle.

$$R = \sum_{j=1}^{N} \frac{P_A - P_{Oj}}{||P_A - P_{Oj}||^3} \tag{1}$$

$$S(v_i) = \frac{R \cdot v_i}{\|R\|\|v_i\|} \tag{2}$$

The notation of variables is in Fig. 1, and v_i is one of the solutions selected in the genetic algorithm. Assuming that there are N obstacles, the repulsive force due to the obstacles is calculated as in Eq. (1), and the term S for safety is defined using Eq. (2) as the cosine similarity between R and v_i.

G is a term that makes the direction vector toward the destination have a high degree of fitness, and is the same as proposed in [5].

$$G(v_i) = \frac{||v_i|| \cos \Delta\theta}{v_{max}} \tag{3}$$

Where $\Delta\theta$ represents the angle between the velocity vector v_i and the vector pointing towards the destination from the robot.

In the decision-making stage, the velocity vector of the robot is output, which is limited to some extent by the direction and current speed of the robot. In other words, if the difference between the direction of the velocity in the previous and the current frames is higher than threshold, it has a limitation to controlling the robot at current velocity. Therefore, the fitness function includes a term C, which aims to decrease the angle between the previous and current velocity.

$$C(v_{i,t}) = \frac{v_{t-1} \cdot v_{i,t}}{\|v_{t-1}\|\|v_{i,t}\|} \tag{4}$$

Eq. (4), the subscript t means the current frame and $t-1$ means the previous frame. That is, Eq. (4) causes that a velocity vector similar to the previous velocity vector has a high fitness.

If a selected solution is within the VO area shown in Fig. 1, it can potentially cause a collision with obstacle in the future. In this case, the fitness, f is set to negative infinity. On the other hand, if the solution is outside the VO area, f is calculated by combining S, G, and C. The weights α, β, and γ are user-defined and represent the relative importance of each term, with their sum equal to 1.

$$f = \begin{cases} -\infty & \text{if } v_i \in VO \\ \alpha S + \beta G + \gamma C & \text{otherwise} \end{cases} \tag{5}$$

2.1.2. Crossover

Among those three recombination methods, the intermediate recombination method, generally showed the best performance in [5], is used to perform the crossover.

$$v_{newx} = v_{1x} + k_x(v_{2x} - v_{1x}) \tag{6}$$

$$v_{newy} = v_{1y} + k_x(v_{2y} - v_{1y}) \tag{7}$$

2.1.3. Mutation

The mutations proposed in this paper are given by Eqs. (8) to (9). A random noise within a specific range was added to the solution as a mutation in [5]. In this paper, to explore a wider range of velocities, a solution is randomly selected within the range of v_{max}.

$$v_{y,max} = \sqrt{v_{max}^2 - v_x} \tag{8}$$

$$\dot{v_y} = random(-v_{y,max}, v_{y,max}) \tag{9}$$

The variables v_x and v_y represent the solution before the mutation, while $\dot{v_x}$ and $\dot{v_y}$ represent the solution after the mutation. The x and y subscripts in Eqs. (8) to (9) can be interchanged. When $\dot{v_y}$ is randomly selected during mutation, the solution should not have a speed greater than v_{max}, so the range of randomly selected v_y is calculated as shown in Eq. (8).

2.2. Control

2.2.1. Torque Control

To drive a four wheeled robot at the velocity determined in decision-making, the torque of the motor and the steering angle must be controlled appropriately. When controlling the torque, the controller is designed to only consider the magnitude of the velocity, assuming that the steering is perfectly aligned with the direction of the speed. The velocity determined in the decision-making is what the controller needs to track, but if the determined speed changes drastically every frame, it becomes difficult to track the speed accurately. Therefore, in this paper, the determined speed over a certain number of frames are averaged to avoid sudden changes in the reference velocity, and the gains of PID are updated in real-time using gradient descent.

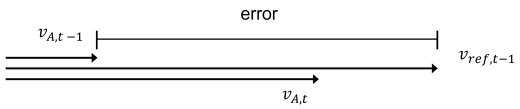

Fig. 2: Input of PID Controller

In Fig. 2, The error refers to the input of the PID controller, not the loss used in Eq. (11). The PID controller outputs torque to reduce the error. On the other hand, the PID gain, K updates as Eq. (11). The optimal gain will be the one that minimizes the error in the next frame. Therefore, the loss function, L used in the gradient descent is calculated by Eq. (10).

$$L = \frac{v_{ref,t-1} - v_{A,t}}{error} \tag{10}$$

$$K_n^t = K_n^{t-1} - \eta_n \frac{\partial L}{\partial K_n}, \quad n \in P, I, D \tag{11}$$

2.2.2. Steering Control

Assuming that the robot's inertia is small, in order for a four wheel robot to drive at the reference velocity, the

steering angle and the direction of the reference velocity v_{ref} must be the same. However, if the steering rotational speed w_{max} is not sufficient when steering the wheels in the direction of the reference, the robot will not move in the direction of the reference velocity. Fig. 3 shows a situation where the robot is moving at a speed of v_A and decides to move at a speed of v_{ref} due to obstacles.

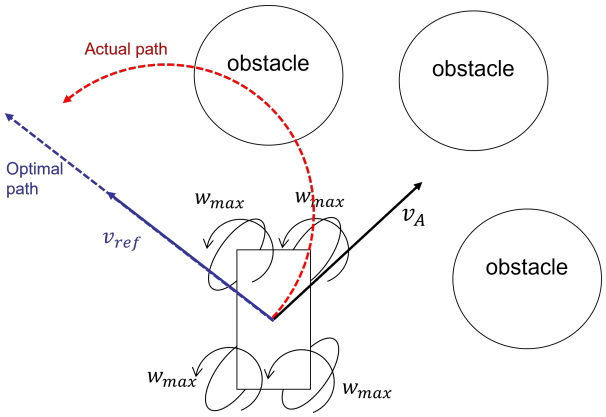

Fig. 3: Steering scenario of four wheeled robot

If the magnitudes of v_A and v_{ref} are the same, the robot will steer with a maximum angular velocity of w_{max} to follow the optimal path. However, if the robot's speed does not decrease, the actual path will follow a red arc with a radius of $r = v/w$, as shown in Fig. 3, which can lead to collisions with obstacles. Therefore, in this paper, the brake is applied when the angle difference between v_A and v_{ref} is greater than a specific value. In addition, a term was added to suppress significant changes in the direction of the velocity vector between the previous and current frames using Eq. (4) mentioned earlier.

3. EXPERIMENTS

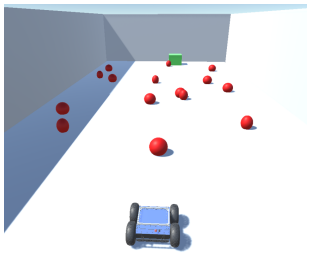

Fig. 4: Virtual environment for simulation

Table 1: Parameters of robot

$v_{max}(m/s)$	$w_{max}(rad/s)$	Mass(kg)	Size(m^2)
3.5	$\pi/2$	38kg	0.8x0.8

The experiments were performed in the Unity virtual environment as shown in Fig. 4. The red spheres represent obstacles, green box and the blue vehicle mean destination and the robot to be controlled. The robot's maximum speed, maximum angular velocity, weight, and size are set as shown in Table 1.

Table 2: Sets of α, β, and γ

	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
α	0.3	0.5	0.7	0.225	0.425	0.625
β	0.7	0.5	0.3	0.625	0.425	0.225
γ	0	0	0	0.15	0.15	0.15

Table 3: Control error, travel time, and the number of collisions according to period and set

Period	Set	Control Error	Travel Time(s)	Number of collisions(times)
5	Set 1	0.162112	14.9577	2
	Set 2	0.190858	16.06577	2.6
	Set 3	0.147255	17.25478	1.4
	Set 4	0.140882	16.23653	1.8
	Set 5	0.17074	17.97266	1.2
	Set 6	0.179045	22.78789	1.2
10	Set 1	0.157811	10.98864	0.4
	Set 2	0.129879	13.92514	0.6
	Set 3	0.148420	15.76225	2.2
	Set 4	0.17398	15.3909	1.6
	Set 5	0.111902	16.31062	0.8
	Set 6	0.210888	20.81502	2.6
15	Set 1	0.143786	9.6153	0.2
	Set 2	0.132887	10.83767	0.6
	Set 3	0.148263	15.18906	1.8
	Set 4	0.192443	14.17479	0.6
	Set 5	0.134359	15.85427	1
	Set 6	0.127579	18.11844	0.8
30	Set 1	0.141737	8.943877	0.4
	Set 2	0.169434	10.29256	1.4
	Set 3	0.110606	13.36603	2.2
	Set 4	0.134479	13.94875	1
	Set 5	0.110192	15.07059	2
	Set 6	0.117533	19.04097	2
50	Set 1	0.116489	8.960921	0.4
	Set 2	0.128591	10.31191	0.2
	Set 3	0.110706	14.24613	3.2
	Set 4	0.117635	12.34133	0.8
	Set 5	0.13614	15.19542	2.6
	Set 6	0.08479	17.69625	1.2

Obstacles were divided into two types. 9 large obstacles and 6 small obstacles were placed on the plane. The radius of the large obstacle is set to $0.25m$, and the radius of the small obstacle is set to $0.2m$. Each obstacle has a speed of $2m/s$ or $1m/s$. The size of the plane is $(10 \times 20)\ m^2$.

In the experiment, only obstacles with a distance of $4m$ or less from the robot in motion were considered to calculate VO and determine the optimal velocity. Since the FPS in the Unity experimental environment was fixed at 30, the angular velocity is $90°/s$ when the steering angle changes by $3°$ during one frame. An algorithm was implemented to brake in proportion to the speed when the steering angle that should change per frame is more than $2°$.

Table 3 shows the results of experiments conducted by varying several parameters of the algorithm proposed in this paper. The values of parameters α, β, and γ, which are used to calculate fitness, were specified into six sets as shown in Table 2. The period in Table 3 means the number of previous samples collected when averaging the speed. When obtaining the average, a weighted average method was used that assigns a higher weight to the speed of the latest frame, and the weight is set as in

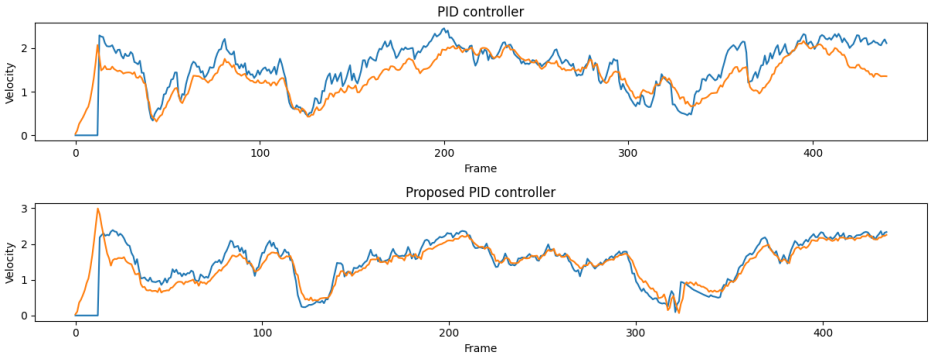

Fig. 5: Comparison between PID controller and Proposed PID controller

Eq. (12).

$$v_{ref,t} = \sum_{i=1}^{M} w_i v_{ref,t-i+1} \tag{12}$$

Control error, travel time to the destination, and number of collisions with obstacles were used as the metrics for comparison. Loss calculated using Eq. (10) was used as control error. During a specific sampling period, a total of five experiments were performed for one set, and the averages of the experiments were shown in the table. Table 3 shows that the control error decreases as the sampling period increases.

In the Fig. 5, the x-axis represents the number of frames in Unity. The first graph shows the reference speed and the actual speed of the robot when the experiment was conducted using the experimentally determined PID gain without using the gradient descent method. The blue graph represents the speeds output in the decision-making stage of autonomous driving, and the orange graph is the actual speed of the robot obtained by tracking and controlling the output speed. The experimentally determined proportional gain, integral gain and derivative gain were set as shown in Table 4.

Table 4: Initial PID gain

κ_P	κ_I	κ_D
20	0.8	8

On the other hand, the second figure is the control result obtained through the gradient descent method using the values in Table 4 as the initial PID gain. Table 5 shows the control errors when using a fixed PID gain and updating PID gain using gradient descent.

Table 5: Control error from Fixed PID gain and varying PID gain.

	Fixed	Varying
Control Error	0.35569	0.29003

4. CONCLUSION

This paper shows the four-wheeled robot behavior controlled by GA for avoiding obstacles in a dynamic environment. In the decision-making stage, VO and APF, which are obstacle avoidance algorithms, were used as the fitness of the genetic algorithm to avoid obstacles. Also, the velocity determined in the decision-making stage affects how easy and stable the control becomes. As a result, it was shown in Table 3 that the error for control was reduced. In addition, Fig. 5 and Table 5 show that the error in the control stage is reduced when the gain is modified in real time through the gradient descent method rather than using the fixed gain of the PID controller. The decision-making and control algorithm proposed in this paper can be applied to self-driving robots for stores or self-driving cars, which are recently increasing. In the future, the perception stage will be included in this work and applied to a robot in the real world.

ACKNOWLEDGMENT

This result was supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(MOE)(2021RIS-003)

REFERENCES

[1] K. Zhou, "Whether the agv/amr can be used in e-commerce," 2022.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[3] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[4] T. Röfer, T. Laue, and D. Thomas, "Particle-filter-based self-localization using landmarks and directed lines," in *RoboCup 2005: Robot Soccer World Cup IX 9*. Springer, 2006, pp. 608–615.

[5] Z. Gyenes, L. Bölöni, and E. G. Szádeczky-Kardoss, "Can genetic algorithms be used for real-time obstacle avoidance for lidar-equipped mobile robots?" *Sensors*, vol. 23, no. 6, p. 3039, 2023.