Dynamic Obstacle Avoidance in a Complex Environment using Meta-heuristic Algorithms for 4-wheel Robot

Junmyeong Kim, Kwanho Kim and Kanghyun Jo
Dept. of Electrical, Electronic and Computer Engineering
School of Electronic Engineering
University of Ulsan, Ulsan, Korea
kjm7029@islab.ulsan.ac.kr, aarony12@naver.com, acejo@ulsan.ac.kr

***Abstract*—Dynamic obstacle avoidance is one of the most important tasks for autonomous navigation of robots. Accordingly, several algorithms have been proposed to determine the velocity at which the robot can drive while avoiding obstacles. Meta-heuristic algorithms are a class of optimization methods that are inspired by natural processes and social behavior. This paper uses meta-heuristic algorithms such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) to determine the optimal velocity. The algorithm contains formulas and methods that help control the 4-wheel robot. Also, this work are experimented in a dynamic and complex virtual environment. The experimental environment is set up as dynamic 45 obstacles. The experiments are consists of about two methods and six parameter sets for considering various situation. When implementing to experiment about obstacle avoidance with proposed methods, collision counts and travel times are decreased 0.74 times and about 9.88 second on average with PSO.**

***Index Terms*—Obstacles avoidance, Meta-heuristic, Robotics**

I. Introduction

In recent years, several studies about obstacle avoidance in robots have been conducted. [1]–[3] Most obstacle avoidance algorithms consider the position and velocity of obstacles to determine a path for the robot to avoid obstacles and reach its destination. However, to implement the obstacle avoidance of robots in real-environments, it is essential to consider the physical limitations of the robot and making decisions regarding its velocity. For estimating a suitable velocity for the robot in real environment, this work proposes methods for initializing the solution set with considering previous velocity and fitness evaluation stages of the Genetic Algorithm (GA) process. Furthermore, when using the Velocity Obstacle (VO) method, the paper expands the range of velocities by considering time to crash and distance between obstacles and robot. Furthermore, Particle Swarm Optimization (PSO), a prominent meta-heuristic method, is employed instead of GA to compare and analyze the proposed approach in more complex experimental scenarios. All experiments are conducted under the assumption that information of positions, velocities of obstacles and robot is provided. The main contributions of this paper are as follows:

- Proposing the modified VO method that considers time to collision and the distance, thereby expanding the solution space.
- Suggesting methods for generating the initial solution set and fitness component for considering the robot's velocity.

II. Background

A. Artificial Potential Field

Artificial Potential Field (APF) [4] is a simple obstacle avoidance algorithm based on the principle of generating a strong force as the distance decreases, similar to gravitational or electromagnetic fields. When encountering obstacles that the robot needs to avoid, it generates a repulsive field, creating the force in the opposite direction of the obstacles. On the other hand, for the goal that the robot needs to reach, an attractive field is generated, creating a velocity toward the destination. By summing up the repulsive velocities for multiple obstacles and the velocity toward the goal, the algorithm determines.

B. Velocity Obstacle

VO [5] is a method for determining the velocity space for avoiding colliding using information about the obstacle's velocity. It assumes that both the robot and the obstacles have a circular shape. Then, the relative velocity between the robot and each obstacle is calculated to predict whether the robot will collide with the obstacle in the future. The set of velocities that are anticipated to result in a collision in the future is defined as the VO. Additionally, considering the robot's current velocity, it defines the set of velocities that the robot can physically achieve in the next frame as Reachable Velocity (RV). The area excluding the VO from the RV is defined as the Reachable Avoidance Velocities (RAV). When determining the robot's velocity, the velocity within the RAV ensures that the robot will avoid collisions with obstacles.

C. Genetic Algorithm

GA [6], a meta-heuristic technique inspired by evolutionary biology, is used to solve optimization problems. It generates

a set of solutions based on random or predefined rules. Each solution is called a chromosome, the values within a solution are called genes, and the set of solutions is called a population. Subsequently, the fitness of each chromosome is evaluated for the given problem, and chromosomes with higher fitness are selected as parents. Through crossover and recombination using the selected parents, offspring are created by exchanging portions of the existing chromosomes. Additionally, mutation probabilistically modifies the values of the offspring to prevent situations where the algorithm gets stuck in local minima. This process gradually improves the quality of solutions, approximating the optimal solution.

D. Particle Swarm Optimization

PSO [7] is a meta-heuristic algorithm that mimics swarm intelligence in nature to solve optimization problems, similar to GA. It forms a swarm consisting of multiple particles, where each particle represents a candidate solution and has its own position and velocity in the algorithm. The particles utilize their current position, velocity, locally discovered the best solution and globally discovered the best solution (swarm best) within the swarm to update their velocities and their positions.

III. Proposed Methods

A. Filtering VO

When determining the optimal velocity, this work employed the VO method to filter out the robot's velocity. When calculating VO in a complex environment, the region of velocity can be limited. However, because of numerous obstacles, the VO also includes the area where the robot can safely navigate. To reduce this area, this paper proposed two methods.

1) ***High collision-time elimination****:* In Fig. 1, the original VO is represented by the dashed line, P denotes the positions of the robot and obstacles, r represents the radius, and v indicates the velocity. The subscript o corresponds to obstacles and A represents the robot. If the relative velocity with respect to an obstacle is v_c during a certain time, it can be guaranteed that there will be no collision with the obstacle within that time. In other words,the set of velocities whose time until collision is greater than T_{max} is removed from VO. The minimum distance at which the robot collides with the obstacle is calculated as in Equation (1).

$$d_{min} = (P_o - P_A)(1 - \frac{r_o + r_A}{\|P_o - P_A\|}) \tag{1}$$

$$v_c = \frac{d_{min}}{T_{max}} \tag{2}$$

By driving the robot with the velocity v_c computed using Equation (2), collisions before T_{max} can be prevented. Therefore, the blue region in Fig. 1 is used as the VO region that the robot cannot navigate through.

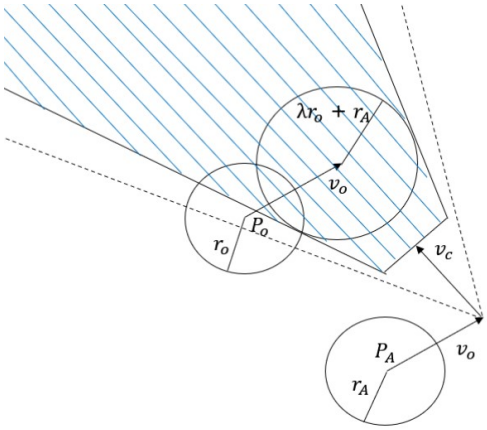

Fig. 1: Filtering a part of VO. The blue area represents the filtered-VO. An area of the dashed line represents the original VO. v_c is determined by high collision-time elimination and the radius of the obstacle is decreased by the distance proportional decrease method.

2) ***Distance proportional decrease****:* In the experimental section of this paper, to compute only the necessary VO for robot navigation, the VO is calculated for considering the obstacles whose distance is shorter than a threshold. However, among the obstacles whose distances are shorter than the threshold, the VO of distant obstacles has less significance, allowing for a reduction in the VO region. The scale factor λ is calculated inversely proportional to the distance, as shown in Equation (3).

$$\lambda = \frac{1}{\|P_o - P_A\|} \tag{3}$$

B. Fitness Function

The fitness function is used to determine which velocity of driving in an area other than VO allows the robot to reach a destination while avoiding obstacles. In the [8], the sum of two components is used as a fitness function to enhance safety while achieving faster arrival at the goal. However, if the optimal velocity is in the opposite direction to the robot's current velocity or in a direction where control is not possible, the robot will not move as desired. In this paper, to address this issue depicted in Fig. 2, an additional term is proposed in the fitness function, utilizing three components as shown in Equation (4).

$$f = \begin{cases} -\infty & \text{if } v_i \in VO \\ \alpha S + \beta G + \gamma C & \text{otherwise} \end{cases} \tag{4}$$

1) ***Safety Component****:* The component related to safety is calculated as shown in Equations (5) and (6).

$$R = \sum_{j=1}^{N} \frac{P_A - P_{oj}}{\|P_A - P_{oj}\|^3} \tag{5}$$

$$S(v_i) = 1 - \frac{\|R - v_i\|}{v_{max}} \tag{6}$$

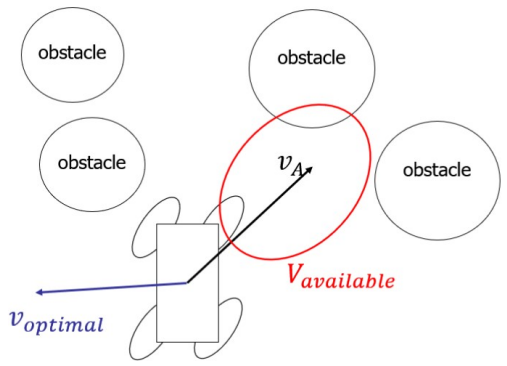

Fig. 2: The problem of control. $v_{optimal}$ is an optimal velocity of the situation while $V_{available}$ is a set of velocities that can be performed by a controller at the next frame.

Equation (5) represents the repulsive component of APF. R is the velocity at which the robot moves away from the obstacles. When the robot moves at a velocity similar to R, it becomes safer from the collision with the obstacles. Equation (6) calculates the similarity between R and the solution. S is limited to the range [0, 1], and a higher value indicates safer velocity.

*2) **Goal Component**:* Equation (7) is same with fitness function in [8] proposed.

$$G(v_i) = \frac{\|v_i\| \cos(\Delta\theta)}{v_{max}} \tag{7}$$

$G(v_i)$ has a higher value when the direction is similar to the goal and the magnitude is closer to v_{max}. That is, the higher the velocity toward the destination, the higher the fitness.

*3) **Control Component**:* In a situation such as Fig. 2, the optimal velocity for avoiding obstacles while reaching the destination is $v_{optimal}$. However, if the robot is unable to control its motor torque and steering to achieve the optimal velocity, it needs to find the optimal velocity within $V_{available}$. In this paper, this work proposes a term C A component C is introduced to ensure that the selected velocity is similar to the current robot's velocity.

$$C(v_i) = 1 - \frac{\|v_A - v_i\|}{v_{max}} \tag{8}$$

As shown in Equation (6), the range of C is constrained to [0, 1], and this term increases the tendency for the robot to move at a velocity similar to its current velocity.

C. Determining Velocity

To solve a problem such as Fig. 2, the robot's next velocity should be in the $V_{available}$ area. However, the distribution of population is following Fig. 3 (a) in the [8]. In other words, when generating candidates of velocity, it does not consider the current velocity of the robot. This paper suggested two novel methods for initializing the solution set by considering robot's velocity v_A.

The first method is using accelerations as a solution set instead of velocities. The robot is unable to accelerate above a specific value because of the motor's maximum torque, steering speed, slip, etc. Therefore, a_{max} is defined as the maximum acceleration that can be changed during one frame by the control motion of the robot. If the solution set is initialized to acceleration considering a_{max}, the expected velocity v_i of the robot can be calculated through Equation (9).

$$v_i = v_A + a_i \tag{9}$$

If the meta-heuristic algorithm is used with the solution set initialized by Equation (9), the problem shown in Fig. 2 will not occur.

The second method is to randomly select and filter the velocities when initializing the solution set by Equation (8). When the control component of the fitness function is multiplied by randomly initialized velocities, solutions with too large a difference from the current velocities are filtered out, as shown in Fig. 3-(c). By repeating the optimization algorithm using the solution set, it is possible to select a velocity that is easy to control.

*1) **Genetic Alogrithm**:* The first is generating the chromosomes that have velocity using two initializing methods proposed. After then, the fitness is calculated about three components to decide the parent chromosomes with the highest fitness value. Offspring is generated with recombination between two parents. In this paper, the intermediate coordinate recombination method proposed in [8] is used. The v' is the offspring and their components follow Equation (10) and (11). v_1 and v_2 represent the velocity of each parent chromosome. k_x and k_y are randomly generated weights.

$$v'_x = v_{1x} + k_x(v_{2x} - v_{1x}) \tag{10}$$

$$v'_y = v_{1y} + k_y(v_{2y} - v_{1y}) \tag{11}$$

*2) **Particle Swarm Optimization**:* The method for setting the particle's initial position p_0 is the same as the initial population used in GA. Each particle's initial velocity v_0 is randomly generated with a constrained range. The next particle's position p_{t+1} is obtained by the sum of the current position p_t and v_t that velocity of the particle. Equation (12) represents this process. v_t is calculated using Equation (13). N means number of total steps, p_l is local best position that discovered by itself and p_g is best position among of p_l, and w_1, w_2 are weight of local and global components. To control the weight of the previous particle velocity, this work applies the inertia rate IR. It is set to be the same value as N at the first time step and decreased at each step. Because of this process, the weights of the previous velocity are decreased. Fig. 4 is shown the process of steps in PSO.

$$p_{t+1} = p_t + v_t \tag{12}$$

$$v_{t+1} = (\frac{IR}{N})v_t + w_1(p_l - p_t) + w_2(p_g - p_t) \tag{13}$$

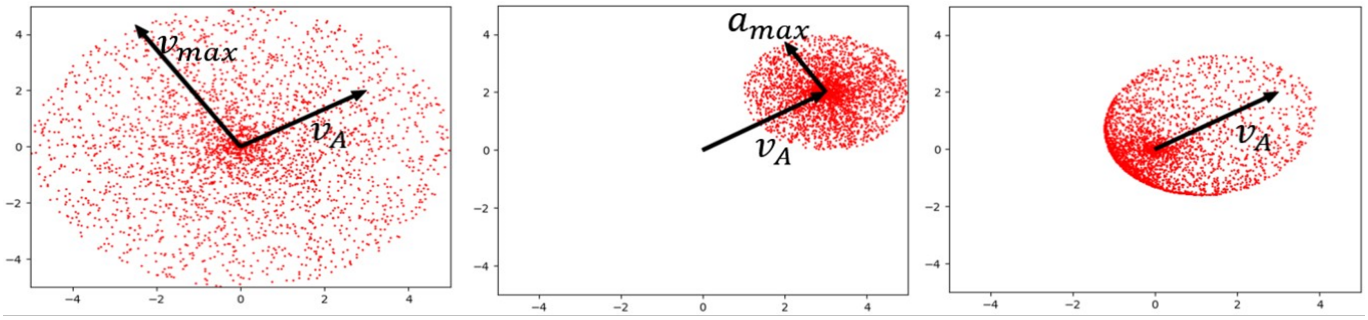

Fig. 3: Comparison of solution sets. 3000 points are randomly selected in each condition. Search space of (b) and (c) is lower than (a). Thus, it requires less convergence time as well as helping control.

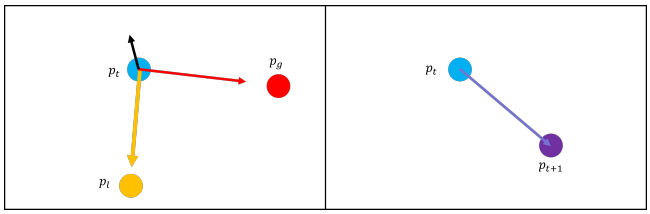

Fig. 4: The process of steps in PSO. If the particle's situation is the same as to the left, the new position of the particle is determined such as right. The blue,red and yellow, purple mean current position of particle, the the global best position, the local best position, and next position of particle. black arrow represents component about v_t.

D. Control Method

To control the robot with the desired velocity, a PID controller is used. The input of the controller is optimal velocity from meta-heuristic algorithms. PID controller outputs a torque for the robot's driving with optimal speed. PID controller only controls torque, not steering. Because the Four Wheel Steering (4WS) robot can steer their all wheels, the steering process is easier than Two Wheel Steering (2WS) robot. When controlling the steering, all wheels are rotated to desired direction for driving that direction. However, the stability of the robot will be problematic when the desired steering angle is too large as shown in Fig. 2. To solve this problem, the robot applies brake torque for unrealistic steering angles.

IV. Experiments

A. Environment

Experiments are conducted in a virtual environment as shown in Fig. 5. Parameters of the robot and obstacles are indicated in Table I and II. PID gains are experimentally determined. Each obstacle shuttles between two points. When moving once, the speed is randomly set in the range of $[0, 3]\, m/s$. The size of the stage is $(20 \times 40)\, m^2$.

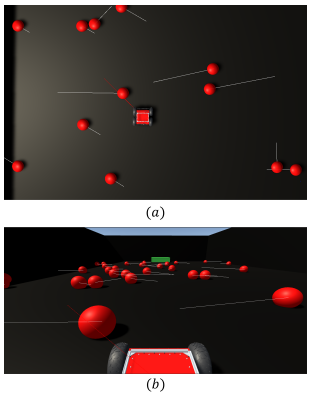

Fig. 5: The environment of experiments. The line segment from objects represents the velocity. (a) Top view of environment. (b) View from the robot.

TABLE I: Robot Configuration

Width(m)	Height(m)	$v_{max}(m/s)$	$a_{max}(m/s^2)$
0.8	0.8	3	1
Mass(kg)	K_P	K_I	K_D
40	20	0.5	8

TABLE II: Obstacle Configuration

Radius(m)	Number	Velocity(m/s)
0.25	45	$0 \sim 3$

B. Implementation Details

For safe driving, initially, the filtering VO method is not used and only obstacles with a distance of $5m$ or less from the robot are considered to calculate VO. However, it is found that the best fitness of the algorithm equals 0 because all the velocities that the robot can select are in the VO region. Therefore, if the fitness of the determined velocity is 0, the filtering VO method is used with $T_{max} = 2$, and obstacles whose distance is shorter than $2m$ are considered.

When using GA and PSO, the number of solution sets is set to 100 and repeated 50 times. 20 solutions are exchanged

TABLE III: Results of Experiments

Method	Set	$Velocity$		$Acceleration$		$Current$	
		$TravelTime$(s)	$Collision$(times)	$TravelTime$(s)	$Collision$(times)	$TravelTime(s)$	$Collision$(times)
PSO	1	24.58618	**1.4**	13.91011	**0.5**	18.97242	0.97
	2	21.14997	2.21	13.14385	1.82	16.12843	0.94
	3	20.50068	1.51	13.10333	1.39	15.49436	**0.82**
	4	17.96307	2.03	12.75	1.76	15.50069	1.48
	5	15.14577	1.56	12.55	1.51	15.17344	1.31
	6	-	-	29.25334	1.54	41.66845	1.51
GA	1	16.75648	**0.58**	13.68867	1.84	17.83086	1.27
	2	16.93558	0.85	12.63149	1.82	19.49191	**0.94**
	3	15.78198	0.6	13.2018	1.67	18.73333	1.93
	4	16.04467	1.95	12.74613	2.18	19.08667	2.19
	5	15.693	1.84	12.27026	2.51	18.63655	1.93
	6	34.10442	0.86	28.16003	**1.02**	38.13097	1.28

by offspring in GA. Brake torque is applied when the required steering angle per frame is over $10\,^{\circ}$.

As shown in Table IV, every experiment is implemented with 6 sets of parameters. To observe the effect of the control component, γ is set up as 1 or 2 from $Set1$ to $Set3$, and set to be 0 from $Set4$.

TABLE IV: Sets of Parameters

	$Set1$	$Set2$	$Set3$	$Set4$	$Set5$	$Set6$
α	1	2	1	1	1	2
β	1	2	2	1	2	1
γ	1	1	2	0	0	0

C. Result

To evaluate the performance of methods and effect of parameters, all experiments are conducted 100 times at each parameter and method. In Table III, $TravelTime$ and $Collision$ represent the averages of the travel time and collision counts during whole experiments $Velocity$ refers to the method that uses velocities as the solution sets, $Acceleration$ means the method that uses acceleration as the population, and $Current$ indicates the method that considering the previous velocity.

In all experiments of PSO and GA, the minimum collision time occurs when the weight of γ is non-zero except GA with acceleration solution set. When comparing the results between $Set1$ and $Set4$, both the original method and the proposed method showed fewer collision count in $Set1$. Especially, when using PSO and acceleration as the solution sets, the collision count in $Set4$ decreased by 1.26 compared to $Set1$.

When using the original populatio method in PSO, the minimum collision time is 1.4. However, when setting the population as acceleration and using the method of considering current velocity, the collision counts are decreased by 0.9 and 0.58. Additionally, for both methods, Travel Time is decreased by about 10.5 seconds and 10 seconds.

When α is set to be larger than β, it resulted in an increase of Travel Time by more than double in all cases. Especially, when using the original population method in PSO and setting the fitness parameters with $Set6$, it showed the result where the robot is unable to reach its destinations.

V. Conclusion

To make the controlling robot easier, this work suggested the modified VO algorithm, novel fitness function, and methods for initializing the solution set in meta-heuristic algorithms. Experiments are implemented in the virtual environment with 45 dynamic obstacles. When applying the component for considering the previous velocity to the fitness function, collision counts are decreased overall in the experiment. The travel time and collision count are decreased by about 10 seconds and 0.9 times when using acceleration as the solution set instead of velocity with the PSO method. The results of experiments prove improvement in performance.

Acknowledgment

This result is 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] X. Cheng, S. Zhang, S. Cheng, Q. Xia, and J. Zhang, "Path-following and obstacle avoidance control of nonholonomic wheeled mobile robot based on deep reinforcement learning," *Applied Sciences*, vol. 12, no. 14, p. 6874, 2022.

[2] M. Al-Mallah, M. Ali, and M. Al-Khawaldeh, "Obstacles avoidance for mobile robot using type-2 fuzzy logic controller," *Robotics*, vol. 11, no. 6, p. 130, 2022.

[3] R. Singh, T. K. Bera, and N. Chatti, "A real-time obstacle avoidance and path tracking strategy for a mobile robot using machine-learning and vision-based approach," *Simulation*, vol. 98, no. 9, pp. 789–805, 2022.

[4] Y. Koren, J. Borenstein *et al.*, "Potential field methods and their inherent limitations for mobile robot navigation." in *ICRA*, vol. 2, no. 1991, 1991, pp. 1398–1404.

[5] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[6] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[8] 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.