robotics

MDPI

Article

Development of Local Path Planning Using Selective Model Predictive Control, Potential Fields, and Particle Swarm Optimization

Mingeuk Kim *, Minyoung Lee, Byeongjin Kim and Moohyun Cha

Korea Institute of Machinery and Materials, Daejeon 34103, Republic of Korea
* Correspondence: kmg0702@kimm.re.kr

Abstract: This paper focuses on the real-time obstacle avoidance and safe navigation of autonomous ground vehicles (AGVs). It introduces the Selective MPC-PF-PSO algorithm, which includes model predictive control (MPC), Artificial Potential Fields (APFs), and particle swarm optimization (PSO). This approach involves defining multiple sets of coefficients for adaptability to the surrounding environment. The simulation results demonstrate that the algorithm is appropriate for generating obstacle avoidance paths. The algorithm was implemented on the ROS platform using NVIDIA's Jetson Xavier, and driving experiments were conducted with a steer-type AGV. Through measurements of computation time and real obstacle avoidance experiments, it was shown to be practical in the real world.

Keywords: autonomous ground vehicle; path planning; model predictive control; particle swarm optimization; potential field; driving simulation; driving test

Citation: Kim, M.; Lee, M.; Kim, B.; Cha, M. Development of Local Path Planning Using Selective Model Predictive Control, Potential Fields, and Particle Swarm Optimization. *Robotics* **2024**, *13*, 46. https://doi.org/10.3390/robotics13030046

Academic Editor: Yugang Liu

Received: 31 January 2024
Revised: 4 March 2024
Accepted: 6 March 2024
Published: 8 March 2024

1. Introduction

Autonomous driving vehicles are a subject of active research worldwide, particularly in the fields of indoor and outdoor robots, automobiles, space robot, and the military. The primary objectives of this research are efficient exploration, saving time, and safety. Various ideas are being researched and presented to achieve these goals.

Autonomous driving consists of three main components: environmental awareness, path planning, and driving control [1]. Environmental awareness involves the use of sensors, such as cameras and LiDAR, to perceive surroundings. Path planning incorporates various methodologies, including graph-search-based, sampling-based, interpolation-based, and optimization-based methods [1]. These techniques aim to formulate a safe driving path based on the information gathered from environmental awareness.

Graph-search-based methods create nodes within the domain and construct a graph [2], which includes algorithms like Dijkstra [3] for finding the shortest path and A-Star [4] for determining the optimal path, using heuristic techniques. Sampling-based approaches randomly sample and connect points within the surrounding space [5]. Examples of such methods include the probabilistic roadmap method [6] and the rapidly explored random tree method [7]. Interpolation-based methods define a function using multiple coefficients and determine these coefficients [8]. Optimization-based methods formulate an optimization problem with constraints and cost functions to find a path that minimizes the cost. Examples of such methods include the potential field algorithm [9,10] and model predictive control (e.g., [11]).

The potential field algorithm is widely employed for real-time collision-free path planning [12]. In this approach, an attractive force is generated in the direction of the goal, while a repulsive force emanates from obstacles. Despite its frequent use due to its mathematical simplicity, issues such as local minima and oscillations around obstacles have been encountered [12]. Sfeir [13] introduced new formulas to mitigate these oscillations and

prevent conflicts when obstacles are situated near a target. Kim [14,15] proposed a novel framework for escaping local minima in a robot's path using the potential field method.

Path planning using model predictive control (MPC) is being studied as one of the important technologies in mobility systems such as autonomous vehicles and robots. MPC-based path planning solves an optimization problem for a set of future control inputs, creating a stable, collision-free, and efficient path.

However, there are still issues with path generation using MPC. The first issue is how to describe obstacles, and the second is the calculation of the optimization problem in MPC [16]. Recently, attempts have been made to integrate Artificial Potential Fields (APFs) into MPC. Potential fields are used to identify obstacles instead of using inequality constraints [17,18]. To tackle the second issue, optimization techniques have been employed, including the use of genetic algorithms (GAs) [19] or particle swarm optimization (PSO) [20–22]. Zuo [16] conducted a study on vehicle lane changes using MPC, potential fields, and PSO.

When generating a path using MPC and potential field, the decision must be made as to whether to choose a safe or fast path. This is typically adjusted by establishing coefficients for the potential field and MPC's cost function. However, it is not possible to change these assigned coefficients while driving. This paper proposes a method for defining multiple sets of weight coefficients when using MPC. We aim to guide path selection through an additional evaluation using a weight coefficient set appropriate for the surrounding environment.

Additionally, using multiple sets of weight coefficients may cause issues with real-time performance. Our objective is to demonstrate the practical applicability of this method in real-world driving. This will be accomplished by measuring computation times and conducting experiments.

This paper is structured as follows: Section 2 describes the selective MPC-PF-PSO algorithm, including the system configuration, potential field, model predictive path planning, particle swarm optimization, and the cost function used. Section 3 presents the simulation results, which demonstrate the algorithm's effectiveness in autonomous driving scenarios. Section 4 delves into the experimental setup, showcasing the algorithm's performance in real-world driving scenarios.

2. Selective MPC-PF-PSO Algorithm

2.1. System Configuration

By simplifying the steering-type vehicle model to a bicycle model, we can obtain the following dynamic equations:

$$\dot{x_V} = v, \dot{\theta} = \frac{vtan\delta}{L} \tag{1}$$

$$v = R\dot{\theta}, R = \frac{L}{tan\delta} \tag{2}$$

$$\vec{dP}_V = \begin{bmatrix} dx \\ dy \end{bmatrix}_V = \begin{bmatrix} Rsind\theta \\ Rcosd\theta \end{bmatrix} \tag{3}$$

$$\vec{dP}_0 = R_0^v \vec{dP}_V = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} dx_V \\ dy_V \end{bmatrix} \tag{4}$$

x_0 and y_0 refer to the global frame, while x_v and y_v refer to the frame fixed to the vehicle. P_0 is the displacement vector from the global frame to the vehicle frame, while θ represents the rotation angle of the vehicle frame in relation to the global frame. The vehicle has a wheelbase of L, a speed of v, and a front wheel steering angle of delta, which affects the rotation angle θ and results in a turning radius of R as shown in Figure 1.

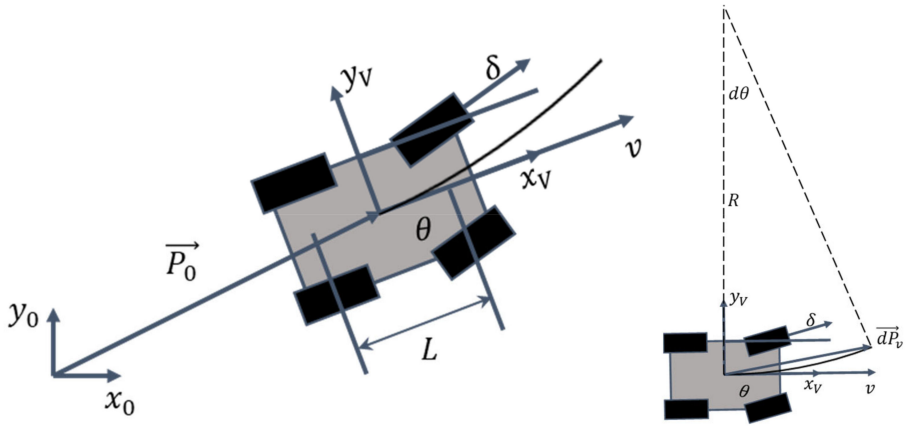


Figure 1. Modeling for steering-type vehicle.

2.2. Potential Field

The potential field algorithm is a method that creates a virtual potential field for obstacles or a goal point, and the robot determines its movement based on this potential field. It was initially proposed by Oussama Khatib [23] in 1986. Due to its simple and intuitive design, it is widely used in the fields of robotics and autonomous driving, and research is continuously being conducted to apply it to actual vehicles [9,24].

Obstacle information in real vehicles is obtained by integrating Lidar and IMU data. The location of the obstacle is then used to calculate the potential field of the area surrounding the vehicle. The potential of each point (U) is calculated based on its distance from the obstacle, using the following equation [25]:

$$U = \begin{cases} 0 & d > D_2 \\ \left(\frac{1}{d} - \frac{1}{D_2}\right)^2 & if\ D_1 < d < D_2 \\ U_{max} & d < D_1 \end{cases} \quad (5)$$

d is the distance between the vehicle and the obstacle, D_1 is the distance at which the vehicle collides with the obstacle, and the potential U is calculated only for obstacles within distance D_2. The potential field, calculated by placing obstacles around the vehicle, is shown in the figure below.

The distance between the vehicle and the obstacle is represented by d, while D_1 represents the distance at which the vehicle collides with the obstacle. The potential is only calculated for obstacles within a distance of D_2. The Figure 2 shows the potential field obtained by placing obstacles around the vehicle.

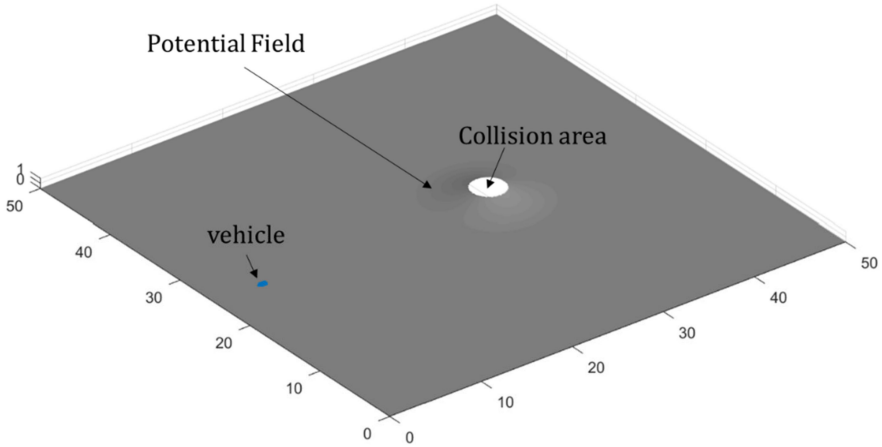


Figure 2. Potential field graph.

2.3. Model Predictive Path Planning

Model predictive control (MPC) is generally used to analytically determine the inputs that optimize cost when planning a path to a destination. However, in real-time systems, numerical optimization techniques are preferred over finding a global optimal input. This paper utilizes MPC to apply the vehicle model and input/output limits, rather than finding the optimal path.

In model predictive path planning, the path is planned by predicting up to T in time intervals dt. The variables at the k-th time step and k + 1-th time step can be expressed using the formulas below, as shown in Equations (1)–(4):

$$\theta(k+1) = \theta(k) + \frac{v tan\delta}{L} dt \tag{6}$$

$$x(k+1) = x(k) + \frac{L}{tan\delta}\left(cos\theta sin\frac{v tan\delta}{L} dt - sin\theta cos\frac{v tan\delta}{L} dt\right) \tag{7}$$

$$y(k+1) = y(k) + \frac{L}{tan\delta}\left(sin\theta sin\frac{v tan\delta}{L} dt + cos\theta cos\frac{v tan\delta}{L} dt\right) \tag{8}$$

$$\delta_{min} < \delta < \delta_{max} \tag{9}$$

$$\dot{\delta}_{min} < \dot{\delta} < \dot{\delta}_{max} \tag{10}$$

where x, y, and θ represent the position and angle of the vehicle. The speed of the vehicle, denoted by v, is assumed to be constant, while the steering angle, δ, is treated as an input to the system.

2.4. Particle Swarm Optimization

PSO is a population-based optimization method that uses a bionic intelligent algorithm. It aims to find the optimal value quickly, using few parameters. The PSO is a method of optimization based on population, where a group of candidate solutions, known as particles, iteratively search for the best solution (fitness) by exchanging information about their discoveries in the search space [26].

Each particle has a position and velocity, and these equations are as follows:

$$V_i^{j+1} = \gamma V_i^j + c_1 rand_1\left(pB_i^j - X_i^j\right) + c_2 rand_2\left(gB^j - X_i^j\right) \tag{11}$$

$$X_i^{j+1} = X_i^j + V_i^{j+1} \tag{12}$$

X_i^j is the position of the i-th particle in the j-th iteration and V_i^j is the velocity of the i-th particle in the j-th iteration. c_1 and c_2 are positive constants, defined as the acceleration coefficients, γ is the inertia weight factor, and $rand_1$ and $rand_2$ are chosen as uniform random values in the range [0:1].

The position of the i-th particle in the j-th iteration is represented by X_i^j, and its velocity is represented by V_i^j. The acceleration coefficients, c_1 and c_2, are positive constants. The inertia weight factor is represented by γ, while $rand_1$ and $rand_2$ are chosen as uniform random values in the range [0:1].

For model predictive path planning, particles are defined for the time range [0:dt:T]. The global best particles are found through iteration. The best particle is expressed as X_{ik}^j, which represents the steering angle of the i-th particle at the k-th time step in the j-th iteration. Similarly, V_{ik}^j denotes the angular rate of the steering of the i-th particle at the k-th time step in the j-th iteration [27].

2.5. Cost Function for Selectiveness

The cost function for obtaining an optimal output through PSO is typically defined in order to find the shortest path while avoiding collisions or to follow a given path as closely as possible while avoiding collisions. Optimization can be achieved by using the cost function and cost map obtained from the potential field to balance safe driving and the shortest route. The equation for the cost function f can be expressed as follows:

$$f\left(X_i^j(t)\right) = w_s U\left(X_i^j(t)\right) + w_d D\left(X_i^j(t)\right) + w_u u(t) \tag{13}$$

where the weights for safety, distance from the global path, and input are represented by w_s, w_d, and w_u respectively. By adjusting these weights, a balance can be achieved between distance from obstacles, adherence to the global path, and the magnitude of steering input changes.

This approach prevents unnecessary avoidance in sparsely populated obstacle spaces and reduces the risk of generating dangerous paths in areas with numerous obstacles. To ensure optimal obstacle avoidance, it is recommended that multiple sets of coefficients $\mathrm{W} = [w_s,\ w_d,\ w_u]$ are defined and optimized accordingly.

The global best (gB) is obtained for each coefficient set W. The cost function can be formulated as follows, utilizing the r-th $gB(gB_r)$, the maximum potential value $\mathrm{Max}(U)|_{gB_r}$, and the distance between the final position of gB_r and the destination point $D_{goal}(gB_r)$.

$$g(gB_r) = k_1 \mathrm{Max}(U)|_{gB_r} + k_2 D_{goal}(gB_r) \tag{14}$$

where choosing the gB_r with a minimum $g(gB_r)$ as the final path will allow us to avoid the problems mentioned above.

3. Simulation

To achieve autonomous driving using the MPC-PF-PSO algorithm, it is crucial to prioritize obstacle avoidance and path maintenance by adjusting the tunable variables which are shown in Table 1.

Table 1. Simulation variables.

Variables	
γ	Inertia of V_i^j in PSO
c_1	Weight of particle best in PSO
c_2	Weight of global best in PSO
w_s	Weight of safety in cost function
w_d	Weight of distance from global path in cost function
w_u	Weight of input in cost function

Pseudo code of the MPC-PF-PSO algorithm is as shown as Algorithm 1. When the vehicle is located at position (5, 25) and its destination is set at (30, 25), the resulting graphs for M = 1, 10, and 30 are shown in the Figure 3. The parameters used were $\gamma = 0.95$, $c_1 = 0.333$, $c_2 = 0.5$, T = 3, dt = 0.2, and N = 40. The bold red line is the optimal path, and other colored lines are path candidates.

Algorithm 1: Pseudo code of the MPC-PF-PSO algorithm

initialize cost map using Potential Field as shown in Figure 2
costmap(x, y) = U(d)

T—prediction time
dt—time step
N—Number of Particles
M—Maximum number of iteration

for all $W_r = [w_s,\ w_d,\ w_u]_r$,

gB^j = Max value
pB^j = Max value

initialize particles randomly
for i = 1: N
$X_i^0 \leftarrow$ a random vector
$V_i^0 \leftarrow$ a random vector
apply Equations (6)–(8) to calc x, y, θ
Calc cost $f(X_i^0)$
if $f(X_i^0) < f\left(pB_i^0\right), pB_i^0 \leftarrow X_i^0$
if $f(X_i^0) < f\left(gB^0\right), gB^0 \leftarrow X_i^0$
end for

while j < M
for i = 1: N
apply Equations (11) and (12) to calc $X_i^j,\ V_i^j$
apply Equations (6)–(8) to calc x, y, θ
Calc cost $f(X_i^0)$
if $f\left(X_i^j\right) < f\left(pB_i^j\right), pB_i^j \leftarrow X_i^j$
if $f\left(X_i^j\right) < f\left(gB^j\right), gB^j \leftarrow X_i^j$
end for
end while
Calc cost $g(gB_r)$
end for
Choose argmin$(g(gB_r))$ as a final path

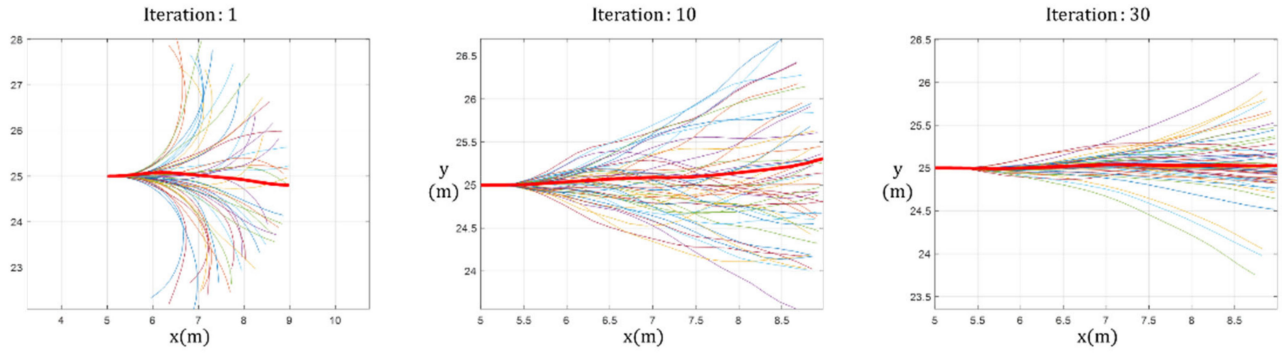


Figure 3. Path change according to number of iterations performed.

If the number of iterations is set to 1, particles are randomly generated and the optimal path is then selected from them. As the number of iterations increases, particles converge to the optimal path through PSO, and the optimal path is selected from them.

When an obstacle is encountered like Figure 4, the path generated will vary depending on the values of $W_r = [w_s,\ w_d,\ w_u]_r$. The resulting paths when applying $W_1 = [0.2,\ 0.5,\ 0]$

and $W_2 = [1.0,\ 0.5,\ 0]$ are shown Figure 5. The red line is the optimal path when W1 is applied, and the blue line is the optimal path when W2 is applied. Other colored lines are path candidates.

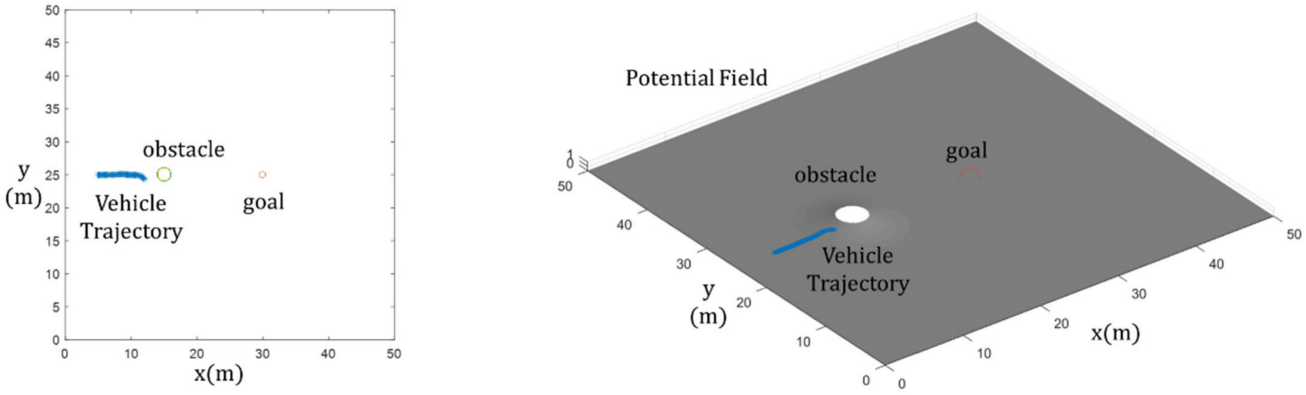
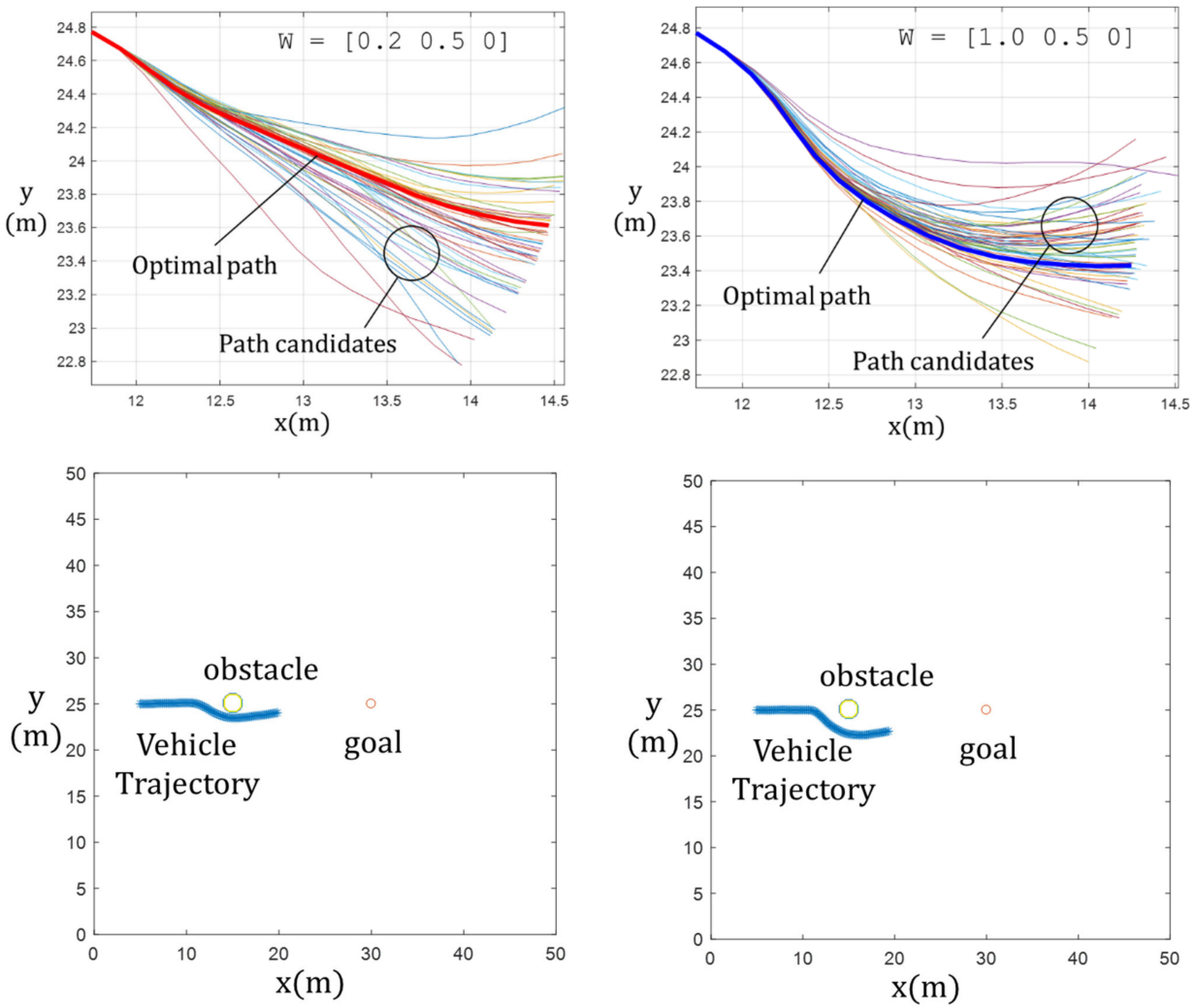


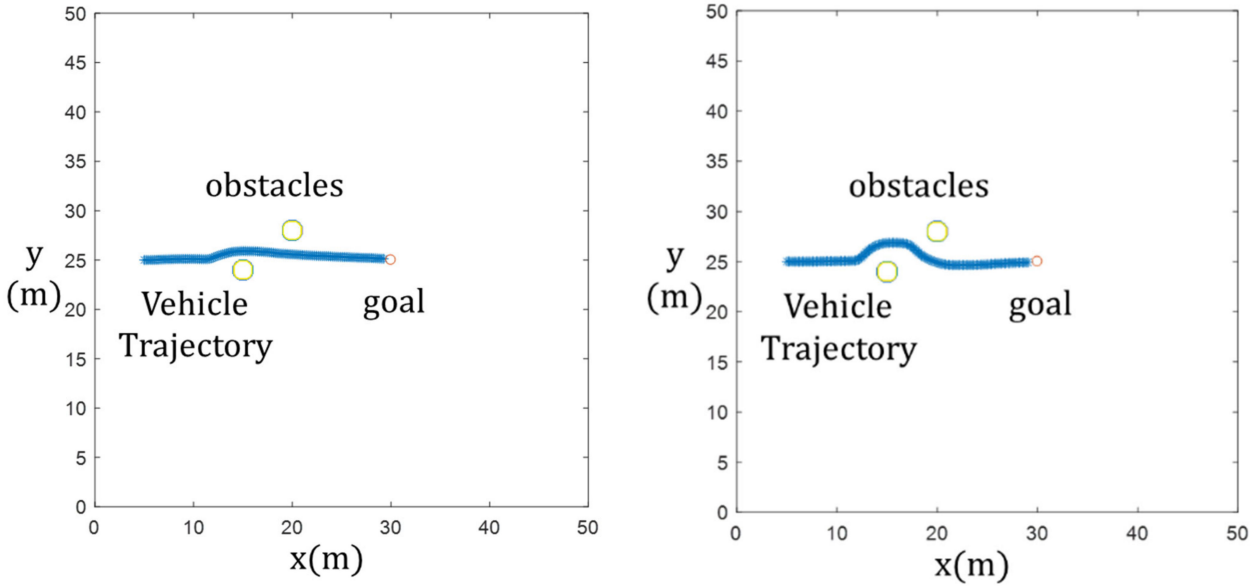
Figure 4. Path generation for obstacle avoidance.



(**a**) generated path when $W_1 = [0.2,\ 0.5,\ 0]$ (**b**) generated path when $W_2 = [1.0,\ 0.5,\ 0]$

Figure 5. Generated paths, according to the coefficient W, for one obstacle.

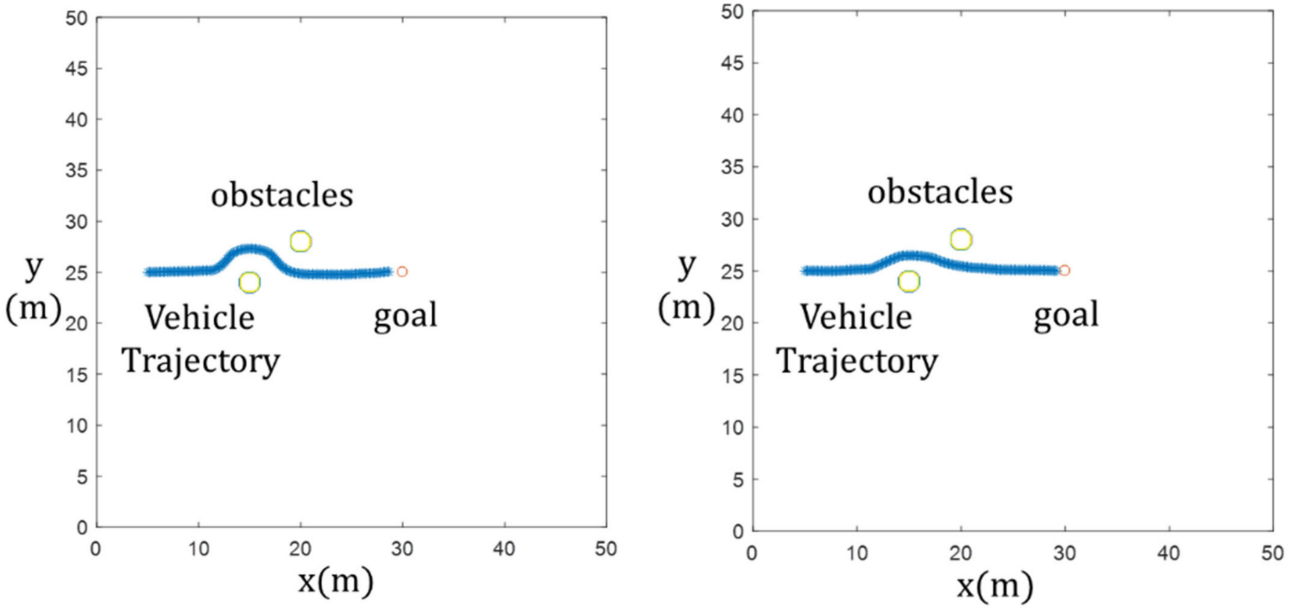
As w_s increases, the algorithm generates a path that avoids obstacles further away. Conversely, decreasing w_s results in a path that approaches obstacles more closely. When there are two obstacles, the algorithm creates a path as shown Figure 6.



(**a**) generated path when $W_1 = [0.2,\ 0.5,\ 0]$ (**b**) generated path when $W_2 = [1.0,\ 0.5,\ 0]$

Figure 6. Generated paths, according to the coefficient W, for two obstacles.

Five sets of W were defined using $W_r = [1.5 - 0.2r,\ 0.5,\ 0]$, each with different w_s values. The generated paths are illustrated in the Figure 7, where $K = \begin{bmatrix} k_1 & k_2 \end{bmatrix}$ in Equation (11) is defined as $K = \begin{bmatrix} 1.5 & 0.5 \end{bmatrix}$ and $K = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}$.

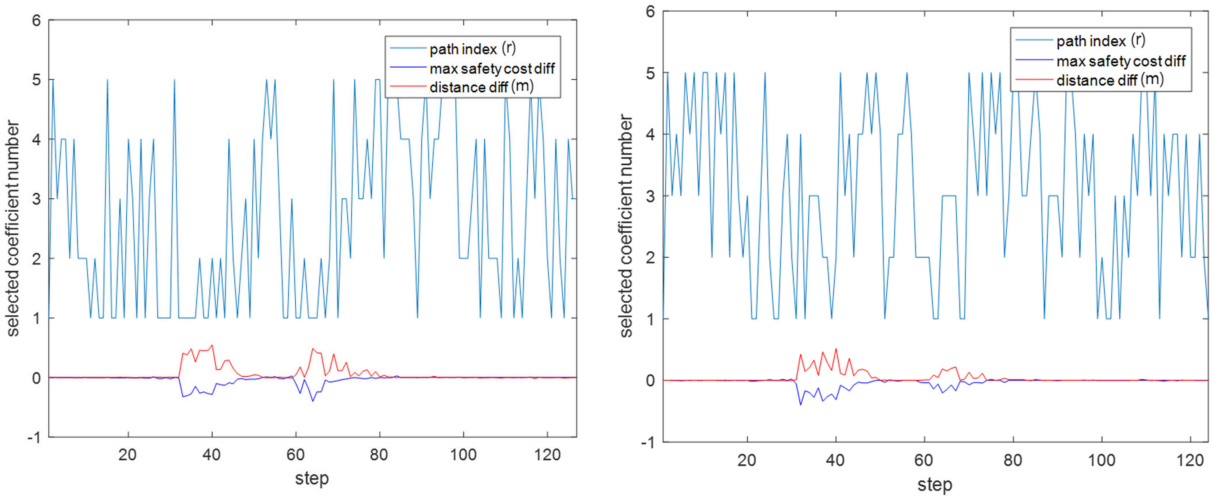


(**a**) generated path when $K = [1.5 \quad 0.5]$ (**b**) generated path when $K = [0.5 \quad 0.5]$

Figure 7. Generated paths, according to the coefficient K, for two obstacles.

The figure below shows the r values of the final selected Wr, and the difference in $\mathrm{Max}(U)|_{gB_r}$ and $D_{goal}(gB_r)$, when r = 1 and r = 5 at each step until the vehicle reaches its destination.

In the Figure 8, variations occur in the values of $\mathrm{Max}(U)|_{gB_r}$ and $D_{goal}(gB_r)$ when encountering the first and second obstacles, and the coefficient K begins to influence the path selection. In case (a), a long-distance route is chosen for safety when encountering an obstacle. Once the obstacle has been passed, the shortest route to the destination is chosen. In case (b), although a safer path is selected when an obstacle is encountered, there is a higher tendency to choose the shortest path to the global path.

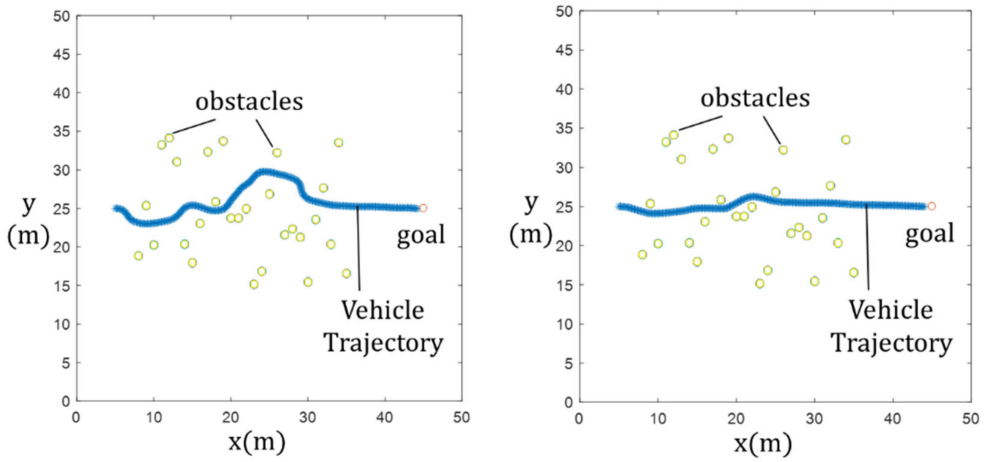


(**a**) selected index of W K = [1.5 0.5] (**b**) generated path using K = [0.5 0.5]

Figure 8. Graphs of selected coefficient index r and difference between $\mathrm{Max}(U)|_{gB_r}$ and $D_{goal}(gB_r)$.

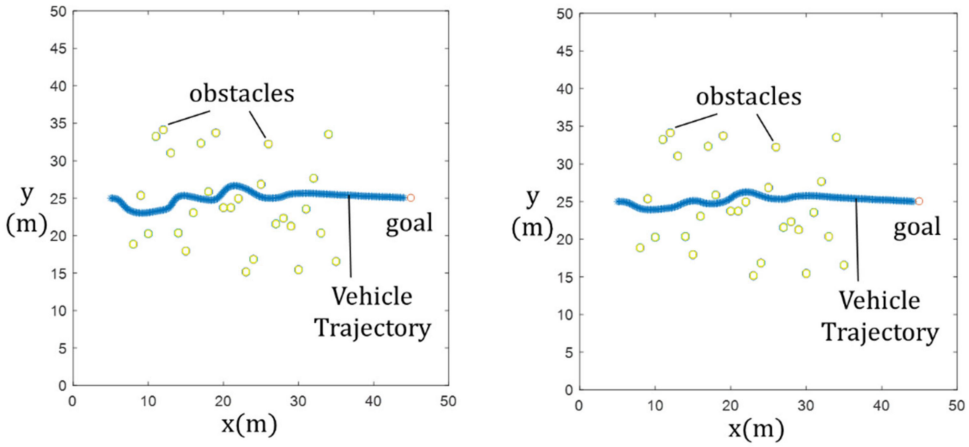
This simulation was performed on a PC equipped with an Intel® Core™ i7-8559U CPU @ 2.7 GHz and 16 GB RAM. 124 time steps were calculated before the vehicle reached its destination. The constants related to MPC that have a significant impact on computation time were set as follows: T = 3, dt = 0.2, N = 40, and M = 30. After calculating the average time taken, it was found that one time step took approximately 33 ms.

When driving with only one set of W in an area with complex obstacles, the vehicle may excessively avoid obstacles, as shown in (a) of Figure 9, or avoidance may occur due to it being too close to obstacles, even in non-dense areas, as shown in (b). However, by defining five sets of W and creating paths through adjusting the constant K, it is possible to avoid obstacles that are far away in less dense areas, as shown in (c) and (d). Additionally, it is possible to achieve a close avoidance of obstacles in dense areas during path generation.



(**a**) generated path when W = [1.5, 0.5, 0] (**b**) generated path when W = [0.5, 0.5, 0]

Figure 9. *Cont.*



(**c**) generated path when K = [1.5 0.5] (**d**) generated path when K = [0.5 0.5]

Figure 9. Generated paths in a multiple-obstacle area.

4. Experiment

A steering-type vehicle was equipped with an RTK GPS for localization, LiDAR for obstacle recognition, and NVIDIA's Jetson AGX Xavier for executing driving control. The appearance of the vehicle is as shown in Figure 10.

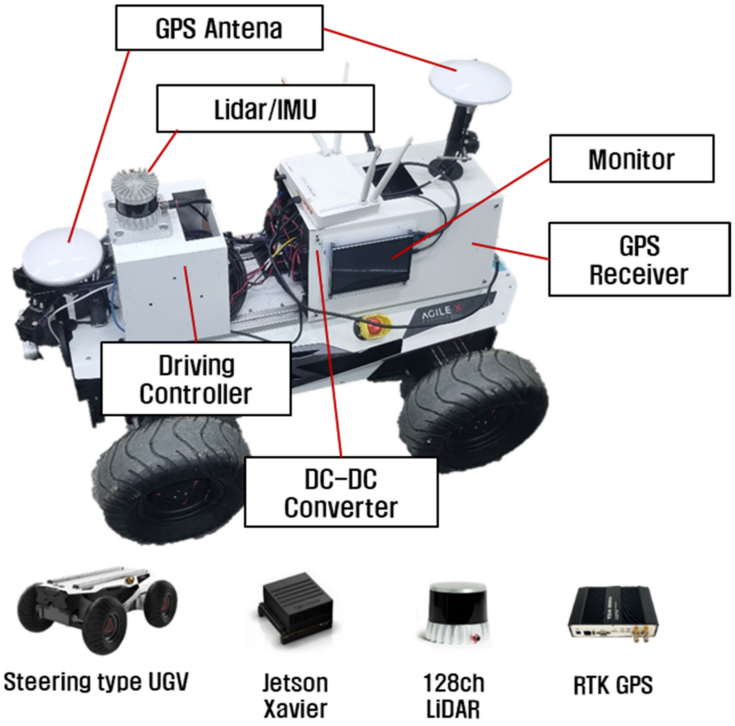


Figure 10. Vehicle for driving experiment.

Using LiDAR, the point cloud information of the surrounding obstacles was acquired. Subsequently, a 2D grid map was generated from that point cloud information, and a grid map with a calculated potential field was obtained as shown in Figure 11. After that, a global path was created, in a straight line, to the destination, and a local path was generated using the MPC-PF-PSO algorithm.

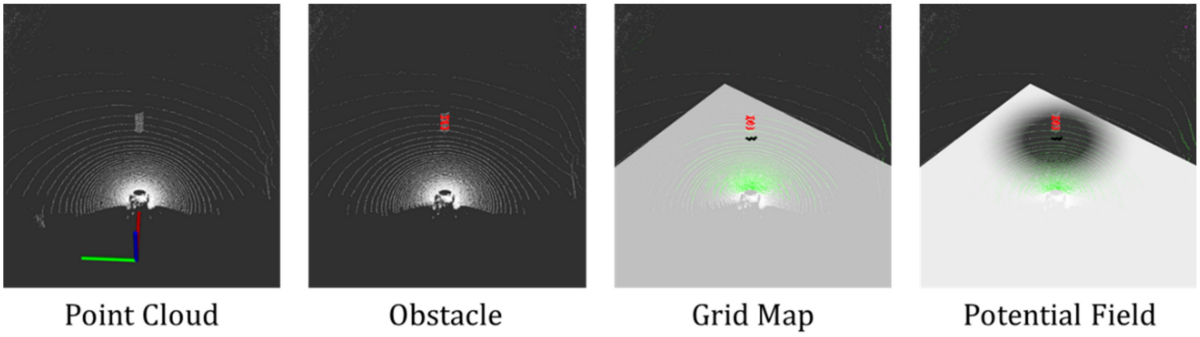


Figure 11. From point cloud to potential field grid map.

Computation time is a crucial factor in obstacle avoidance during navigation. We have implemented the MPC-PF-PSO algorithm and a package that computes the potential field of an obstacle point cloud in an ROS Melodic environment. We measured the computation time used by each package.

The MPC-PF-PSO-related constants were defined as T = 3, dt = 0.2, N = 40, M = 30, and 5 sets of W. It took 7 ms to generate a 2D grid map containing potential field information, and the MPC-PF-PSO algorithm's operation took 24 ms as shown in Figure 12.

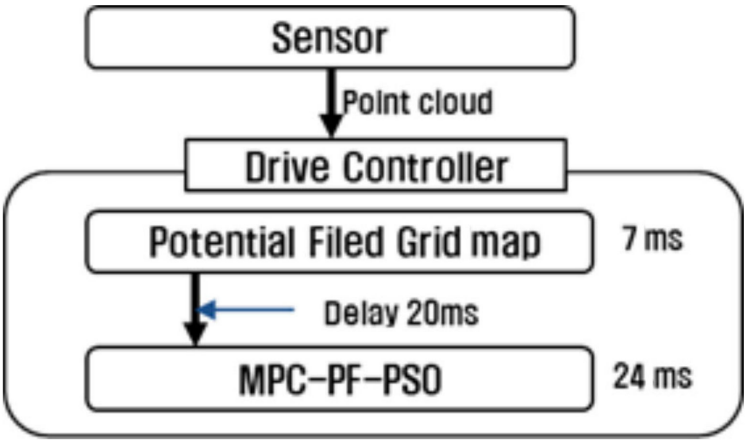


Figure 12. Computation times for each package.

A driving experiment was performed by applying the MPC-PF-PSO algorithm. Obstacles were placed, using bricks, and a global path was established in a straight line to the destination point as shown in Figure 13. Experiments were conducted by altering the constant K. When the coefficient k_1 (for the maximum potential $\mathrm{Max}(U)|_{gB_r}$) was relatively larger than the coefficient k_2 (for the distance to the destination $D_{goal}(gB_r)$), a path was generated that avoided the obstacle further, as confirmed in the simulation.

The driving vehicle and data visualization for the cases of K = [0.5 0.5] and K = [1.5 0.5] are shown in Figure 14. We measured the distance using log data to see how far the vehicle traveled from the obstacle, and the results are shown in the Figure 15. For K = [0.5, 0.5] and K = [1.5, 0.5], the shortest distances between the obstacle and the global path were 0.44 m and 0.62 m, respectively. The shortest distances between the obstacle and the vehicle were 1.03 m and 1.25 m, respectively. For K = [0.5, 0.5], even though a distance from the global path to the obstacle were closer than that of K = [1.5, 0.5], the vehicle drove closer to the obstacle.

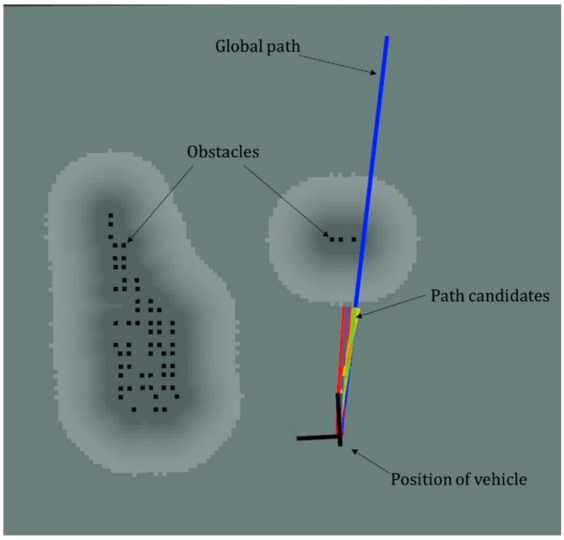
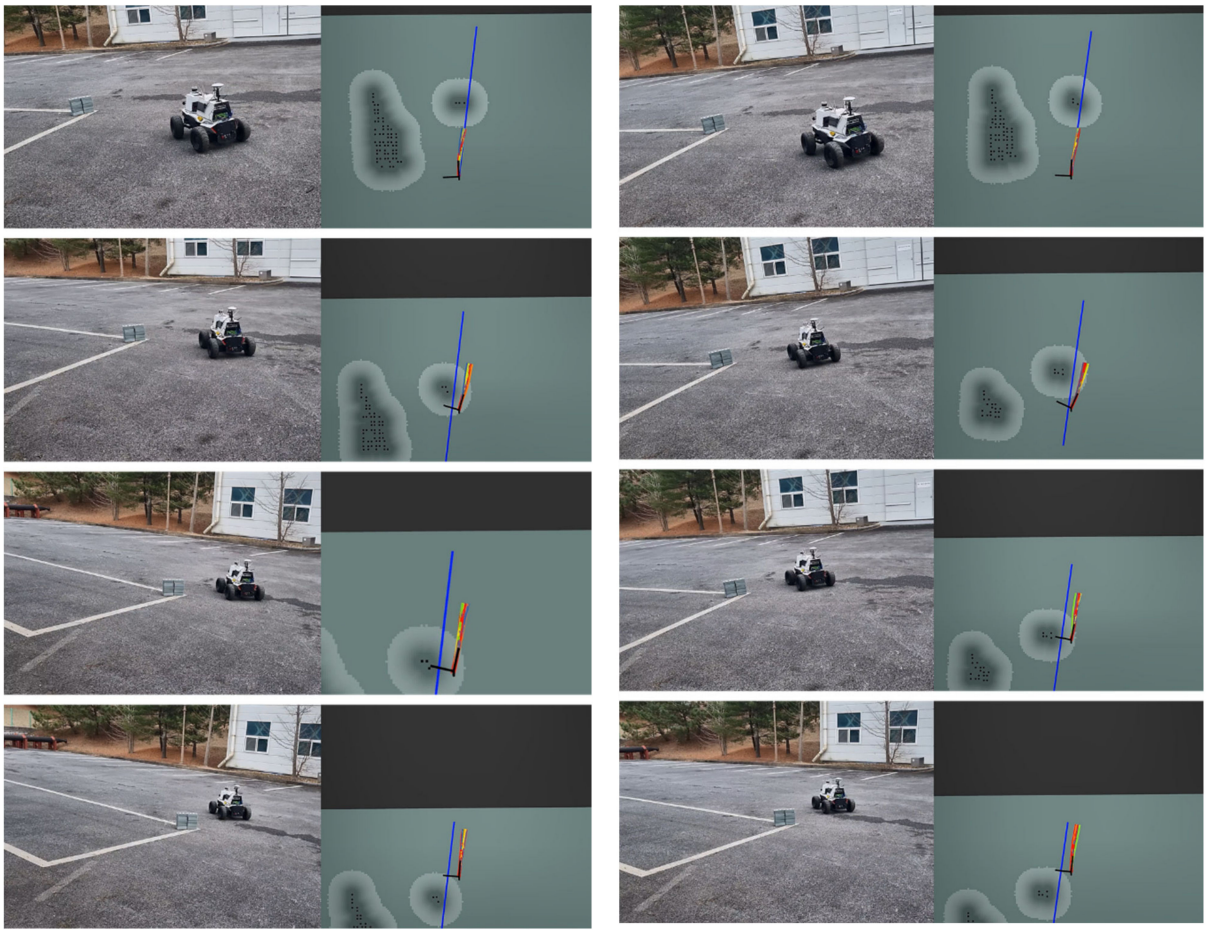
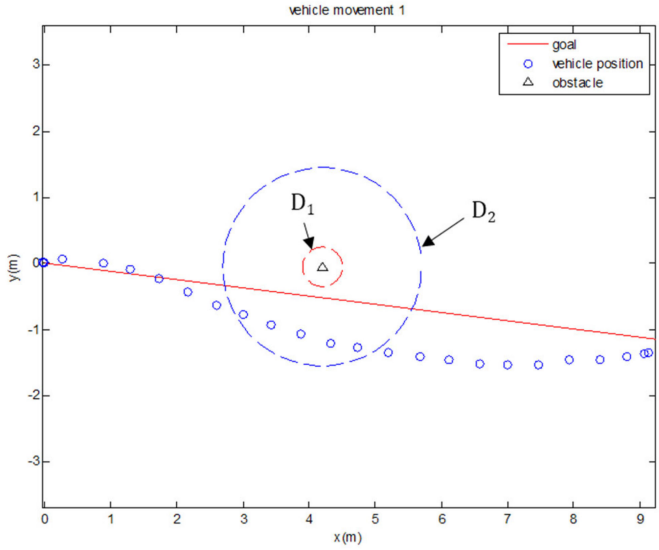


Figure 13. Visualized data during obstacle avoidance experiments.

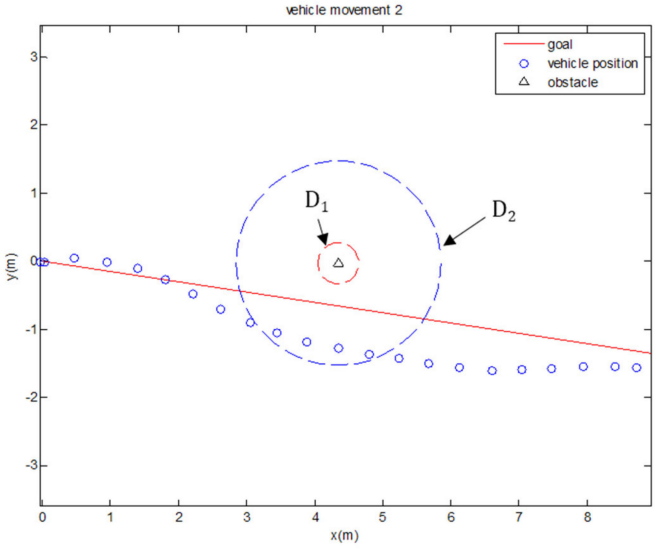


(**a**) Experiment with $K = [0.5 \quad 0.5]$ (**b**) Experiment with $K = [1.5 \quad 0.5]$

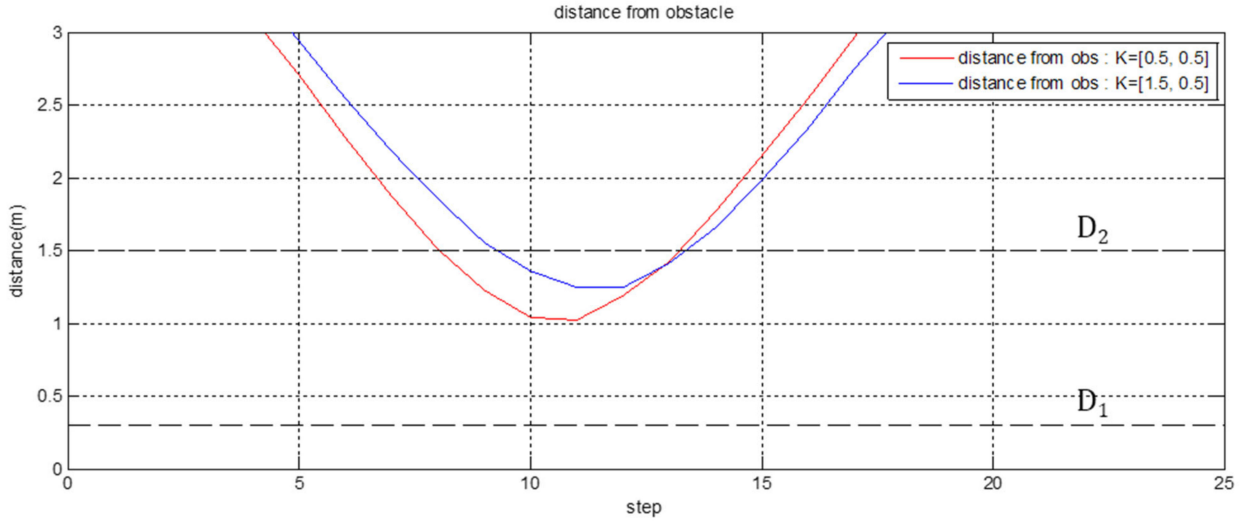
Figure 14. Obstacle avoidance experiments with different K coefficients.



(**a**) Experiment with $K = [0.5 \quad 0.5]$



(**b**) Experiment with $K = [1.5 \quad 0.5]$



(**c**) Distance from obstacle with different K coefficients.

Figure 15. Results of obstacle avoidance experiments.

5. Discussion

In this paper, our focus was on developing an algorithm that can be applied in practical scenarios, not only for obstacle avoidance but also for real-time decision making to ensure safe navigation. We applied a potential field to assess safety, used MPC to consider the characteristics of the actual moving vehicle, and employed the PSO algorithm to ensure real-time responsiveness. Additionally, we introduced a method for utilizing various coefficient sets to determine whether to select a safe path or a fast path according to the surrounding environment. The algorithm was implemented through ROS on Jetson Xavier, which was released by NVIDIA. Through computational time measurements and experiments, we demonstrated that this algorithm is suitable for practical use.

While the proposed method is suitable for real-time applications, a challenge lies in the low consistency of its path generation, attributed to the utilization of randomness at each step of finding the optimal path. Future research will focus on enhancing path consistency, considerations for dynamic obstacles, and developing path generation methods for irregular terrains.

Author Contributions: Conceptualization and methodology: M.K. and M.L.; formal analysis and software: M.K. and B.K.; data curation, writing—original draft, and writing—review and editing: M.K.; validation: M.L. and M.C.; supervision, project administration, and funding acquisition: M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Research Project of KIMM (No. NK242I), the National Research Foundation of Korea (NRF) through a grant funded by the Korean government (MSIT) (No. 2020M3C1C1A02086421), and the National Research Council of Science & Technology (NST) grant by the Korea government (MSIT) (No. CRC23041-400).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. González, D.; Pérez, J.; Milanés, V.; Nashashibi, F. A review of motion planning techniques for automated vehicles. *IEEE Trans. Intell. Transp. Syst.* **2015**, *17*, 1135–1145. [CrossRef]
2. Lavalle, S.M. *Planning Algorithms*; Cambridge University Press: New York, NY, USA, 2006.
3. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
4. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
5. Elbanhawi, M.; Simic, M. Sampling-based robot motion planning: A review. *IEEE Access* **2014**, *2*, 56–77. [CrossRef]
6. Kavraki, L.E.; Svestka, P.; Latombe, J.-C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
7. LaValle, S.M.; Kuffner, J.J. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
8. Brezak, M.; Petrovic, I. Real-time approximation of clothoids with bounded error for path planning applications. *IEEE Trans. Robot.* **2014**, *30*, 507–515. [CrossRef]
9. Koren, Y.; Borenstein, J. Potential field methods and their inherent limitations for mobile robot navigation. In Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, USA, 9–11 April 1991.
10. Kim, K. Predicted Potential Risk-Based Vehicle Motion Control of Automated Vehicles for Integrated Risk Management. Ph.D. Thesis, Seoul National University, Seoul, Republic of Korea, 2016.
11. Kim, J.; Pae, D.; Lim, M. Obstacle Avoidance Path Planning based on Output Constrained Model Predictive Control. *Int. J. Control Autom. Syst.* **2019**, *17*, 2850–2861. [CrossRef]
12. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. Heuristic approaches in robot path planning: A survey. *Robot. Auton. Syst.* **2016**, *86*, 13–28. [CrossRef]
13. Sfeir, J.; Saad, M.; Hassane, H.S. An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment. In Proceedings of the 2011 IEEE International Symposium on Robotic and Sensors Environments (ROSE), Montreal, QC, Canada, 17–18 September 2011.
14. Kim, D.H. Escaping route method for a trap situation in local path planning. *Int. J. Control Autom. Syst.* **2009**, *7*, 495–500. [CrossRef]
15. Kim, D.H.; Shin, S. Local path planning using a new artificial potential function configuration and its analytical design guidelines. *Adv. Robot.* **2006**, *20*, 115–135. [CrossRef]
16. Zuo, Z.; Yang, X.; Wang, Y.; Han, Q.; Wang, L.; Luo, X. MPC-based cooperative control strategy of path planning and trajectory tracking for intelligent vehicles. *IEEE Trans. Intell. Veh.* **2020**, *6*, 513–522. [CrossRef]
17. Ji, J.; Khajepour, A.; Melek, W.W.; Huang, Y. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. *IEEE Trans. Veh. Technol.* **2016**, *66*, 952–964. [CrossRef]
18. Huang, Z.; Wu, Q.; Ma, J.; Fan, S. An APF and MPC combined collaborative driving controller using vehicular communication technologies. *Chaos Solitons Fractals* **2016**, *89*, 232–242. [CrossRef]
19. Du, X.; Htet, K.K.K.; Tan, K.K. Development of a genetic-algorithm-based nonlinear model predictive control scheme on velocity and steering of autonomous vehicles. *IEEE Trans. Ind. Electron.* **2016**, *63*, 6970–6977. [CrossRef]
20. Van Heerden, K.; Fujimoto, Y.; Kawamura, A. A combination of particle swarm optimization and model predictive control on graphics hardware for real-time trajectory planning of the under-actuated nonlinear acrobot. In Proceedings of the 2014 IEEE 13th International Workshop on Advanced Motion Control (AMC), Yokohama, Japan, 14–16 March 2014.
21. Thomas, J. Particle swarm optimization based model predictive control for constrained nonlinear systems. In Proceedings of the 2014 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Vienna, Austria, 2–4 September 2014.
22. Zuo, Z.; Dai, L.; Wang, Y.; Yang, X. Fast nonlinear model predictive control parallel design using QPSO and its applications on trajectory tracking of autonomous vehicles. In Proceedings of the 2018 13th World Congress on Intelligent Control and Automation (WCICA), Changsha, China, 4–8 July 2018.
23. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [CrossRef]

24. Kim, M.; Yoo, S.; Lee, D.; Lee, G. Local path-planning simulation and driving test of electric unmanned ground vehicles for cooperative mission with unmanned aerial vehicles. *Appl. Sci.* **2022**, *12*, 2326. [CrossRef]
25. Choset, H. *Robotic Motion Planning: Potential Functions*; Robotics Institute, Carnegie Mellon University: Pittsburgh, PA, USA, 2010.
26. Shin, J.; Kwak, D.; Kwak, K. Model predictive path planning for an autonomous ground vehicle in rough terrain. *Int. J. Control Autom. Syst.* **2021**, *19*, 2224–2237. [CrossRef]
27. LI, X.; Wu, D.; He, J.; Bashir, M.; Liping, M. An improved method of particle swarm optimization for path planning of mobile robot. *J. Control Sci. Eng.* **2020**, *2020*, 3857894. [CrossRef]