applied sciences

MDPI

Article

Path Planning Based on Artificial Potential Field with an Enhanced Virtual Hill Algorithm

Hyun Jeong Lee [1], Moon-Sik Kim [2,*] and Min Cheol Lee [1,*]

[1] School of Mechanical Engineering, Pusan National University, Pusan-si 46241, Republic of Korea; lhjeong98@gmail.com
[2] Department of Intelligent Mobility Engineering, Kongju National University, Cheonan-si 31080, Republic of Korea
* Correspondence: mskim2@kongju.ac.kr (M.-S.K.); mclee@pusan.ac.kr (M.C.L.)

Abstract: The artificial potential field algorithm has been widely applied to mobile robots and robotic arms due to its advantage of enabling simple and efficient path planning in unknown environments. However, solving the local minimum problem is an essential task and is still being studied. Among current methods, the technique using the virtual hill concept is reliable and suitable for real-time path planning because it does not create a new local minimum and provides lower complexity. However, in the previous study, the shape of the obstacles was not considered in determining the robot's direction at the moment it is trapped in a local minimum. For this reason, longer or blocked paths are sometimes selected. In this study, we propose an enhanced virtual hill algorithm to reduce errors in selecting the driving direction and improve the efficiency of robot movement. In the local minimum area, a dead-end algorithm is also proposed that allows the robot to return without entering deeply when it encounters a dead end.

Keywords: mobile robots; artificial potential field; local minima problem; enhanced virtual hill; dead-end algorithm

Citation: Lee, H.J.; Kim, M.-S.; Lee, M.C. Path Planning Based on Artificial Potential Field with an Enhanced Virtual Hill Algorithm. *Appl. Sci.* **2024**, *14*, 8292. https://doi.org/10.3390/app14188292

Academic Editor: Christos Bouras

Received: 27 August 2024
Revised: 9 September 2024
Accepted: 12 September 2024
Published: 14 September 2024

1. Introduction

Unlike complex global path planning [1], which requires a vast amount of prior information, the local method allows for path planning, including obstacle avoidance, in an unknown environment with a small amount of computation. Local path planning using the APF (artificial potential field) technique has been widely used in real-time path planning due to its easy implementation and high computational efficiency [2–9]. APF is formed by the sum of the attractive potential field to reach the destination and the repulsive potential field to avoid obstacles [10]. Due to the potential at each point, the robot travels to the destination without colliding. In this process, when the sum of potentials approaches '0', local minima may occur, in which the robot stops operating [11,12]. In order to apply the APF technique, this problem must be solved, and various techniques have been studied so far. There are methods of modifying the potential field function or the repulsive force function [6,13–15] or adding vortex fields or local attractors [16–18]. In [17], vortex fields prevent collisions between obstacles and robots and enable smooth running when the robot moves through narrow corridors. However, it is difficult to determine when to apply vortex potential for optimal performance, and it may not be effective in complex environments. References [7,19] propose a short and optimal path planning method using bacterial foraging optimization. In [7], virtual particles are initially moved randomly, the cost is calculated at each location, the best particle is found, and the robot's driving path is created. The optimal path can be found even in complex environments. Reference [19] shows that when a robot is trapped in a local minimum, a virtual obstacle suitable for the location condition is created, and the robot is guided to move to a new path. However, both of these techniques require an iterative learning process to optimize the path and virtual

obstacles. In the bacterial potential field method [20], the problems of the APF are solved by applying the bacterial evolutionary algorithm (BEA) in addition. There are also methods to plan the optimal path while solving the local minimum using simulated annealing (SA), a genetic algorithm (GA), or reinforcement learning [21–24]. Reference [24] introduces an algorithm that converges to the optimal path using reward and value functions. The optimal path can be found through automatic learning without prior information. However, since it starts randomly, it can be unstable and requires a lot of calculations to optimize the path. References [25,26] propose new algorithms to be combined with APF. In [26], the improved A* algorithm is fused with APF, and search accuracy and efficiency are improved using a heuristic method. To escape the local minimum, a virtual target point is set in [27,28], and a grid-based obstacle cell is used in [29,30]. In [30], efficient computation is possible using a discrete grid, and local minima can be avoided by creating a new repulsive potential between adjacent obstacles. However, if the resolution of the grid is low, precise path planning is difficult, and if the resolution is high, the amount of calculation increases. A perpendicular approach based on APF has been proposed as a local path plan for unknown obstacles along with a global path plan performed before the departure of the robot [31]. This method can solve the local minima problem with a relatively short calculation time, but it is difficult to apply in an unknown environment because it requires prior information about obstacles.

Recently, research on learning-based optimization techniques has been introduced to overcome the shortcomings of APF. These methods can find the optimal path for robots even in complex environments and efficiently avoid and escape local minima. However, there are issues such as the robot being unstable at the beginning of the learning process, results varying depending on the initial conditions, and high computational load. Additionally, in unknown environments, the robot's responsiveness is reduced until the learning process is completed, making immediate path planning difficult. On the other hand, the artificial potential field algorithm with a virtual hill used in this paper has the advantage of being able to avoid obstacles, escape local minima, and reach the destination without a learning process in unknown environments.

In [32], a method of escaping from the local minimum using a virtual hill is introduced. The virtual hill generates extra force instead of attractive force to repel the robot from a local minimum. The virtual hill technique does not require prior information about space and obstacles, modeling [33] and learning [23,24] processes for generating potential fields may be omitted, and a new local minimum is not created. It also has the advantage of being applicable to dynamic environments and being easy to implement. And in most cases, a robot can escape from the local minimum. This is an easy and safe way for a robot to move to a destination while avoiding obstacles in an unknown environment without being trapped in a new local minimum. The virtual hill technique is a method of moving along the outline of the obstacle that caused the local minimum. The robot determines which direction to follow among the left and right obstacles to drive as soon as it is trapped in the local minimum. Efficient and reliable path planning is possible if the robot selects the obstacle that leads to the shortest distance. In the previous study [32], the direction of movement was selected only in the relative positional relationship between the robot, the closest obstacle, and the destination without considering the shape of the path. As a result, the robot may go a long distance or choose a dead end. Therefore, there is a need to reduce errors in the selection of the direction of driving regardless of obstacle shape and to improve the efficiency of the robot's movement.

This paper proposes 'the enhanced virtual hill' technique, a method of determining the driving direction that considers the shape of surrounding obstacles. At the moment the robot is trapped in a local minimum, the distance to obstacles is measured, and it is determined which of the two directions is the more open path. The driving efficiency of the robot can be improved by blocking closed-path driving in advance. Additionally, a dead-end algorithm is proposed that allows the robot to come back without going deep

when it encounters a dead end. Through simulations, it is confirmed that the robot's moving distance is reduced when the proposed methods are applied.

2. Simulation Environments

For evaluating the path planning algorithm, a mobile robot model, LiDAR sensor, visualization function, and maps provided by MATLAB 2022b's Mobile Robotics Simulation Toolbox are utilized. Figure 1a,b shows the directions of the linear velocity and angular velocity of the robot and the environment of the Mobile Robotics Simulation Toolbox, respectively. The forward/inverse geometry of the robot is provided, and the position, posture, movement path, sensor measurement distance(the dashed blue lines), etc. of the robot are shown through the visualization function, as in Figure 1b. For the simulation, the radius of the wheel of the mobile robot is set to 0.1 [m], and wheelbase, which is the distance from the left wheel to the right wheel, is set to 0.5 [m]. The local coordinate system of the robot is fixed to the center of the robot. In the simulation environment, the LiDAR sensor is configured in a radial form, with the center of the robot as the origin. According to the user settings, the robot obtains obstacle information of up to 4 m at 19 points through the LiDAR sensor. LiDAR sensors are widely used to create maps or model and track objects using numerous measurement points. However, in this paper, since large amounts of data in the form of a cloud are not required, the LiDAR sensor is simulated by a low-cost ultrasonic sensor widely used in mobile robots. In the reference paper [32], 5–7 robot skeleton points are set, and the distance from each point to the obstacle is calculated and simulated. Although this paper also uses the LiDAR sensor supported by MATLAB's Toolbox, it receives distance data by dividing 360 degrees into 19 points as if it were equipped with 19 ultrasonic sensors. Computation time can be saved by reducing the amount of processed data. The maximum measurement distance is set to 4 m. This study targets mobile robots that provide services in offices or commercial facilities. Accordingly, 4 m is considered the distance that people can feel as an open space in general indoor office environments. At too far a distance, the spacing between each sensor data increases, making it inappropriate to treat it as information about continuous obstacles. Reference [32] is referred to for the path planning and control process of the robot. All simulations are saved as figures, data files, and videos.

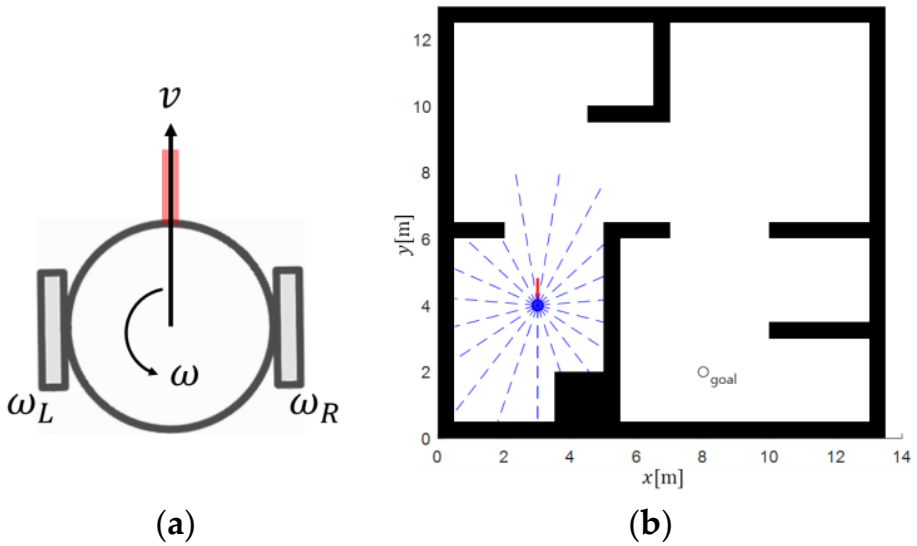


Figure 1. MATLAB 2022b simulation environment: (**a**) mobile robot model, linear velocity v, angular velocity ω; (**b**) visualizer of the Mobile Robotics Simulation Toolbox.

3. Virtual Hill Concept and Open Path Indicator, *New* e_b

The virtual hill concept allows the robot to escape from the local minimum by generating extra force instead of attractive force when in the local minimum area. As explained in

Section 3.1, the extra force follows the direction of the unit tangent vector $\mathbf{e}_t$ of γ, which represents the trajectory of the nearest obstacle while the robot is moving within the local minimum. The unit tangent vector is determined as the cross product of the unit normal vector $\mathbf{e}_n$ and the unit binormal vector $\mathbf{e}_b$. Since $\mathbf{e}_n$ is the vector between the nearest obstacle and the robot, the decisive factor that determines the robot's driving direction is $\mathbf{e}_b$. Unlike the previous $\mathbf{e}_b$ determined according to the relative positions of the robot, the closest obstacle, and the goal, the *new* $\mathbf{e}_b$ proposed in this paper operates as an open path indicator. *new* $\mathbf{e}_b$, which is determined by the shape of surrounding obstacles, makes the robot travel on a more open path, improving path efficiency.

3.1. Virtual Hill Concept

APF can be expressed as the sum of the attractive potential corresponding to the goal and the repulsive potential generated by the obstacle [10]. The artificial forces that determine the control output of the robot, that is, the linear velocity and angular velocity, are the negative gradients of the potentials [10,12]. The attractive potential U_{att}, force $\mathbf{F}_{att}$, and the repulsive potential U_{rep}, force $\mathbf{F}_{rep}$, are obtained as follows.

$$U_{att} = \begin{cases} k_a d^2, & d \le d_0 \\ k_a\left(2d_0 d - d_0^2\right), & d > d_0 \end{cases} \tag{1}$$

$$\mathbf{F}_{att} = -\nabla U_{att} = \begin{cases} -2k_a\left(\mathbf{P} - \mathbf{P}_{goal}\right), & d \le d_0 \\ -2k_a d_0 \frac{\mathbf{P}-\mathbf{P}_{goal}}{d}, & d > d_0 \end{cases} \tag{2}$$

$$U_{rep} = \begin{cases} \frac{1}{2}k_r\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2, & \rho \le \rho_0 \\ 0\,, & \rho > \rho_0 \end{cases} \tag{3}$$

$$\mathbf{F}_{rep} = -\nabla U_{rep} = \begin{cases} k_r\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)\frac{\mathbf{P}-\mathbf{P}_{co}}{\rho^3}, & \rho \le \rho_0 \\ 0\,, & \rho > \rho_0 \end{cases} \tag{4}$$

d_0 is the radius of the quadratic range, k_a is the proportional gain of the function, and $d = \left\|\mathbf{P} - \mathbf{P}_{goal}\right\|$. In the quadratic area, the attractive force is proportional to distance between the goal and the robot. If d_0 is large, a robot smoothly and slowly stops at the goal. $\mathbf{P}$ and $\mathbf{P}_{goal}$ are the position vectors of the robot and the goal. ρ_0 is a potential field's distance limit of repulsive potential influence, and ρ is the shortest distance between the robot and its closest obstacle, where $\rho = \|\mathbf{P} - \mathbf{P}_{co}\|$. $\mathbf{P}_{co}$ is the position vector of the closest obstacle to the robot. The entire artificial force $\mathbf{F}_{total}$ is $\mathbf{F}_{att} + \mathbf{F}_{rep}$. The robot is controlled by this force, which is converted to speed and angular velocity according to the force-to-velocity conversion method [32].

However, before arriving at the goal, the robot is trapped in local minima and stops driving when the attractive force and the repulsive force are the same or very similar. In [32], the virtual hill allows the robot to escape from the LM (local minimum) by generating an extra force instead of an attractive force when in the LM area. The artificial force, $\mathbf{F}_{total}$, applied to the robot from the moment of being trapped in the LM to the moment of escape, is $\mathbf{F}_{rep} + \mathbf{F}_{ext}$. After the robot is out of the LM, $\mathbf{F}_{total}$ = $\mathbf{F}_{att}$+$\mathbf{F}_{rep}$ is applied to move to the goal. Extra force can be applied even in complex environments and does not create a new local minimum. Extra potential is applied from the moment the robot is trapped in the LM until the escape is completed and is defined as follows.

$$U_{ext} = -k_{e1}\Psi + k_{e2}\rho^2 \tag{5}$$

k_{e1} and k_{e2} are proportional constants. ρ is the distance between the robot and the closest obstacle. Ψ is the path integral, which is the line integral for γ and expressed as

$$\Psi = \int_{\gamma} d\Psi, \qquad \gamma(t) \ : \ \{\mathbf{Q}(t^*) : t_0 \leq t^* \leq t \ \text{ and } \ t_0 \leq t \leq t_k \ \} \tag{6}$$

$$\mathbf{Q}(t_i) = \mathbf{P}_{co}(t_i) \qquad \text{where } i = 0, 1, \cdots, K \tag{7}$$

γ is defined as the path of the closest obstacles, indicating the trajectory of the closest obstacle while the robot is moving within the LM. The concept of γ is shown in Figure 2a [34]. t_0 is the time when the robot is trapped in the LM, and t_k is the time when it escapes from the LM. $\mathbf{Q}$ is the position vector of the closest obstacle on the continuous trajectory. $\mathbf{P}_{co}$ is the position vector of the closest obstacle measured discretely by the range sensor. t_i represents a discrete sampling time of the range sensor. Assuming that $\mathbf{Q}$ is continuously interpolated and differentiable, Ψ can be written as follows.

$$\Psi(t) = \int_{\gamma(t)} d\Psi = \int_{t_0}^{t} \frac{d\Psi}{dt} dt = \int_{t_0}^{t} \|\dot{\mathbf{Q}}\| dt \qquad \text{where} \quad \dot{\mathbf{Q}} = d\mathbf{Q}/dt \tag{8}$$

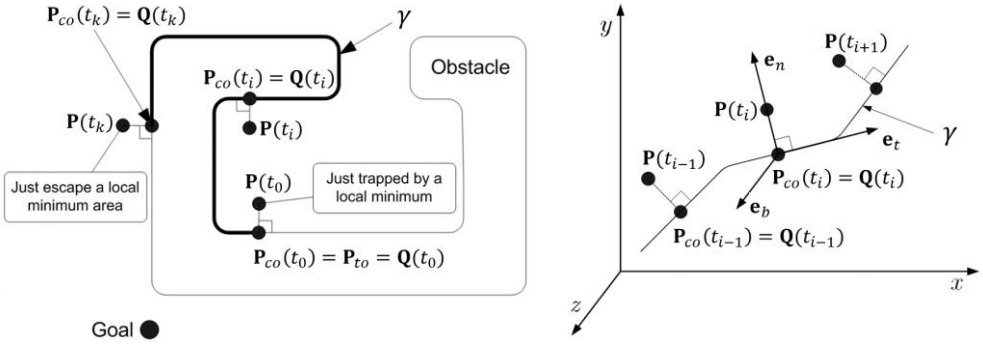


Figure 2. Within the local minimum area: (**a**) concept of γ; (**b**) concept of $\mathbf{e}_t$, $\mathbf{e}_n$, and $\mathbf{e}_b$ with respect to γ.

$\Psi(t)$ denotes a distance from $\mathbf{Q}(t_0)$ to $\mathbf{Q}(t)$ along γ. The extra force is the negative gradient of extra potential, and it is shown in Equation (9).

$$\mathbf{F}_{ext} = -\nabla U_{ext} = k_{e1} \nabla \Psi - k_{e2} \nabla \left(\rho^2 \right) \tag{9}$$

This can also be expressed as Equation (10) [34].

$$\mathbf{F}_{ext} = k_{e1} \mathbf{e}_t - 2 k_{e2} \rho \mathbf{e}_n \tag{10}$$

In Figure 2b, $\mathbf{e}_t$ is defined as the unit tangent vector of γ, and the derivative of $\mathbf{Q}$ has the direction of $\mathbf{e}_t$.

$$\mathbf{e}_t = \frac{\dot{\mathbf{Q}}}{\|\dot{\mathbf{Q}}\|} = \mathbf{e}_n \times \mathbf{e}_b \tag{11}$$

$\mathbf{e}_t(t_0)$ can be obtained by the cross product of the unit normal vector $\mathbf{e}_n$ and unit binormal vector $\mathbf{e}_b$.

$$\mathbf{e}_n = \frac{\mathbf{P} - \mathbf{P}_{co}}{\|\mathbf{P} - \mathbf{P}_{co}\|} = \frac{\mathbf{P} - \mathbf{P}_{co}}{\rho} \tag{12}$$

$$\mathbf{e}_b = \mathbf{e}_t \times \mathbf{e}_n = \mathbf{e}_t \times \frac{\mathbf{P} - \mathbf{P}_{co}}{\rho} \tag{13}$$

$\mathbf{P}_{co}$ is the closest point from the robot to γ, and $\overline{\mathbf{PP}_{co}}$ is always vertical to γ. The initial direction of γ is determined by $\mathbf{e}_t(t_0)$. In order for the initial direction of γ to be directed toward goal, $\mathbf{e}_b$ is defined as follows.

$$\mathbf{e}_b = \frac{\left(\mathbf{P}_{goal} - \mathbf{P}_{to}\right) \times \left(\mathbf{P}(t_0) - \mathbf{P}_{to}\right)}{\left\| \left(\mathbf{P}_{goal} - \mathbf{P}_{to}\right) \times \left(\mathbf{P}(t_0) - \mathbf{P}_{to}\right) \right\|} \qquad where \quad \mathbf{P}_{to} = \mathbf{P}_{co}(t_0) \tag{14}$$

$\mathbf{P}(t_0)$ is the position vector of the robot when it is trapped in the LM. As shown in Figure 3, $\mathbf{e}_b$ is set to $\mathbf{K}$ or $-\mathbf{K}$ at the time of t_0 according to the relative positions of the robot, the closest obstacle, and the goal. At the moment the robot is trapped in the LM, $\mathbf{e}_t(t_0)$, which is the direction for the robot to move, is determined according to the sign of $\mathbf{e}_b$.

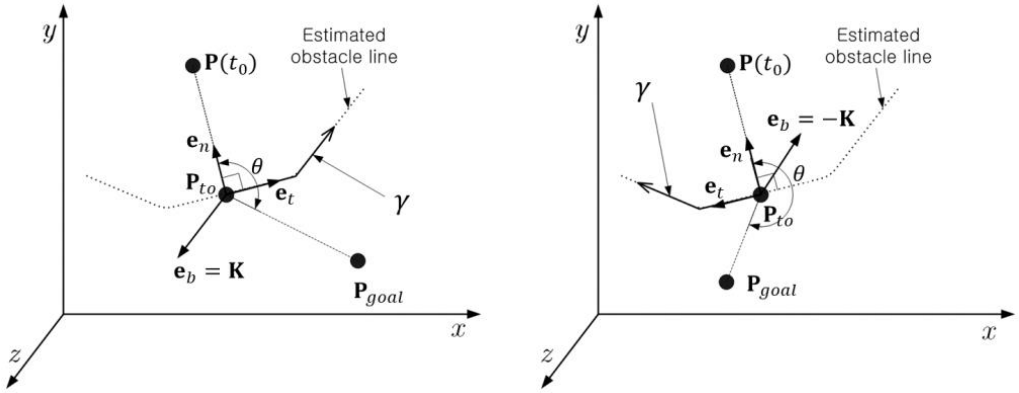


Figure 3. Direction of γ: (**a**) when $\mathbf{e}_b = \mathbf{K}$; (**b**) when $\mathbf{e}_b = -\mathbf{K}$.

3.2. Problems with Previous e_b

$$\boldsymbol{F}_{total} = \boldsymbol{F}_{rep} + \boldsymbol{F}_{ext} = \begin{cases} k_{e1}\boldsymbol{e}_t + \left(k_r\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2} - 2k_{e2}\rho\right)\boldsymbol{e}_n, & \rho \le \rho_0 \\ k_{e1}\boldsymbol{e}_t - 2k_{e2}\rho\boldsymbol{e}_n, & \rho > \rho_0 \end{cases} \tag{15}$$

From the moment the robot is trapped in the LM to the moment of its escape, the artificial force applied to the robot is $\mathbf{F}_{rep} + \mathbf{F}_{ext}$. In Equation (10), the first term $k_{e1}\mathbf{e}_t$ of $\mathbf{F}_{ext}$ acts for a robot to move along the obstacle. The second term $-2k_{e2}\rho\mathbf{e}_n$ acts in the opposite direction to the repulsive force so that the robot does not move too far away from the obstacle. The extra force operates to move the robot along the outline of the obstacle until the robot escapes the LM without being too far away from the obstacle. In the previous virtual hill algorithm, $\mathbf{e}_b$ was determined by the location relationship, and the form of the surrounding obstacle was not considered. Therefore, depending on the form of the map, the robot sometimes set an inefficient path. Figure 4 shows the moment when the robot is trapped in the LM, and $\mathbf{P}_{to}$ is the position of the closest obstacle at the moment when it is trapped in the LM. According to Equation (14) used in the previous algorithm, $\mathbf{e}_b$ has the $-\mathbf{K}$ direction. In the LM section, since $\mathbf{e}_t$ indicating the driving direction is $\mathbf{e}_n \times \mathbf{e}_b$, the robot moves in the negative direction (clockwise). Eventually, it goes around the blocked path, as shown in Figure 5 (Point E is the moment it is determined that the robot has escaped the LM). Depending on which direction is selected at the moment when the robot is trapped in the LM, it may take quite a long path, or a dead path may be selected. Determining $\mathbf{e}_b$ by considering only the relative positions, as in Equation (14), may complicate the movement path, as in Figure 5. Therefore, a more effective path planning algorithm that considers the form of obstacles is required.

3.3. Open Path Indicator, New e_b

new e_b is proposed, in which the obstacle form is considered. As soon as the robot is trapped in the LM, it is necessary to select in which of the two directions to move. This is divided into the $(+)$ direction and $(-)$ direction based on $\mathbf{P}(t_0)$. For each direction, the rate of change in the obstacle contour is calculated. The direction with the larger rate of change is considered the more open path, that is, the driving direction. This direction setting helps

the robot avoid a closed path and reach the destination with a shorter moving distance. More efficient path planning becomes possible when the driving direction is determined in consideration of the shape of the obstacle.

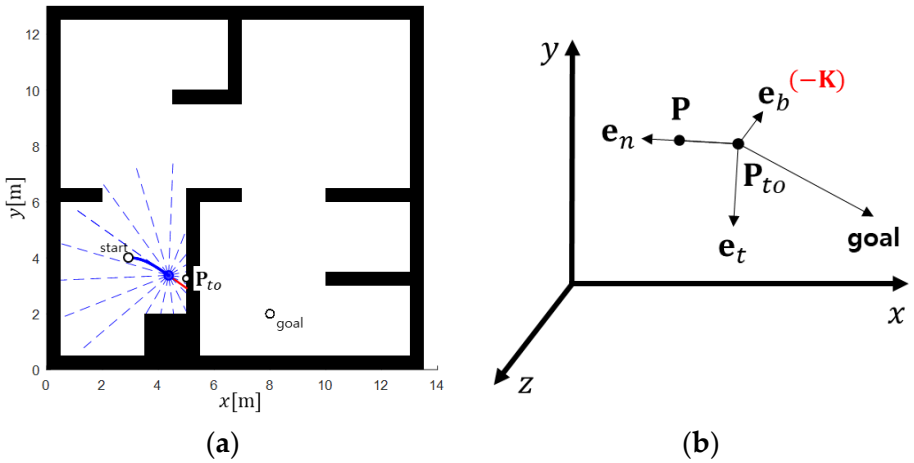


Figure 4. Direction determined by *previous* $\mathbf{e}_b$: (**a**) the moment the robot is trapped in the local minimum; (**b**) $\mathbf{e}_t$ determined by *previous* $\mathbf{e}_b$.

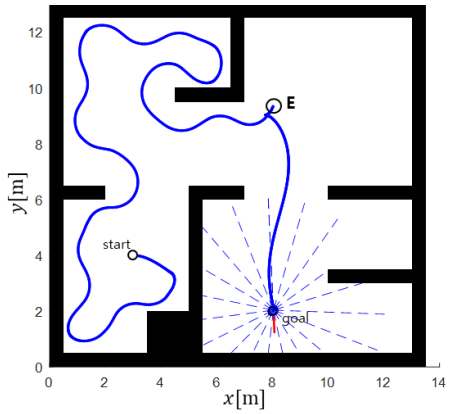


Figure 5. Driving path determined by *previous* $\mathbf{e}_b$. The path length is 44.26 [m].

3.3.1. Two-Directional Obstacle Measurement and *New* e_b

In the simulation environment, the range sensor is configured in a radial form with the center of the robot as the origin. The degree of openness in both directions is estimated using the rate of change in the detection distance according to the measurement angle of each sensor data. In order to calculate the rate of change in the obstacle detection distance, it is expressed as a polar coordinate system fixed to the center of the robot. Figure 6 shows the polar coordinate system with the origin at the center of the robot and the angle of each sensor data set in the simulation. The angle and detection distance for each sensor data are expressed as θ_i and r_i. At the moment when the robot is trapped in the LM, the rate of change in the outline of the obstacle is calculated for both the left and right directions based on the n-th sensor data that measured $\mathbf{P}_{to}$. The one with the larger value is considered the open path and becomes the direction for the robot to move. When the contour of the two-directional obstacle is expressed as continuous functions χ_p and χ_m, the moment of the LM is shown in Figure 7. The differential lengths $d\chi_p$ and $d\chi_m$ in the polar coordinate system can be written as follows.

$$d\vec{\chi_p} = dr\ \mathbf{a}_r + r\ d\theta\ \mathbf{a}_\theta \qquad d\vec{\chi_m} = dr\ \mathbf{a}_r + r\ d\theta\ \mathbf{a}_\theta \tag{16}$$

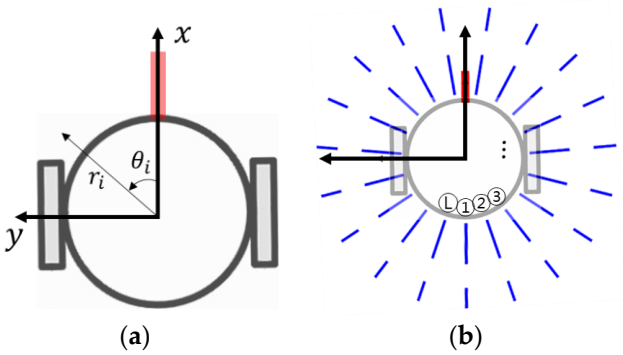


Figure 6. Local polar coordinate system with an origin at the center of the robot: (**a**) sensor measurement angle θ_i and detection distance r_i; (**b**) sensor data numbers $(1 \cdots \mathrm{L})$ and measurement angles.

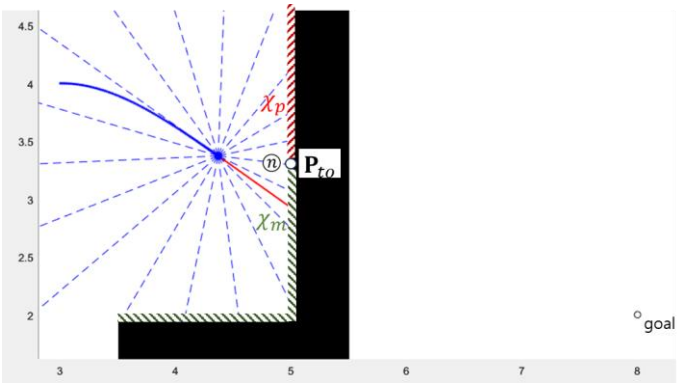


Figure 7. The $(+)$ directional obstacle χ_p and $(-)$ directional obstacle χ_m based on the n-th sensor data in which $\mathbf{P}_{to}$ is measured (enlargement of Figure 4a).

The rate of change in the obstacle contour for θ, $d\chi_p/d\theta$ and $d\chi_m/d\theta$, is as follows.

$$\left|\frac{d\chi_p}{d\theta}\right| = \sqrt{\left(\frac{dr}{d\theta}\right)^2 + r^2} \qquad \left|\frac{d\chi_m}{d\theta}\right| = \sqrt{\left(\frac{dr}{d\theta}\right)^2 + r^2} \tag{17}$$

The obstacle contour is discretized as distance values received from the L sensor data, $\{\mathbf{R}_1(r_1,\theta_1),\ \mathbf{R}_2(r_2,\theta_2),\ \ldots,\ \mathbf{R}_i(r_i,\theta_i),\ \ldots \mathbf{R}_L(r_L,\theta_L)\}$. If $dr/d\theta$ is replaced with $\Delta r/\Delta\theta$, the sum of the power of the rate of change in each of the two directions based on the n-th sensor data may be written as follows.

$$P_{sum} = \sum_{i=n}^{n+L/2} \left(\frac{r_{i+1} - r_i}{\Delta\theta}\right)^2 + r_i^2 \qquad M_{sum} = \sum_{i=n-L/2}^{n} \left(\frac{r_{i+1} - r_i}{\Delta\theta}\right)^2 + r_i^2 \tag{18}$$

The angle and obstacle measurement distance of the i-th and $(i+1)$-th sensor data are expressed in the polar coordinate system, as in Figure 8. The angle change $\Delta\theta$ with the neighboring sensor data is determined according to the sensor resolution L and is calculated as $2\pi/L$. P_{sum} is the sum of squares of the rate of change in the obstacle contour in the $(+)$ direction based on the n-th sensor data, and M_{sum} is the sum of squares of the rate of change in the $(-)$ direction. The direction with the further distance to the obstacle and the greater rate of change is regarded as an open path. That is, $\mathbf{e}_b$ is set to $+\mathbf{K}$ when $P_{sum} > M_{sum}$ and to $-\mathbf{K}$ when $P_{sum} < M_{sum}$. In the map of Figure 4, *previous* $\mathbf{e}_b$ is $-\mathbf{K}$, and the robot rotates in the direction of $-\theta$. *new* $\mathbf{e}_b$, determined by the new algorithm, is $+\mathbf{K}$, and the robot rotates in the direction of $+\theta$, as shown in Figure 9. More efficient path planning is possible. If the degree of opening in both directions is similar by satisfying the following conditions, *previous* $\mathbf{e}_b$ is maintained. *threshold* is determined experimentally.

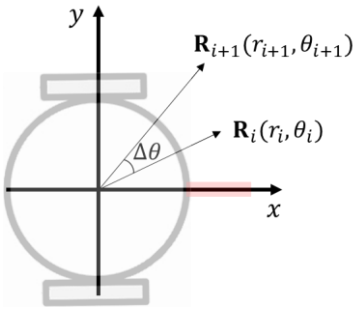


Figure 8. The i-th and the $(i+1)$-th vectors of the L sensor data.

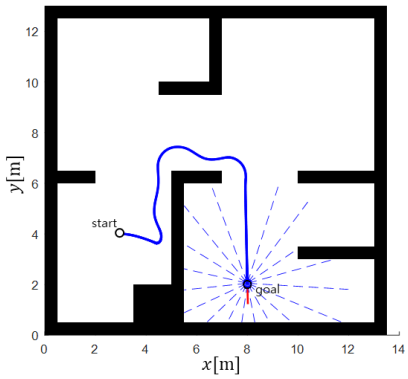


Figure 9. Driving path determined by *new* $\mathbf{e}_b$ (in comparison to Figure 5). The path length is 13.82 [m].

$$if \quad P_{sum} > threshold \;\&\; M_{sum} > threshold \quad then \quad new\ \mathbf{e}_b = previous\ \mathbf{e}_b \tag{19}$$

3.3.2. Dead-End Algorithm

When the robot encounters a dead end while traveling to the destination, it is efficient to avoid it unless the destination is inside the dead end. When a robot encounters a dead end, if it is not on the way to its destination, it can avoid the dead end using the dead-end algorithm. Figure 10a shows the path planning results when $\mathbf{e}_t$ is determined by *previous* $\mathbf{e}_b$. This is the path where the robot trapped in the LM at point $\mathbf{P}(t_0)$ moves to the destination along the obstacle ABCD section using the virtual hill technique. The ABCD section is a dead end and unnecessarily increases the robot's travel distance and time. In this case, the robot can also come back through the dead-end algorithm just as people come back after confirming that it is a dead-end path.

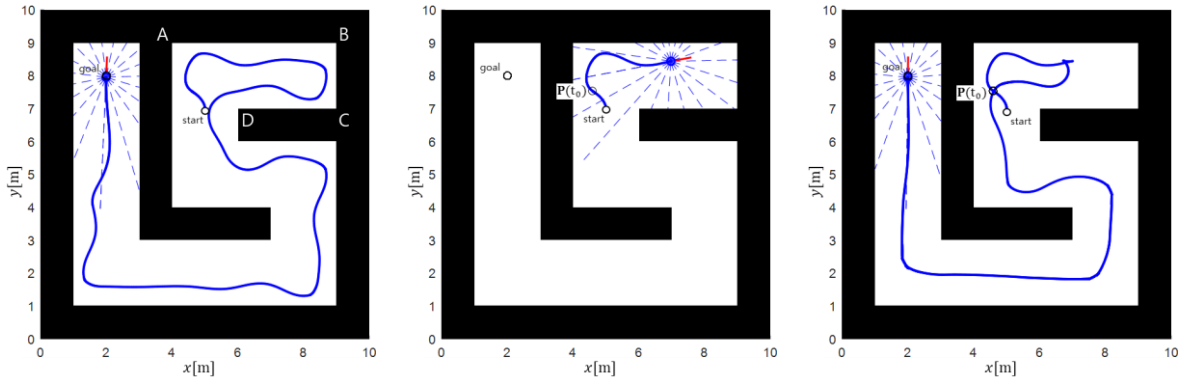


Figure 10. Dead-end algorithm: (**a**) driving path determined by a virtual hill with *previous* $\mathbf{e}_b$. The path length is 32.87 [m]. (**b**) The moment the robot notices a dead end. (**c**) Driving path determined by the dead-end algorithm. The path length is 28.99 [m].

The robot returns to the point $\mathbf{P}(t_0)$ and then moves in the opposite direction. The robot travels along obstacles closer to the destination. If the dead-end algorithm is applied when encountering a dead end, the robot can avoid the dead end without completely entering it, as shown in Figure 10c. This algorithm is applied between the beginning and the end of the LM. The flowchart of the dead-end algorithm is shown in Figure 11. The criteria for determining that the robot is at a dead end are as follows.

$$if r(\theta) < maxRange, -\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2} then\ the\ path\ is\ a\ dead\ end. \tag{20}$$

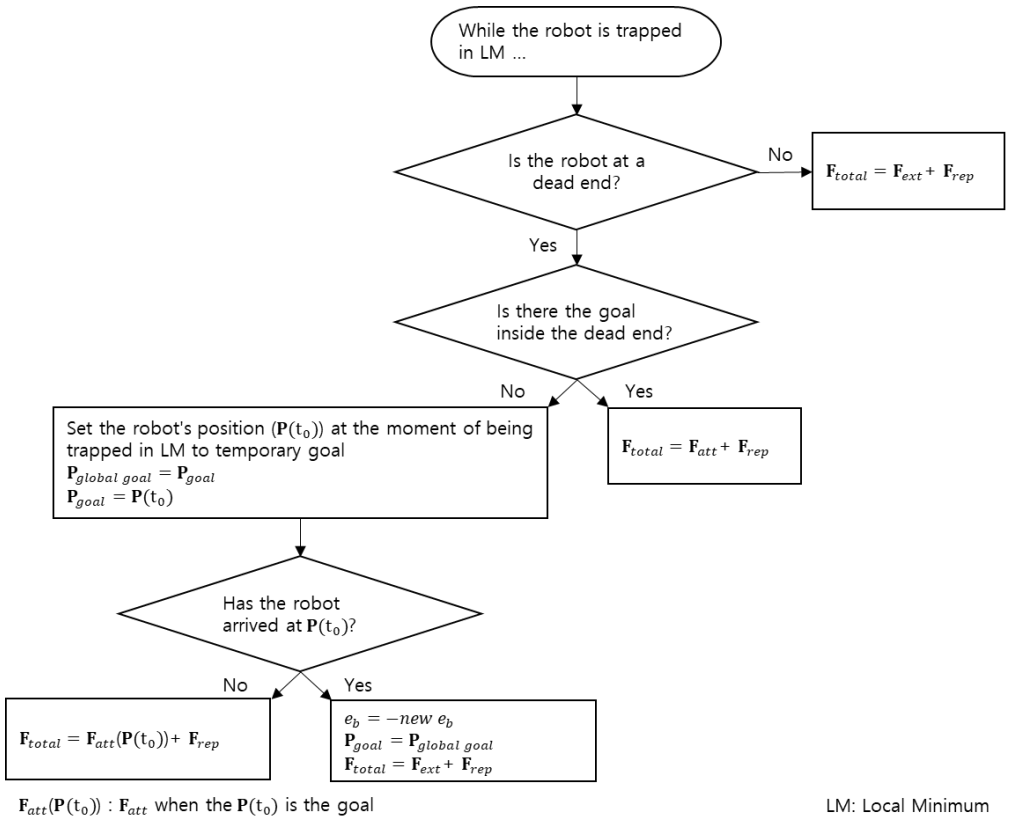


Figure 11. Flowchart of the dead-end algorithm.

$r(\theta)$ is the distance of the obstacle detected by the sensor, and $maxRange$ is the sensor's maximum detection distance. If the obstacle detection distance of all sensor data in the $[-\pi/2, \pi/2]$ section corresponding to the front portion of the robot is shorter than $maxRange$, it is determined that the robot has entered a dead end. If the robot is at a dead end, whether to enter or avoid further depends on the location of the destination. The following two conditions are examined to determine whether the destination is inside or outside the dead end.

$$\begin{cases} \left|\mathbf{P}^{\mathbf{L}}_{goal}\right| < r_k \\ -\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}, \quad \mathbf{P}^{\mathbf{L}}_{goal} = \left|\mathbf{P}^{\mathbf{L}}_{goal}\right| \angle\alpha \end{cases} \tag{21}$$

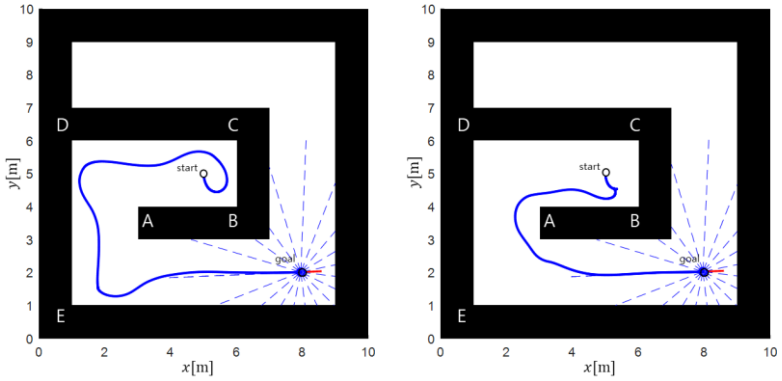
$\mathbf{P}^{\mathbf{L}}_{goal}$ represents the location vector of the destination for the local coordinate system with an origin at the center of the robot, as shown in Figure 6a. r_k is the measured distance of the k-th sensor data, which is at the angle closest to the vector $\overrightarrow{\mathbf{P}_{goal}} - \mathbf{P}$ among all sensor data. $\left|\mathbf{P}^{\mathbf{L}}_{goal}\right| < r_k$ means that there is no obstacle between the robot and the destination. α is the angle of $\mathbf{P}^{\mathbf{L}}_{goal}$ in the local coordinate system. $-\pi/2 \leq \alpha \leq \pi/2$ means that the goal is at the front portion of the robot. If both conditions are satisfied, it is determined that the

destination is inside the dead end. If the destination is inside the dead end, the robot is controlled to the destination with $\mathbf{F}_{total} = \mathbf{F}_{att} + \mathbf{F}_{rep}$. This $\mathbf{F}_{att}$ is an attractive force heading to the initially set goal (destination). If the destination is outside the dead-end road, the robot returns to $\mathbf{P}(t_0)$ and moves in the opposite direction. In order for the robot to return to $\mathbf{P}(t_0)$, $\mathbf{P}_{goal}$ is temporarily changed to $\mathbf{P}(t_0)$, and $\mathbf{F}_{total} = \mathbf{F}_{att}(\mathbf{P}(t_0)) + \mathbf{F}_{rep}$ is applied to the control. $\mathbf{F}_{att}(\mathbf{P}(t_0))$ described an attractive force generated when $\mathbf{P}(t_0)$ is set as $\mathbf{P}_{goal}$. When the robot arrives at the position $\mathbf{P}(t_0)$, $\mathbf{P}_{goal}$ is reset to the initial goal, and it moves in the opposite direction to $e_t(t_0)$. $\mathbf{F}_{total} = \mathbf{F}_{ext} + \mathbf{F}_{rep}$ is applied until the robot escapes from the LM.

4. Simulations

Some conditions for simulation are referred to in [32]. The virtual hill algorithm has several parameters for controlling the robot, such as the constant coefficients of the extra force, the application distance of the reactive force, and the like. Depending on the setting, the degree to which the robot approaches or moves away from obstacles, the straightness of the driving, and similar parameters may vary. Some maps provided by MATLAB are applied in sizes of 10 [m] × 10 [m]. The total artificial force is converted into linear velocity and angular velocity by the force-to-velocity conversion method [32]. The path length is indicated in each figure description.

Figures 12 and 13 show the results of the enhanced virtual hill algorithm. In the previous algorithm, when the robot is trapped in an LM (local minimum), the direction of progress is determined by the relative positions of $\mathbf{P}(t_0)$, $\mathbf{P}_{to}$, and the goal. According to this algorithm, as shown in Figure 12a, the robot moves along the obstacle BCDE. On the other hand, when the enhanced virtual hill algorithm including *new* $\mathbf{e}_b$ is applied, the robot moves along the obstacle BA, as shown in Figure 12b. When the robot reaches the goal by following the obstacle BA, the efficiency of the driving path may be improved. In the case of Figure 13, according to the direction determination by the previous virtual hill algorithm, that is, *previous* $\mathbf{e}_b$, the robot goes around in the $(+)$ direction, as shown in Figure 13a. The robot moves along the obstacle ABCDEFGH until it escapes the LM. Since the path determined by *new* $\mathbf{e}_b$ goes to the goal along obstacle OA, unnecessary driving may be avoided. In Figure 13, the driving distance varies considerably depending on the direction determination at the LM.

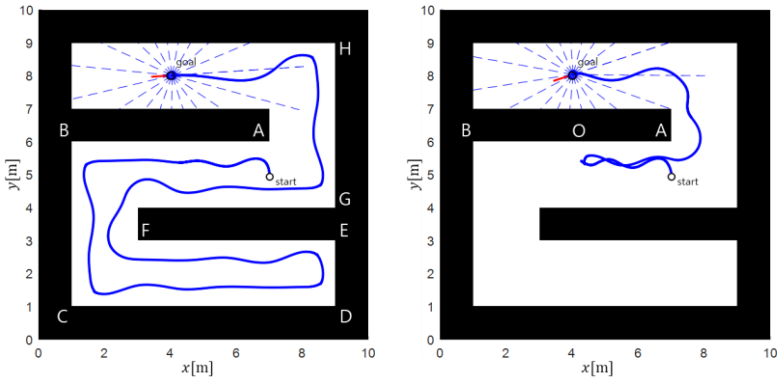


(**a**) (**b**)

Figure 12. Driving paths: (**a**) Determined by *previous* $\mathbf{e}_b$. The path length is 16.51 [m]. (**b**) Determined by *new* $\mathbf{e}_b$. The path length is 11 [m].

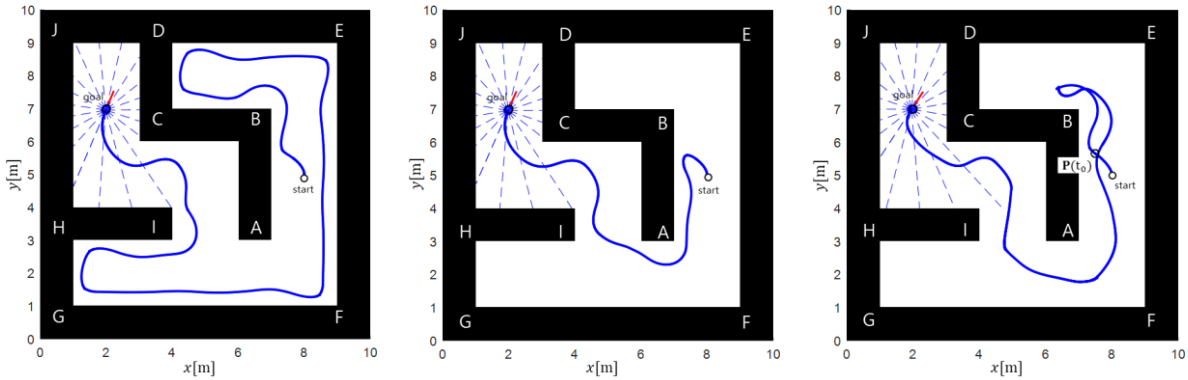
Figures 14–19 show the simulations of the driving path by *new* $\mathbf{e}_b$ and the dead-end algorithm. If the robot recognizes that it is at a dead end while driving in the LM section, it returns to the robot's position $\mathbf{P}(t_0)$. At $\mathbf{P}(t_0)$, the robot moves in the opposite direction and follows a closer obstacle to its goal. In the case of Figure 14a, at the point of t_0, it travels in the $(-)$ direction and travels along the obstacle ABCDEFGHI for a long distance. Figure 14b travels along the $(+)$ direction according to *new* $\mathbf{e}_b$ and travels the shortest distance along

the obstacle BA. Figure 14c shows the result of applying the dead-end algorithm during driving, as shown in Figure 14a. After recognizing the obstacle BCDE as a blocked road, the robot returns to $\mathbf{P}(t_0)$ and moves in the direction of obstacle BA. In Figure 17c, as the robot rotates in the $(+)$ direction, the recognition of the blocked road, arrival at $\mathbf{P}(t_0)$, and setting of the opposite direction occur almost simultaneously, drawing a path similar to that in Figure 17b without any special driving.



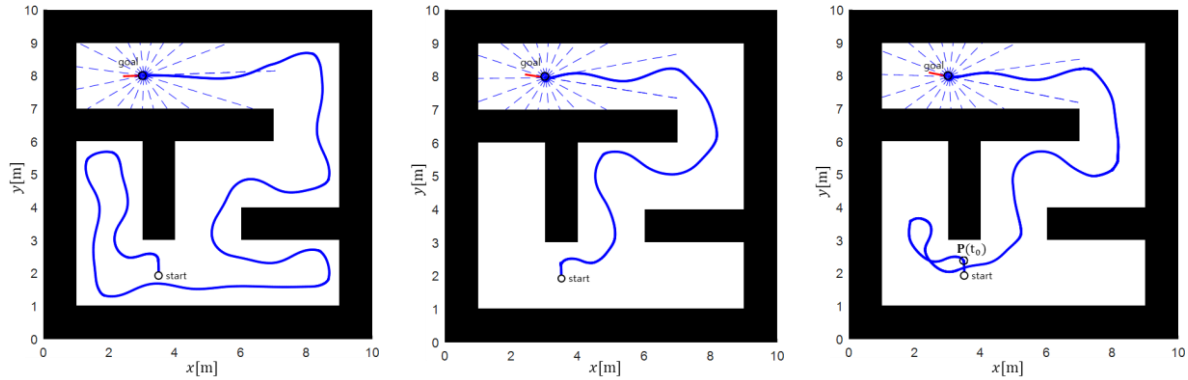
(**a**) (**b**)

Figure 13. Driving paths: (**a**) Determined by *previous* $\mathbf{e}_b$. The path length is 41.08 [m]. (**b**) Determined by *new* $\mathbf{e}_b$. The path length is 12.91 [m].



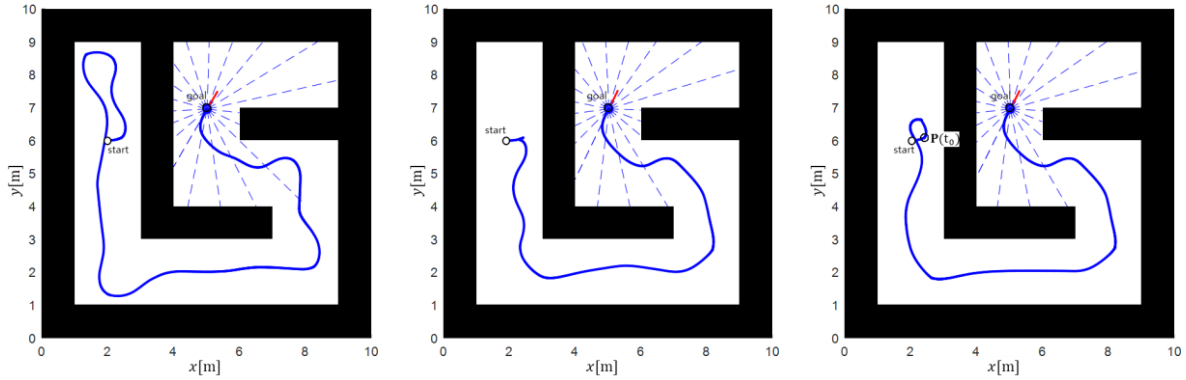
(**a**) (**b**) (**c**)

Figure 14. Driving paths: (**a**) Determined by *previous* $\mathbf{e}_b$. The path length is 36.66 [m]. (**b**) Determined by *new* $\mathbf{e}_b$. The path length is 12.87 [m]. (**c**) Determined by *previous* $\mathbf{e}_b$ and the dead-end algorithm. The path length is 21.7 [m].



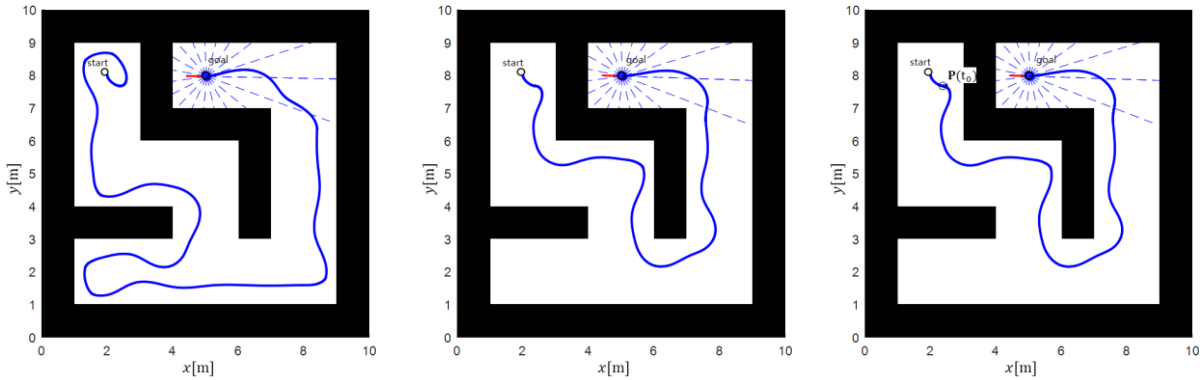
(**a**) (**b**) (**c**)

Figure 15. Driving paths: (**a**) Determined by *previous* $\mathbf{e}_b$. The path length is 35.95 [m]. (**b**) Determined by *new* $\mathbf{e}_b$. The path length is 15.98 [m]. (**c**) Determined by *previous* $\mathbf{e}_b$ and the dead-end algorithm. The path length is 21.72 [m].



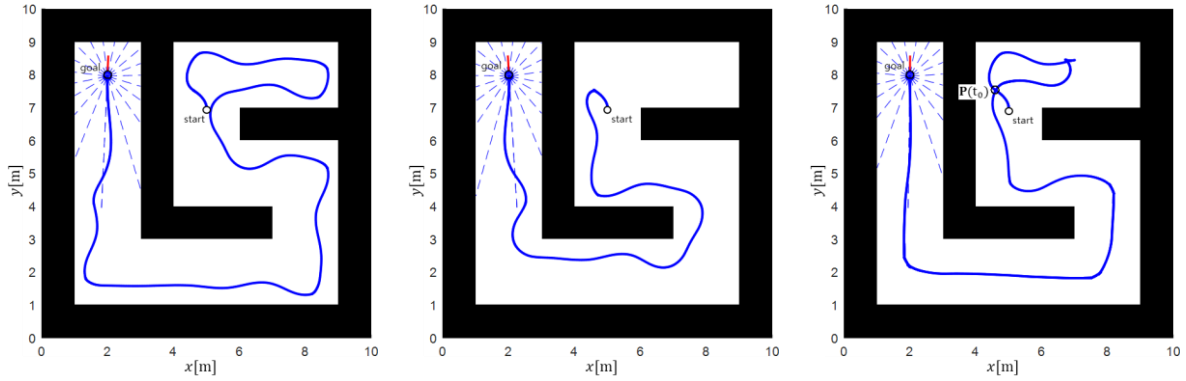
(**a**) (**b**) (**c**)

Figure 16. Driving paths: (**a**) Determined by *previous* $\mathbf{e}_b$. The path length is 25.6 [m]. (**b**) Determined by *new* $\mathbf{e}_b$. The path length is 17.33 [m]. (**c**) Determined by *previous* $\mathbf{e}_b$ and the dead-end algorithm. The path length is 19.02 [m].



(**a**) (**b**) (**c**)

Figure 17. Driving paths: (**a**) Determined by *previous* $\mathbf{e}_b$. The path length is 32.34 [m]. (**b**) Determined by *new* $\mathbf{e}_b$. The path length is 18.96 [m]. (**c**) Determined by *previous* $\mathbf{e}_b$ and the dead-end algorithm. The path length is 19.14 [m].



(**a**) (**b**) (**c**)

Figure 18. Driving paths: (**a**) Determined by *previous* $\mathbf{e}_b$. The path length is 32.87 [m]. (**b**) Determined by *new* $\mathbf{e}_b$. The path length is 20.17 [m]. (**c**) Determined by *previous* $\mathbf{e}_b$ and the dead-end algorithm. The path length is 28.99 [m].

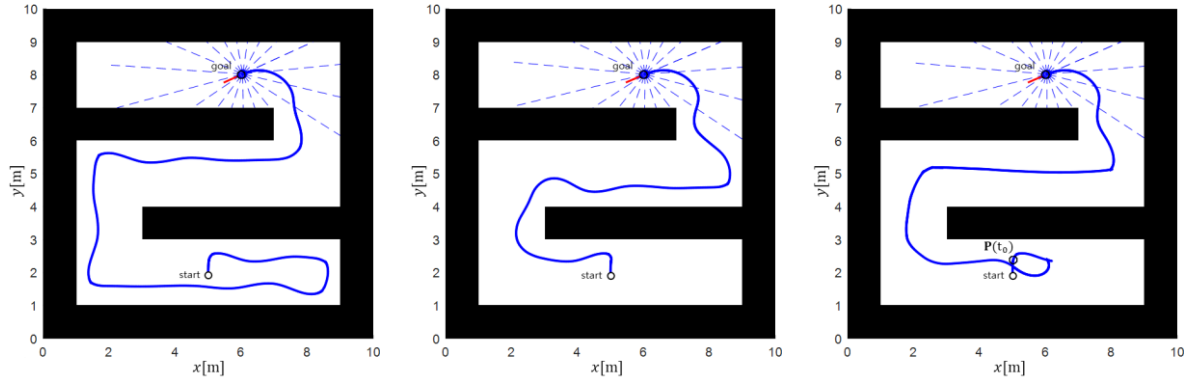


Figure 19. Driving paths: (**a**) Determined by *previous* $\mathbf{e}_b$. The path length is 26.12 [m]. (**b**) Determined by *new* $\mathbf{e}_b$. The path length is 16.61 [m]. (**c**) Determined by *previous* $\mathbf{e}_b$ and the dead-end algorithm. The path length is 19.48 [m].

In Figure 20, (c) shows the moment of the LM, (a) shows the driving path determined by *previous* $\mathbf{e}_b$, and (b) shows the driving path determined by the *new* $\mathbf{e}_b$. Using *new* $\mathbf{e}_b$, $\mathbf{e}_t$ is determined so that the robot may travel on a more open path. However, depending on the map, the direction that was the more open path may be blocked, as shown in Figure 20. The more open direction is the (+) direction when comparing (+) and (−) directions from the robot's position shown in Figure 20c, but the robot eventually encounters a dead end. Nevertheless, there is no significant difference in path length between Figure 20a,b. In Figure 20a, the robot avoids the dead end using *previous* $\mathbf{e}_b$ but follows obstacles far from the destination. In Figure 20b, the robot enters the dead end, but the path length is reduced by following obstacles closer to the destination due to *new* $\mathbf{e}_b$.

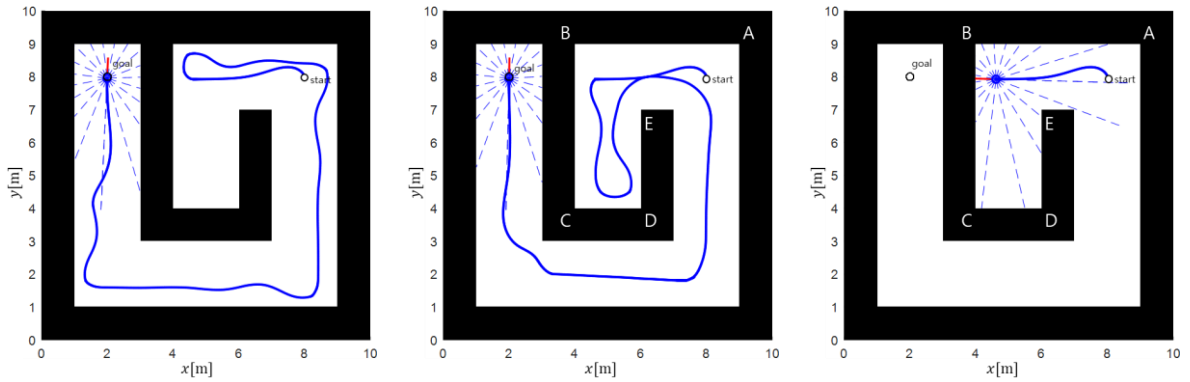


Figure 20. For a map that contains a dead end: (**a**) The driving path determined by *previous* $\mathbf{e}_b$. The path length is 29.41 [m]. (**b**) The driving path determined by *new* $\mathbf{e}_b$. The path length is 30.91 [m]. (**c**) The moment of the local minimum.

Table 1 shows how much the robot's mileage is reduced by the proposed algorithm. It represents the distance and ratio reduced by *new* $\mathbf{e}_b$ and the dead-end algorithm based on the path determined by *previous* $\mathbf{e}_b$ for all simulations shown in the paper. The driving distance is reduced by up to 68 [%] using *new* $\mathbf{e}_b$ and by up to 44 [%] using the dead-end algorithm. It is confirmed that the virtual hill algorithm is improved by the proposed method.

Table 1. Reduction in mileage according to algorithms.

	Mileage Using *Previous* $\mathbf{e}_b$ [m]	Mileage Using *New* $\mathbf{e}_b$ [m]	Mileage Using Dead-End Algorithm [m]	Reduction Proportion Using *New* $\mathbf{e}_b$ [%]	Reduction Proportion Using Dead-End Algorithm [%]
Figure 9	44.26	13.82	-	68.78	-
Figure 12	16.51	11	-	33.37	-
Figure 13	41.08	12.9	-	68.60	-
Figure 14	36.66	12.87	14.96	64.89	40.81
Figure 15	35.95	15.98	14.23	55.55	39.58
Figure 16	25.6	17.33	6.58	32.30	25.70
Figure 17	33.66	19.02	14.83	43.49	44.06
Figure 18	32.87	20.17	3.88	38.64	11.80
Figure 19	26.12	16.61	6.64	36.41	25.42
Figure 20	29.41	30.91	-	−5.10	-
Average [%]	-	-	-	43.69	31.23

5. Discussion

In order to compare with the previous virtual hill algorithm, simulations are performed on the same environment for both *previous* $\mathbf{e}_b$ and *new* $\mathbf{e}_b$. Also, the dead-end algorithm is applied when entering a blocked road, and how the path changes is checked. The circle mark is displayed at the position of $\mathbf{P}(t_0)$ for visual confirmation at the moment the blocked path is recognized. All simulations are saved as pictures, data files, and videos.

In Figures 12–19, each (a) is the result when *previous* $\mathbf{e}_b$ is applied, and each (b) is the result when *new* $\mathbf{e}_b$ is applied. They show that the moving distance can be significantly reduced when determining the driving direction by comparing the degree of openness of the two-way route regardless of the relative position of the robot and the goal. (c) Shows the result of applying the dead-end algorithm when a route including a blocked road is selected. It is possible to reduce the moving distance by more than about 4 m by coming back without entering deep into the dead end. In the service environment, if the mobile robot travels at a low speed of less than 0.2 m/s, it can save more than about 20 s. The saving distance and time can vary depending on the measurable distance of the sensor and control techniques. Very occasionally, the path along the *new* $\mathbf{e}_b$ is more inefficient, as shown in Figure 20. The maximum measurement distance of the sensor is 4 m. Since the distance from the robot position in Figure 20c to the obstacle CD is about 4 m, it is difficult for the robot to recognize the dead end road in the $(+)$ direction. Unless the robot has map information in advance or creates a map while moving, it may encounter dead ends when moving through an unknown space. The dead-end algorithm prevents the robot from having to drive unnecessarily deep into dead ends.

6. Conclusions

The virtual hill technique, which moves along a nearby obstacle when a robot is trapped in an LM, is easy to apply to an unknown variable environment. The virtual hill technique does not require information about space and obstacles, does not create a new the LM even in a complicated environment, and has the advantage of being easy to apply. However, there is a problem in that the mileage can vary greatly depending on which directional obstacle the robot moves along. In this study, the performance of the virtual hill technique is improved by compensating for this shortcoming using *new* $\mathbf{e}_b$ and the dead-end algorithm. The enhanced virtual hill algorithm makes it possible for a robot to reach its goal on a shorter path. Also, when a robot encounters blocked roads, it can come back without entering through the dead-end algorithm.

In future research, when the robot is trapped in the LM and extra potential is generated, it can be programmed to check in advance whether there is a dead end nearby and exclude that direction. In cases like Figure 20a, it is necessary to recognize in advance that if the robot moves in the $(+)$ direction, it will lead to a dead end. In order to recognize dead

ends from a long distance, the sensor's obstacle measurement distance must be longer, and more sensor data must be processed for accuracy. It is unnecessary to process this much data in each control cycle. A method is needed to obtain detailed information about distant obstacles only when needed. To verify the proposed algorithm, additional simulations and experiments in diverse and dynamic environments are required. It is also necessary to establish indicators to quantitatively evaluate the stability and flexibility of robot driving. Through these, it is expected that the performance and limitations of the proposed algorithm will be analyzed, and shortcomings can be resolved in various conditions where static/dynamic obstacles exist in combination.

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

Funding: This work was supported by the Development of High-Level Cognitive Prediction Sensor Technology for Autonomous Driving Mobility (RS-2022-00144500, Development of 3D Ultrasonic Sensor Technology for Vehicles based on Meta Structure) funded by the Ministry of Trade Industry & Energy (MOTIE, Republic of Korea).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

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

References

1. Rimon, E.; Koditschek, D.E. Exact robot navigation using cost functions: The case of distinct spherical boundaries in E^n. In Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 24–29 April 1988. [CrossRef]
2. Kim, J.-O.; Khosla, P. Real-time obstacle avoidance using harmonic potential functions. *IEEE Trans. Robot. Automat. Robot. Autom.* **1992**, *8*, 338–349. [CrossRef]
3. Feder, H.J.S.; Slotine, J.-J.E. Real-time path planning using harmonic potentials in dynamic environments. In Proceedings of the IEEE International Conference on Robotics and Automation, Albuquerque, NM, USA, 20–25 April 1997. [CrossRef]
4. Chengqing, L.; Ang, M.H.; Krishnan, H.; Yong, L.S. Virtual obstacle concept for local-minimum-recovery in potential-field based navigation. In Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings, San Francisco, CA, USA, 24–28 April 2000. [CrossRef]
5. Sfeir, J.; Saad, M.; Saliah-Hassane, H. An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment. In Proceedings of the IEEE International Symposium on Robotic and Sensors Environments (ROSE), Montreal, QC, Canada, 17–18 September 2011. [CrossRef]
6. Chen, L. UUV path planning algorithm based on virtual obstacle. In Proceedings of the IEEE International Conference on Mechatronics and Automation, Tianjin, China, 3–6 August 2014. [CrossRef]
7. Hossain, M.A.; Ferdous, I. Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. *Robot. Auton. Syst.* **2015**, *64*, 137–141. [CrossRef]
8. Lin, P.; Choi, W.Y.; Lee, S.-H.; Chung, C.C. Model predictive path planning based on artificial potential field and its application to autonomous lane change. In Proceedings of the 20th International Conference on Control, Automation and Systems (ICCAS), Busan, Republic of Korea, 13–16 October 2020. [CrossRef]
9. Souza, R.M.J.A.; Lima, G.V.; Morais, A.S.; Oliveira-Lopes, L.C.; Ramos, D.C.; Tofoli, F.L. Modified artificial potential field for the path planning of aircraft swarms in three-dimensional environments. *Sensors* **2022**, *22*, 1558. [CrossRef] [PubMed]
10. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [CrossRef]
11. Khosla, P.; Volpe, R. Superquadric artificial potentials for obstacle avoidance and approach. In Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 24–29 April 1988. [CrossRef]
12. Volpe, R.; Khosla, P. Manipulator control with superquadric artificial potential functions: Theory and experiments. *IEEE Trans. Syst. Man Cybern.* **1990**, *20*, 1423–1436. [CrossRef]
13. Weerakoon, T.; Ishii, K.; Nassiraei, A.A.F. An artificial potential field based mobile robot navigation method to prevent from deadlock. *J. Artif. Intell. Soft Comput. Res.* **2015**, *5*, 189–203. [CrossRef]

14. Xia, X.; Li, T.; Sang, S.; Cheng, Y.; Ma, H.; Zhang, Q.; Yang, K. Path planning for obstacle avoidance of robot arm based on improved potential field method. *Sensors* **2023**, *23*, 3754. [CrossRef]
15. Li, X.; Li, G.; Bian, Z. Research on autonomous vehicle path planning algorithm based on improved RRT* algorithm and artificial potential field method. *Sensors* **2024**, *24*, 3899. [CrossRef]
16. De Medio, C.; Oriolo, G. Robot obstacle avoidance using vortex fields. In *Advances in Robot Kinematics*; Springer: Vienna, Austria, 1991. [CrossRef]
17. Szczepanski, R. Safe artificial potential field—Novel local path planning algorithm maintaining safe distance from obstacles. *IEEE Robot. Autom. Lett.* **2023**, *8*, 4823–4830. [CrossRef]
18. Melchiorre, M.; Scimmi, L.; Salamina, L.; Mauro, S.; Pastorelli, S. Robot collision avoidance based on artificial potential field with local attractors. In Proceedings of the 19th International Conference on Informatics in Control, Automation and Robotics, Lisbon, Portugal, 14–16 July 2022. [CrossRef]
19. Abdi, M.I.I.; Khan, M.U.; Güneş, A.; Mishra, D. Escaping local minima in path planning using a robust bacterial foraging algorithm. *Appl. Sci.* **2020**, *10*, 7905. [CrossRef]
20. Montiel, O.; Orozco-Rosas, U.; Sepúlveda, R. Path planning for mobile robots using Bacterial Potential Field for avoiding static and dynamic obstacles. *Expert Syst. Appl.* **2015**, *42*, 5177–5191. [CrossRef]
21. Lee, S.; Park, J. Cellular robotic collision-free path planning. In Proceedings of the Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments, Pisa, Italy, 19–22 June 1991. [CrossRef]
22. Li, Q.; Wang, L.; Chen, B.; Zhou, Z. An improved artificial potential field method for solving local minimum problem. In Proceedings of the 2nd International Conference on Intelligent Control and Information Processing, Harbin, China, 25–28 July 2011. [CrossRef]
23. Findi, A.H.; Marhaban, M.H.; Kamil, R.; Hassan, M.K. Collision prediction based genetic network programming-reinforcement learning for mobile robot navigation in unknown dynamic environments. *J. Electr. Eng. Technol.* **2017**, *12*, 890–903. [CrossRef]
24. Yao, Q.; Zheng, Z.; Qi, L.; Yuan, H.; Guo, X.; Zhao, M.; Liu, Z.; Yang, T. Path planning method with improved artificial potential field—A reinforcement learning perspective. *IEEE Access* **2020**, *8*, 135513–135523. [CrossRef]
25. Li, Q.; Ma, Q.; Weng, X. Dynamic path planning for mobile robots based on artificial potential field enhanced improved multiobjective snake optimization (APF-IMOSO). *J. Field Robot.* **2024**, *41*, 1843–1863. [CrossRef]
26. Kozhubaev, Y.; Yang, R. Simulation of dynamic path planning of symmetrical trajectory of mobile robots based on improved A* and artificial potential field fusion for natural resource exploration. *Symmetry* **2024**, *16*, 801. [CrossRef]
27. Wang, P.; Gao, S.; Li, L.; Sun, B.; Cheng, S. Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm. *Energies* **2019**, *12*, 2342. [CrossRef]
28. Lv, Q.; Hao, G.; Huang, Z.; Li, B.; Fu, D.; Zhao, H.; Chen, W.; Chen, S. Localized path planning for mobile robots based on a subarea-artificial potential field model. *Sensors* **2024**, *24*, 3604. [CrossRef]
29. Borenstein, J.; Koren, Y. Real-time obstacle avoidance for fast mobile robots. *IEEE Trans. Syst. Man Cybern.* **1989**, *19*, 1179–1187. [CrossRef]
30. Jung, J.-H.; Kim, D.-H. Local path planning of a mobile robot using a novel grid-based potential method. *Int. J. Fuzzy Log. Intell. Syst.* **2020**, *20*, 26–34. [CrossRef]
31. Dalai, S.; Irfan, M.; Singh, S.; Kishore, K.; Akbar, S.A. Heuristic guided artificial potential field for avoidance of small obstacles. In Proceedings of the 21st International Conference on Control, Automation and Systems (ICCAS), Jeju, Republic of Korea, 12–15 October 2021. [CrossRef]
32. Park, M.G.; Lee, M.C. A new technique to escape local minimum in artificial potential field based path planning. *KSME Int. J.* **2003**, *17*, 1876–1885. [CrossRef]
33. Ren, J.; McIsaac, K.A.; Patel, R.V.; Peters, T.M. A potential field model using generalized sigmoid functions. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2007**, *37*, 477–484. [CrossRef] [PubMed]
34. Park, M.G.; Lee, M.C. Real-time path planning in unknown environment and a virtual hill concept to escape local minima. In Proceedings of the 30th Annual Conference of IEEE Industrial Electronics Society, IECON 2004, Busan, Republic of Korea, 2–6 November 2004. [CrossRef]