International Journal of Computational Intelligence Systems
Vol. **14(1)**, 2021, *pp.* 1066–1077
DOI: https://doi.org/10.2991/ijcis.d.210305.002; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

# Modified 2-Way Wavefront (M2W) Algorithm for Efficient Path Planning

Ayesha Maqbool[1,*], Alina Mirza[2], Farkhanda Afzal[3]

[1]*Department of Computer Science, NBC, National University of Sciences and Technology, Islamabad, Pakistan*
[2]*Department of Electrical Engineering, MCS, National University of Sciences and Technology, Islamabad, Pakistan*
[3]*Department of H&BS, MCS, National University of Sciences and Technology, Islamabad, Pakistan*

**ARTICLE INFO**

*Article History*

Received 16 Jan 2020
Accepted 15 Feb 2021

*Keywords*

Robotic path planning
Safe path generation
Wavefront algorithms
Self-organizing maps
Artificial potential fields
Navigational function
Glasius model

**ABSTRACT**

In this paper modified 2-way wavefront algorithm(M2W) is introduced for the discretized path planning problem. The proposed scheme uses the Glasius model, wavefront navigational function, and adaptation of Artificial Potential Fields (APF) for effective obstacle avoidance. Unlike the APF, it does not suffer from local minima, and it addresses the shortcoming of the navigational method by generating paths that are not "too close" to obstacles. Furthermore, compared to the Glasius model, M2W's computation time is significantly reduced, especially in a complex workspace with a higher density of obstacles. The proposed algorithm is also simulated with an additional set of planning constraints to demonstrate the adaptability of the M2W for constraint planning problems.



## 1. INTRODUCTION

For the last six decades, with maturity in robotics and computers, numerous path planning methods have evolved addressing different needs and requirements. Each method has its strengths and weaknesses, with no one fix for all; the specific system requirements mostly dictate the choice of planning methods. Existing work for robot path planning can be divided into two broad categories: (i) graph-based methods and (ii) discretized space methods. All graph-based methods generate a geometrical representation of the free and obstacle spaces. These geometric models are then searched for the collision-free paths [1].

A popular method for the graph-based approach is to map the free space onto the vertices and edges of a Voronoi diagram [2,3] and then search for the minimum path over the vertices of the graph using search strategies. These search strategies may vary according to specifics objectives [4]. Similarly, cell decomposition methods [5,6] divide the free space into nonoverlapping variable-size cells of a specific geometric orientation, e.g., rectangles, polygons. It is important to note here that all graph-based methods still require exhaustive searching method.

Real-time path planning requires real-time mapping of the workspace and reconstruction of the free and obstacle spaces' dimensions and definitions. A logical solution to this problem is discretizing the workspace in a grid-like manner into fixed-sized units, commonly known as cells. Research on discretized methods focuses on two areas: i) mapping and ii) optimization. The mapping strategies focus on creating grid-like probabilistic maps of the workspace, and optimization strategies look for the best solution over these maps. Artificial Potential Field (APF) [7–10] is a good example of a nongeometric mapping where the targets are mapped as an attractive field and the obstacles as the repulsive field. In APE each point of workspace is mapped as a differentiable potential function of distance either from target or obstacle. These functions normally obey Laplace's equation. The path planning is then performed by simulating motion from the highest potential to the lowest potential. However, the workspace modeled by APF suffers from local minima, especially when mapping convex-shaped obstacles. Path generation by gradient search over an APF containing local minima will result in the robot getting stuck at some intermediate state before reaching the target location. This can be avoided by performing the search over discretized workspaces for a globally optimum solution using optimization algorithms [11–16]. The advantage of these algorithms is the detailed mapping of the decision space [13,17–20] and the disadvantage is the expensive computation as most of the optimization methods are nonpolynomial (NP) hard.

Another class of mapping algorithms called navigational functions employs a variation of the A* method for workspace mapping. The navigational function maps the workspace as a function of the

*Corresponding author. Email: ayesha.maqbool@mcs.edu.pk

distance from the target in the form of a wavefront filling all gaps between the obstacles. Starting from the target location the algorithm searches for the un-marked free states with the minimum cost, i.e., the minimum distance from the current state. The states are marked with the accumulated cost in an incremental order. The newly marked states are then considered as the current state to be processed. The method requires maintaining priority queues for the list of current, marked, un-marked and next to process states. These navigational functions [21,22] do not suffer from local minima and thus do not require any optimization method for valid path selection. Having a single minimum point assures valid path generation even by gradient search over the navigational map. However, the approach suffers from the "too close" problem. It is evident that the algorithm is entirely focused on the distance from the target, and the distance from the obstacles is wholly ignored in the original set up. Thus generated path may run too close to obstacles with the possibility of the robot crashing into the obstacle. **Safe path** not only avoids obstacles but also maintains a suitable distance from obstacles. In search for shortest path navigational method generates paths that traverses the obstacle boundaries thus creating maneuvrability issues for robot. Safe path compromise the generation of least distance by adding suitable buffer between obstacle and robot.

In Decison theoretic based methods, path planing in discretized workspaces is performed using optimization algorithms like fruitfly optimization [23], genetic algorithms [24], Particle Swarm Optimization [25], A*, D* [13,14], value, or policy iteration [15,16,26]. The common principle with these optimization approaches is the allocation of some punishment for moving toward obstacles and reward for moving toward the target. The selection of the best path is then performed by searching for ordered series of actions resulting in maximum accumulative reward. The advantage of these algorithms is the detailed mapping of the decision space [13,17–19,24] and the disadvantage is the expensive computation as most of the optimization methods are NP hard even in the best case scenario.

Discretized mapping is very similar to a self-organizing map (SOM) a variation of neural network. The commonly used SOM model for path planning is the Glasius model [27] which states that the value of neuron $\lambda_i$ in discrete time is determined by the sigmoid function $g$ of the weighted sum of the neighboring of neurons $\lambda_j$ (determined by the function $w_{ij}$ plus the external input $I_i$

$$\lambda_i(t+1) = g\left(\sum_j^n w_{ij}\lambda_j(t) + I_i\right) \text{ where } i \neq j \tag{1}$$

These path planning models are the modified form of Cohen–Grossberg neural networks [28]. In existing models for path planning [29–32] the output of the neuron is mapped as a function of the sum of its neighboring neurons. SOM suffers from an undefined number of iterations and a lack of guaranteed convergence.

In general, the discretized methods appear to be a good fit for real-time dynamic path planning requirements. However, existing individual schemes lack in either one or other aspect of the performance. The proposed scheme, modified 2-way wavefront (M2W), combines the wavefront navigational function and SOM with the inspiration from the APF for effective obstacle avoidance. As a result, the approach does not suffer from local minima and generates the guaranteed shortest path efficiently and without potential conflicts by getting too close to obstacles. The algorithm works in the presence of both static and dynamic obstacles of any shape or size. M2W addresses the shortcoming of the navigational method by generating paths that are not "too close" to obstacles. Unlike the APF, the maps do not have local minima, and the computation time is significantly improved with respect to the workspace's complexity. In short, M2W meets all requirements efficiently. Table 1 highlights the benefits of using M2W in comparison to the existing methods.

In this paper, we aim for a simple yet efficient stand-alone method of path planning that can facilitate workspace mapping and safe path generation at a sufficient level of abstraction to facilitate the real-time path planning in dynamic environments. Key features of the proposed algorithm are

- It generates an optimum path from source to the destination point.
- It considers the safety of an agent by avoiding obstacles at a safe distance.
- It guarantees convergence, i.e., either return a feasible path or report unreachable destination.
- It is computationally feasible in terms of execution time.
- It provides a simple and easy to way to update changes in the environment, obstacles, moving target, etc.

This paper presents two main contributions

1 Firstly, detailed mathematical and algorithmic details of M2W are provided to address the challenges of existing path planning techniques, primarily focusing on discretized methods.

2 Secondly, it establishes the soundness of M2W by simulating six path planning scenarios (U-shaped, Spiral Maze, Moving Target, three-dimensional, and configuration space planning). In each simulation, the suitable modification and performance aspects of M2W are elaborated.

In the rest of the paper that follows, problem description layout and algorithmic details of the M2W path planning are presented in Section 2. To establish the soundness of the proposed algorithm simulated results are presented in Section 3. The conclusion is presented in Section 4.

## 2. M2W PATH PLANNING

### 2.1. Path Planning Formulation

Path planning involves searching for optimum, safe, and obstacle-free routes from a given starting location to the specified target location. The main elements of interest here are the agent/robot, obstacles, initial location of agent/robot, and its respective final location. In our work, we follow the definition of the path planning problem in terms of sets and spaces [33]. For an agent/robot $A$:

- The region of interest for path planning is defined as the workspace $W$, usually some subset of Euclidean space $\mathbb{R}^z$ with $z = 2$ or 3. The workspace defines the boundaries of the system.

**Table 1** Comparison: M2W vs. existing methods.

| Methods Features | Geometric Methods | Discretized Methods | | | | |
|---|---|---|---|---|---|---|
| | | APF | Navigational Method | SOM | Decision Theocratic | M2W |
| **Convergence** | Not Guaranteed | Local Minima | Too Close Problem | Guaranteed | Guaranteed | Guaranteed |
| **Real-Time Update** | Expensive & Complex | Simple | Simple | Simple | Complex | Simple |
| **Search** | Sophisticated search | Sophisticated search | Steepest Descent Method | Steepest Descent Method | Sophisticated search | Steepest Descent Method |
| **Deterministic Runtime** | Yes | Yes | Yes | No | Yes | Yes |
| **Safety** | Yes | Yes | No | Yes | Yes | Yes |
| **Optimum path** | Not Guaranteed | Not Guaranteed | Yes | Yes | Yes | Yes |

APF, Artificial Potential Fields; M2W, Modified 2-Way Wavefront; SOM, self-organizing map.

In a discretized workspace, $W$ is structured as a grid of equally sized units commonly known as cells or states.

- The obstacles $O_j, j = 1, \ldots, n$ are the rigid bodies that limit the agent's motion. The orientation or configuration of all the obstacles in $W$ is represented as $C_{obs}$.
- The obstacle-free region (free space) where $A$ is allowed to move is defined as $C_{free}=W/C_{obs}$.
- Valid paths from an initial configuration $C_s$ of the agent to the final configuration $C_f$ are the closed real intervals into the free space.

The path planning problem is then the generation of a continuous sequence of the positions and orientations of $A$ from some starting position $a$ and configuration $C_s$ to some final position $T$ and configuration $C_f$ without running into $C_{obs}$. Figure 1 illustrates the sets and spaces for the path planning problem with a triangular agent moving from point $a$ to $T$ and from configuration $C_s$ to $C_f$ respectively. Note that the workspace is discretized; thus, the free and obstacle spaces are divided into homogeneous cells or states of equal size.

## 2.2. Theoretical Background

The proposed scheme maps the free space as a distance map defined over the obstacle-free region. This distance map is analogous to a wavefront navigational function that calculates the free space map as aggregate Euclidean distances between obstacle-free states. The inspiration taken here for the navigational function is a Lyapunov-like function [34] that guarantees a valid path from any approachable state in the system to the target state. This scheme assures that the agent will not get stuck at any intermediate location before reaching the target, and the workspace map will not suffer from local minima.

Let $\rho$ be a metric that defines the free space distance from the target state to any given state on the grid. In the absence of obstacles, $\rho$ is simply the Euclidean distance between the given state and the target. However, the presence of obstacles may result in the shortest free paths becoming longer than the Euclidean distance. Metric $\rho$ can therefore be seen as the sum of Euclidean distances over the free space. For instance, consider the Euclidean distance between point $a$ to the target $T$ with an obstacle in the direct line from $a$ to

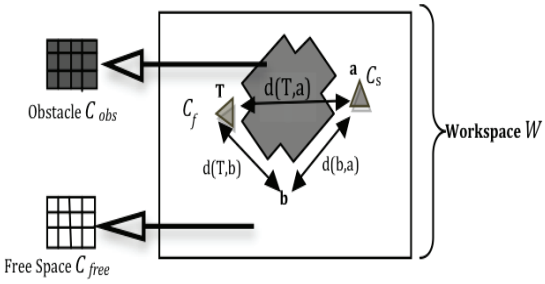



**Figure 1** Example of the path planning domain highlighting the workspace, obstacles, free space, agent, and target.

$T$. Also consider an intermediate point $b$ that is accessible through obstacle-free path from both $a$ and $T$ as shown in Figure 1. Assume the obstacle-free Euclidean distance $d(T, b)$ from point $b$ to the target location $T$ is known and also the obstacle-free Euclidean distance $d(a, b)$ from point $a$ to $b$ is known, and we want to determine the free space distance $\rho(a, T)$ from point $a$ to the target $T$. It is evident that free space distance $\rho(a, T)$ from $a$ to the target $T$, is the sum of the distance from point $a$ to point $b$ and the from $b$ to the target location $T$, i.e., $d(a, b) + d(b, T)$. This definition of the free space metric is extendible to any number of intermediate states with obstacle-free direct path between themselves and states $a$ and $T$, i.e., $\rho(a, T)=d(a, b_1)+d(b_1, b_2)+d(b_2, b_3), \ldots, d(b_p, T)$ where $b_1, b_2, \ldots, b_p$ are $p$ intermediate states.

Let $S$ be a set of states in a workspace, then the free space metric $\rho$ defined over $S$ is a sum of Euclidean distances and $\rho$ preserves all the properties of a distance metric (positiveness $\{\forall g, h \in S, \rho(g, h) \geq 0\}$ and the triangle inequality $\{\forall g, h, z \in \rho(g, h) + \rho(h, z) \geq \rho(g, z)\}$). These properties make the metric space defined by $\rho$ similar to Euclidean space in the absence of the obstacles. States that cannot be reached directly from the target location due to obstacles can be reached by travelling longer distances through free space around obstacles. Such states are effectively mapped to states that are farther from the target than the Euclidean distance of the same state. The resulting metric space defined by $\rho$ is homeomorphic to Euclidean space. This metric space, which is essentially formed by stretching the original free space because of obstacles using the metric $\rho$, is called the free space manifold $M_{free}$.

## 2.3. Mapping of Free Space to Free Space Manifold

The previous section established the layout of the optimal workspace map for the path planning by defining the free space manifold. Here lies the main challenge of path planning, i.e., to map the workspace $W$ to $M_{free}$. To calculate the metric $\rho$, we must calculate the sum of Euclidean distances between the free space states. In the case of obstacle-free paths, this is trivial. Still, in the presence of obstacles, we must define intermediate locations so that the path is broken down into obstacle-free path sections. The overall path length is as minimum as possible. This requires two stages. The first one is to find the obstacle-free distance from a certain location to all intermediate/target locations whose distance from the target is known. Secondly, we have to select the intermediate location with a minimum distance from the target. Figure 2 illustrates the free space mapping procedure for the problem presented in Figure 1. The two-headed arrows are the Euclidean distances between pairs of states. The optimum distance from point $a$ to $T$ is calculated by finding the sum of all distances from intermediate states to states containing $a$ and $T$ and then selecting the sum with the minimum distance. In Figure 2, it can be seen that there are many possible paths through the free space from point $a$ to $T$, and the solid lines show the optimum distance. In the proposed M2W, optimum distances are estimated as follows.

Let $f_\varphi$ be the obstacle-free Euclidean distance function that takes any two states $\theta$, $\phi$ of the workspace and returns the obstacle-free Euclidean distances between the states or returns $\infty$ if there is an obstacle in the way:

$$f_\varphi = \begin{cases} d(\theta,\phi) & \text{free distance between } \theta \text{ and } \phi \\ \infty & \text{otherwise} \end{cases} \tag{2}$$

Define $f_\rho$ to be the function that returns $\rho(a, T)$ by searching for the state in the neighborhood of $a$ that is closest to the target location $T$

$$\begin{aligned} f_\rho(\theta, T) = min((f_\varphi(\theta,\eta(1)) + f_\varphi(\eta(1), T)), \\ (f_\varphi(\theta,\eta(2)) + f_\varphi(\eta(2), T)), \cdots, (f_\varphi(\theta,\eta(k)) \\ + f_\varphi(\eta(k), T))) \end{aligned} \tag{3}$$

where $\eta = \Omega(\theta, r_d)$ is a list of $k$ neighboring states that are returned by neighborhood function $\Omega$. $\Omega$ returns the states that in certain radius $r_d$ of the state $\theta$, and $i = 1, \ldots, k$ is the number of states set selected by $\Omega$. The whole setup depends upon $f_\varphi(\theta, \phi)$, i.e., identifying the states that can be directly accessed from a given state. The complexity of the function depends upon the degree of the neighborhood in which the search is performed. For instance, finding the free path to a state with several states between the target and destination states will require the traversal of all states existing in the direct path between the two states to verify whether all intermediate states are free. Similarly, for the $\eta$, the increase in radius $r_d$ of the neighborhood increases the number of states retrieved and consequently increases the number of states compared to find the minimum distance in $f_\rho$.

The optimum choice for the states to be searched would be the ones in the immediate neighborhood, i.e., $r_d = 1$ as it returns the minimum number of states. The navigational planning performs the same neighborhood search by selecting only the states with the minimum associated cost.

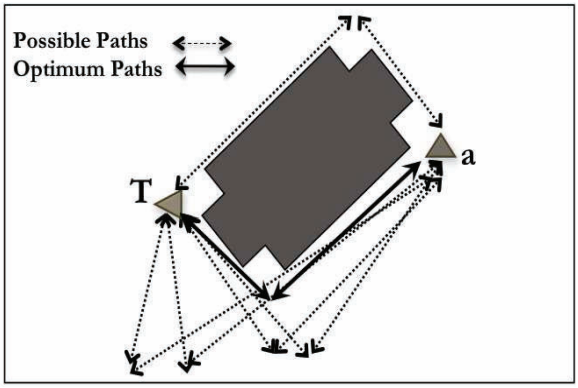



**Figure 2** Free space mapping considering only one intermediate point. Candidate paths, out of many only few are shown as dotted lines and optimum path by solid lines.

## 2.4. Workspace Model for Efficient Path Planing

The proposed system has similar setup to the Glasius model [27] which states that the value of neuron $\lambda_i$ in discrete time is determined by the sigmoid function $g$ of the weighted sum of the neighboring of neurons $\lambda_j$ (determined by the function $w_{ij}$) plus the external input $I_i$

$$\lambda_i(t+1) = g\left(\sum_j^n w_{ij}\lambda_j(t) + I_i\right) \text{ where } i \neq j \tag{4}$$

In our work, for the workspace composed of $n$ states, the value of state $\sigma_i$ is defined by the $f_\rho$ instead of the sum of neighboring neurons ($\sum_j^n w_{ij}\lambda_j(t)$) as in Glasius model. Let $\tau_i(t)$ be a function that determines the tuning value for $\sigma_i$ at any given time $t$ by considering the value of states in a certain region of the neighborhood

$$\tau_i(t+1) = min(w_{ij}\sigma(t) + \beta(\sigma_i)\zeta_{\sigma i \sigma j}) + I_i \text{ where } i \neq j \tag{5}$$

The $w_{ij}$ is the weight matrix that performs the same function as that of the neighborhood function $\Omega$ (Eq. (3))

$$w_{ij} = \begin{cases} 1 & d(\sigma_i, \sigma_j) \leqslant c_d \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $c_d$ is the neighborhood limiting constant Euclidean distance that defines the boundary of the neighborhood. $\zeta_{\sigma_i\sigma_j}$ is the cost of moving from state $\sigma_i$ to $\sigma_j$ in terms of the minimum Euclidean distance. The states are arranged in the form of a $r \times c$ grid of $n$ states. The sensory input to the system regarding the obstacle, target, free states is defined by the $r \times c$ matrix $I$:

$$I(\sigma_i) = \begin{cases} 1 & \sigma_i \text{ is the target state} \\ \infty & \sigma_i \text{ is obstacle} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

In order to address the "too close" problem an $r \times c$ matrix $\beta$ is maintained. It computes the repulsive effect of the obstacles and maps the states closer to obstacles as states being farther from the target. Depending upon the number of obstacle states $\mu$ in the neighborhood i.e., the number of $I(\sigma_i)$ with the value $\infty$. The range of the repulsion is mapped by a normalizing factor $K$. The effect of repulsive force gets weaker with the distance, thus a constant cost $\gamma$ is subtracted from the maximum repulsive states in the neighborhood:

$$\psi(\sigma_i) = max(w_{i1}\beta(\sigma_n), \ldots, w_{in}\beta(\sigma_n), K\mu) \tag{8}$$

$$\beta(\sigma_i) = \begin{cases} \psi(\sigma_i) - \gamma & if\&sp; \psi(\sigma_i) \geqslant 1 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

## 2.5. Efficient Algorithm for Optimum Path Generation

Algorithm 1 provides the details of M2W working, here $\beta(\sigma_i)$ is computed in real time along with the $\sigma_i$ and its value depends upon the order in which the states are processed and computed. The repulsive decrement $\gamma$ is the rate with which the repulsive force is reduced with increase in the distance from the obstacle. Care has to be taken in case of states belonging to the free space whose $\tau(\sigma_i)$ is undefined for a certain neighborhood under consideration. In such a case $\gamma$ in Eq. (9) may make $\tau(\sigma_i)$ return a negative value. Thus until a free state is surrounded by the states whose either distances to target are unknown or they belong to the obstacle state set, $\sigma_i$ should be left unchanged i.e., 0. Any given state's values should only change when one of the neighboring state's value gets assigned according to the free space distance from the target. That is the reason for the value $\sigma_i$ of state $i$ to be

$$\sigma_i(t) = \begin{cases} \tau_i & \tau_i(t) \geqslant 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

**Algorithm 1: Algorithm for M2W Path Planner**

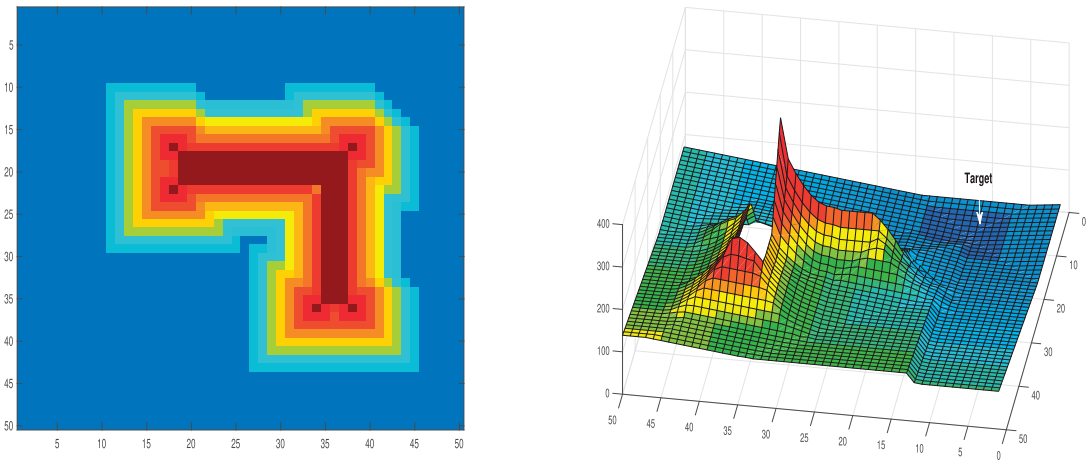
1: Initialize $n$ states of the workspace grid with $r \times c$ dimensions $\sigma_{[r \times c]} \leftarrow 0$
2: Set the target state to one $\sigma_T \leftarrow 1$
3: Initialize iteration counter $i \leftarrow 0$
4: Set maximum radius of wave $s \leftarrow$ maximum(r, c)
5: **while** There are any unmapped states or number of iterations are less than the number of states (Any($\sigma_{r \times c} = 0$) or ( $i \leq n$)) **do**
6: Increment iteration counter $i \leftarrow i + 1$
7: **for** Outward Wave: States form target state to the maximum radius $k = 1$ to $s$ **do**
8: Select neighboring States $\eta \leftarrow \Omega(\sigma_T, k)$
9: **for all** Neighboring states $\sigma_j$ in $\eta$ **do**
10: Evaluate $\psi(\sigma_j), \beta(\sigma_j), \tau(\sigma_j), \sigma_j$ {Equations (6), (8), (9), (10)}
11: **end for**
12: **end for**
13: **for** Inward Wave: States starting from maximum radius to target state $k = s$ to 1 **do**
14: Select neighboring States $\eta \leftarrow \Omega(\sigma_T, k)s$
15: **for all** $\sigma_j$ in $\eta$ **do**
16: Evaluate $\psi(\sigma_j), \beta(\sigma_j), \tau(\sigma_j), \sigma_j$ {Equations (6), (8), (9), (10)}
17: **end for**
18: **end for**
19: **end while**

Each state's value depends upon the neighboring state; the order in which the states are processed is of great importance. At the beginning of the path planning process, all states are set to arbitrarily high values to facilitate the minimum operation, and only the target state is set as one. The states in the immediate neighborhood, i.e., at the unit distance from the target, are processed first and then the ones at two units away etc. until the entire workspace is processed. The states are processed in the form of a circular wave going outward from the target state. Here the proposed algorithm is different from the navigational function [21,22] as it does not maintain any queues, so our method is more efficient in the absence of obstacles. And in lack of queues of free space, the states hidden behind concave obstacles do not get aligned to the free space by unidirectional updates. Thus the reverse wave originating from the boundary moving inward and terminating at the target state is implemented. The reverse wave not only maps the states covered by concave U-shaped obstacles, but it also helps in mapping the repulsive effect of the obstacles by using matrix $\beta$.

It can be seen in Eq. (9) that as $\beta(\sigma_i)$ adds to the distance of the states surrounding the obstacles, it does not create any local minima. In the proposed setup, the reverse wave maps the obstacle surrounded states with the reference of the closest free state, i.e., the escaping path; thus, the states are mapped as $\rho$. Thus $\psi$ obstacle repulsion function Eq. (8) depends upon $K\mu$, the scaled value of the total number of obstacle states in the neighborhood of a given state. The path generated in the sum of the simple obstacles may pass nearer to the corner's obstacle. The free space state existing right next to the corner of the $C_{obs}$ has comparatively fewer obstacle states in the neighborhood. As $K\mu$ depends upon the intensity of $\mu$, the problem of mapping corners can be easily eradicated by choosing an appropriate mechanism for maintaining the $\mu$ count. In the following simulations, this is achieved by a simple mapping as $\mu = 1.5 * n_{Max} - \mu$ where $n_{Max}$ is the maximum number of neighboring states, i.e., eight. In comparison to the navigational function, the proposed work also prefers the obstacle-free paths over the paths closer to obstacles. The $\beta$ matrix is updated with the $\sigma_i$. This makes the repulsive force depend upon the complexity of the shape of obstacle space.

In the case of more complex obstacles, the repulsive effect is more substantial near convex structures. Figure 3a shows the $\beta$ matrix of an "L-shaped" obstacle, and it can be seen that the effect of the repulsion is more substantial near the turn/curve. The target state is near the bottom end of the workspace; Figure 3b shows the $W_\rho$ mapped workspace having the target as the local minima; the empty space is the obstacle since $\sigma_i = \infty$. The states near the obstacle boundary have higher $\rho$ values than the neighboring states, but the overall map is smooth without any local minima.

There is a possibility for the set of free states to be surrounded by the obstacle in such a way that there is no free path leading toward the target state. The current work identifies such a case by maintaining a counter of the number of iterations and forcing the




(a) $\beta(\sigma_i)$ Repulsive effect of obstacles. (b) Free-space map of the workspace

**Figure 3** | Free space mapping of mission space with an L-shaped obstacle. The mapping with safety considerations prohibits the robot from getting too close to an obstacle.

algorithm to terminate if the number of iterations exceeds the number of free states. This brute force selection of the terminating criterion is adopted only to elaborate on the proposed work's worst-case scenario. In the case of extremely complex workspaces where for each update wave, backward or forward, only one state gets classified. The maximum number of iterations is not more than the number of free states in the workspace. Here counter helps in the identification of the lack of solution.

## 3. SIMULATIONS

In this section, we present simulated studies carried out to demonstrate the effectiveness of the proposed approach. Simulation and The four features of path planning are demonstrated in the following simulations.

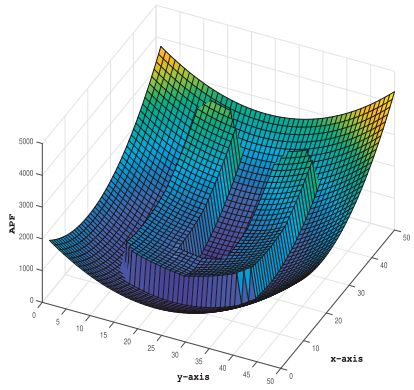
1. U-shaped Obstacles: This case outlines the better performance of M2W in the presence of U-shaped obstacles. It is shown that a. M2W generates workspace without local minima, b. the generated paths are safe and c. the shortest.
2. Spiral Maze: The spiral maze path problem is simulated to elaborate the a.the strength of discretized workspace mapping and b. relation of the execution time of algorithm with the number of free states. It is shown that in the case of complex and dense workspace, the execution time of the algorithm is reduced with a reduced number of free cells.
3. Moving targets: The simulations show how M2W performs in a dynamic environment. Each time M2W generates optimum paths to moving the target's location.
4. Three Dimension: The simulation shows how M2W can easily be adopted from 2-dimensional space to three-dimensional space.
5. Configuration Space Planning: This simulation demonstrates the adaptability of M2W with additional constraints of object orientation.

In the following examples, the workspace is represented as a two and three-dimensional lattice. We focus on mobile robot path planning and obstacle avoidance in both static and dynamic environments. As each state searches for the free space only in the immediate neighborhood, $c_d$ in Eq. (6) is set to be 1.5. By choosing $c_d$ as 1.5, each state has eight neighboring states. The states are mapped in ascending order of the distance from the target. An optimal path can be found even before the entire workspace is mapped completely. The algorithm can terminate as soon as the current state of the robot is evaluated and classified. In all simulations, the agent can move to any of the eight neighboring cells. Once the workspace is mapped, the optimum path is generated by the steepest descent method. Simulation and graphical results are carried out using Matlab.

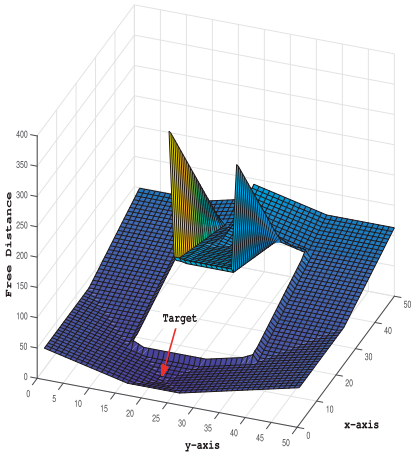
### 3.1. Case 1: U-Shaped Obstacles

The first simulation illustrates the proposed algorithm in the presence of concave obstacles. This is the typical problem where APF fails due to presence of local minima. The workspace has 2500 states arranged in a two-dimensional grid of 50 rows and 50 columns. With a single U-shaped obstacle, Figure 4, simulated results are shown to establish the performance of the path planning algorithm with respect to the complexity of concave obstacles. Figure 4 shows the comparison of APF and M2W based mapping in workspace. Here the workspace is mapped using well known harmonic APF [35].

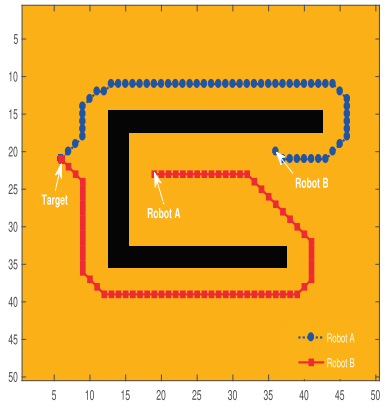
From Figure 4a, it can be seen that in the APF map, the force of attraction is stronger in the area enclosed by the obstacle than the force felt at the free path leading out from the U-shaped obstacle. This situation makes it impossible for a robot to be positioned inside the obstacle to reach the target location outside the obstacle. On the other hand, in M2W, the states are mapped as the free distance from the target location. The states enclosed by the obstacle are mapped as farther from states leading out of the obstacle. This helps M2W find a path out of the obstacle, whereas the path generated by the APF never reaches the target.




(a) The APF Mapping




(b) The Free space Mapping using M2W.




(c) Path generated by M2W

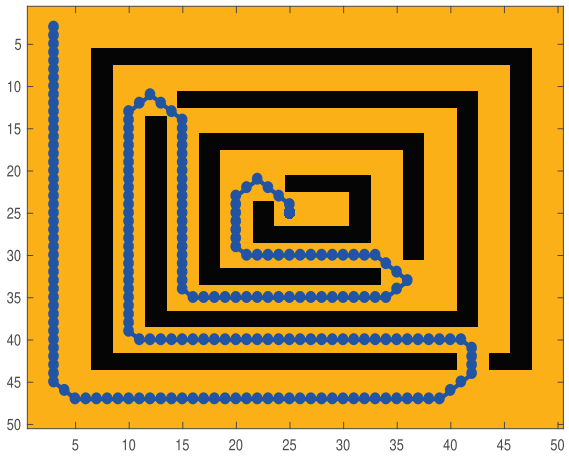
**Figure 4** The free space mapping using Modified 2-Way Wavefront (M2W) and Artificial Potential Fields (APF). The map by M2W does not suffer from local minima where as the path generated by APF never reaches to target.

The algorithm requires one iteration, updating the workspace in both directions: firstly from target to the boundary states and then from the boundary states toward the target. Two examples are shown with different initial positions. The path generated for both robots are smooth and do not suffer from local minima. The robots reach the destination state without moving toward the obstacle, i.e., from the first step to the very last step, the path is the optimum solution under the given circumstances. Furthermore, to elaborate that the optimum paths do not pass through the states too-near the obstacles, the repulsion normalizing factor $k$ from Eq. (8) is chosen to be 3.5. Even a substantial repulsion effect does not cause any irregular behavior in the optimum path.

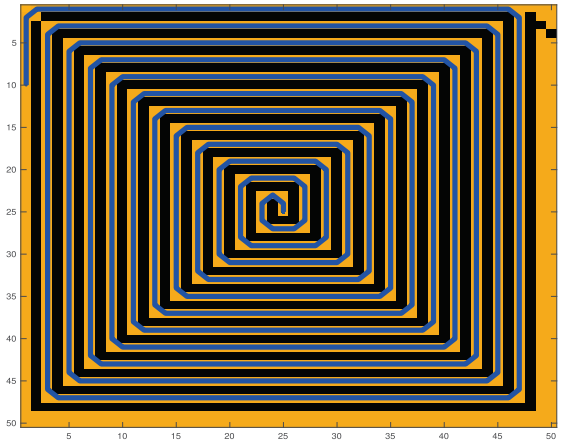
An extreme test of the M2W is shown in Figure 4c: where the first and second optimum (shortest and safe) paths for robot A have a minimal difference in terms of distance, but very significant difference in terms of the route taken. A safe path for robot A with a starting location near to the convex region is 75 steps through the smaller arm (Figure 4c: robot A's best path) and 77 steps (Figure 4c: robot A's next best path) from the long arm. Even though it may seem that robot A has to cover longer distances, whereas, on the contrary, the robot is choosing the smaller of the two safe paths. The escape route's selection through the smaller arm instead of, the longer one is evidence of the cumulative distance-based mapping, which always results in valid optimum paths.

## 3.2. Spiral Maze Problem

We now demonstrate the M2W behavior on spiral maze environments (Figure 5) again with the workspace of 50×50 cells. In Figure 5a, to escape from the center of the maze, the planner took nine iterations to map the entire workspace. On the other hand, the same scenario mapped backward, i.e., mapping from outside toward the center, took only five iterations. The number of iterations depends upon the convex obstacles limiting the states' update with respect to the outward update wave. The inward wave evaluates the free path leading to the convex obstacle. The proceeding outward wave maps the free space with respect to that path until the wave hits another convex-shaped obstacle. In the simulation



(a) Simple Maze



(b) Complex Maze

**Figure 5** | Spiral maze problem simulation.

of a simple maze with a target location at the center of the maze, the waves originating from the middle move toward the boundary states equivalently in all directions. This causes the wave to encounter the convex obstacle in at least two directions at each square-shaped obstacle before finding a path leading out of the respective square. The state's path planning at the top-left corner of the workspace took only five iterations as the wave originating from the corner is more aligned to the escape paths. Thus each square gets mapped at each iteration.

In the second denser scenario, Figure 5b, it took 21 iterations to map the maze problem with the target location in the center of the maze. Note that out of 2500 states, 1125 states are obstacle states. Even though it took 21 iterations to generate the free space map, the maximum clearance method would have taken as many iterations as that of obstacle states.

## 3.3. Moving Target and Obstacles

In a situation where the target is dynamic, and the robot has the task to chase and capture the target, the sensory information in terms of the input matrix, $I$, must be processed continuously to update the environment's changes. Figure 6 shows a simulation of such a case where the seeker robot and target are both moving with the constant rate of one block per observation. The obstacles are shown in solid black, and $c_d$ was set to 1.5. The target initially starts from near the top-left corner of the workspace, and the seeker's initial location is in the bottom center. The point marked as "A" on the target's path shows the target moving through a narrow passage, and the corresponding point A on the seeker's robot track shows a shift in the seeker's path firstly to avoid the obstacle and secondly, as a response for the target moving through the narrow passage.

At point B the robot catches up to the target, but to demonstrate, the behavior of the M2W simulation does not terminate at this point. At point B, the robot's path is shorter than that of the target, thus showing that even in case of the target's zigzag movement, the seeker will have a stable straight path resulting in the seeker catching up with the target. Point C shows the seeker's path at a safe distance

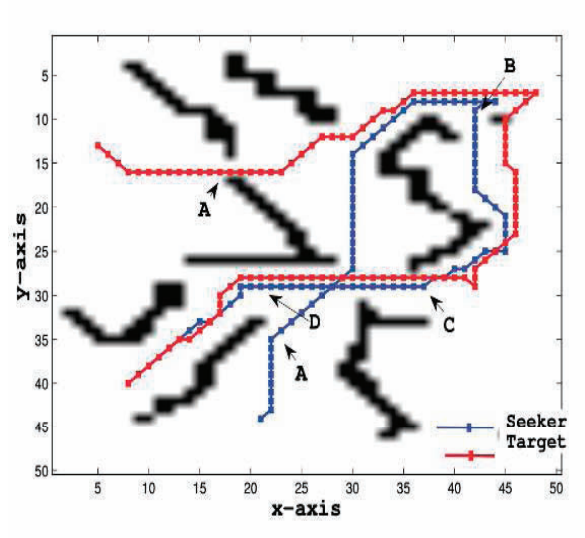
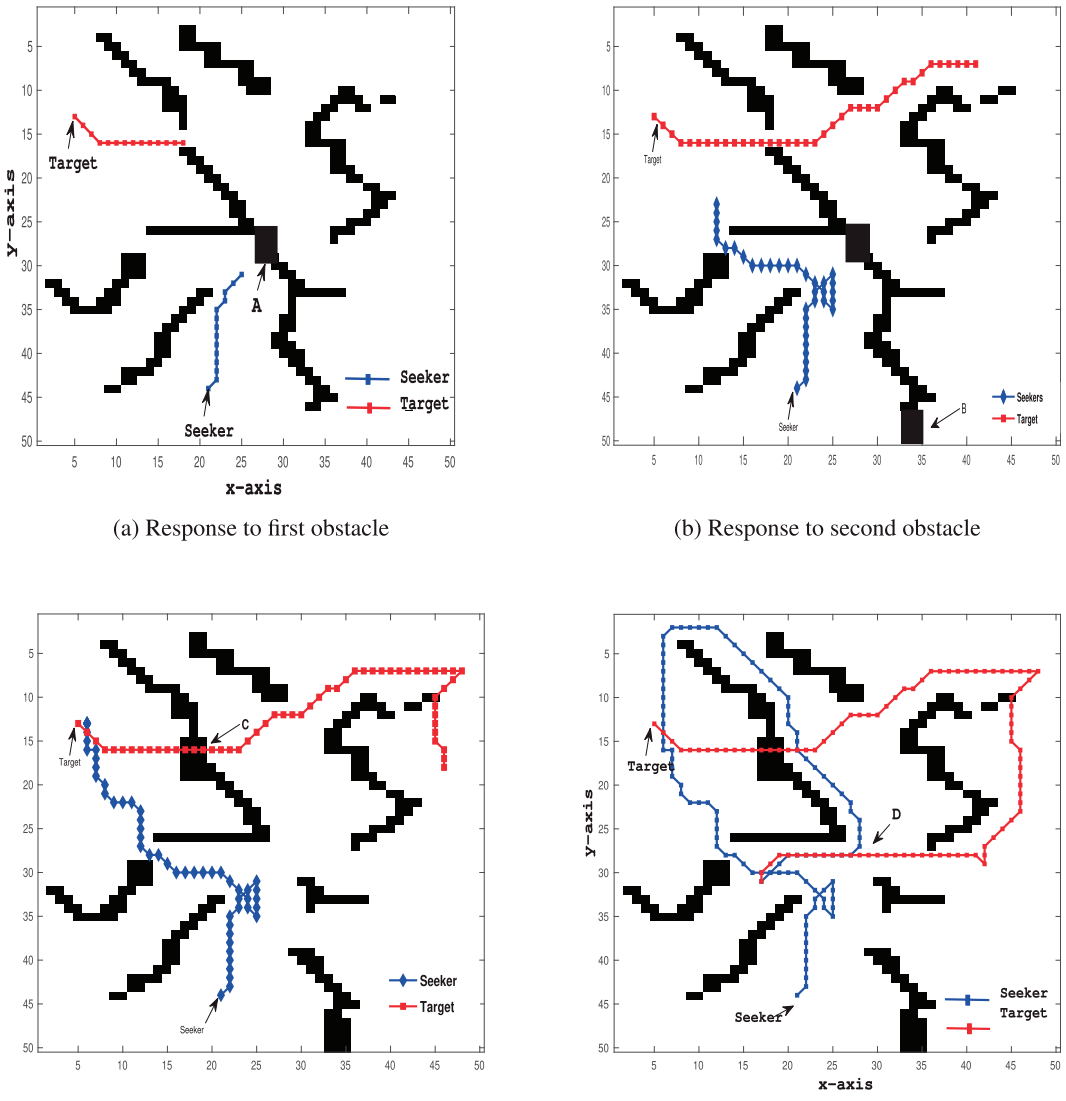



**Figure 6** | Moving Target: The path generated by using Modified 2-Way Wavefront (M2W) in pursuit of the target is stable and secure.

from the obstacle in comparison to the target that is moving very close to the obstacle. This indicates that the seeker's well-being has a priority over the capture of the target, and the seeker does not run into a potentially harmful situation while following the target.

Figure 7 shows the same workspace with a dynamic environment, having moving obstacles and moving targets. In Figure 7a, when the seeker robot approaches the point marked as A, an obstacle (shown as a black box) appears in its path, the seeker then moves toward the bottom opening near the point B. Figure 7b shows that as the seeker approaches to point B, the opening is restricted by the appearance of another obstacle. Having the path from A to B closed the seeker backtracks and moves toward the only viable opening. When it approaches point C, Figure 7c, another obstacle is introduced while the obstacle at A is removed. This dynamic behavior of




(a) Response to first obstacle

(b) Response to second obstacle

(c) Response to third obstacle

(d) Response to obstacle clearance

**Figure 7** | Moving target and obstacles: with obstacles appearing during the mission restricting the robots' path.

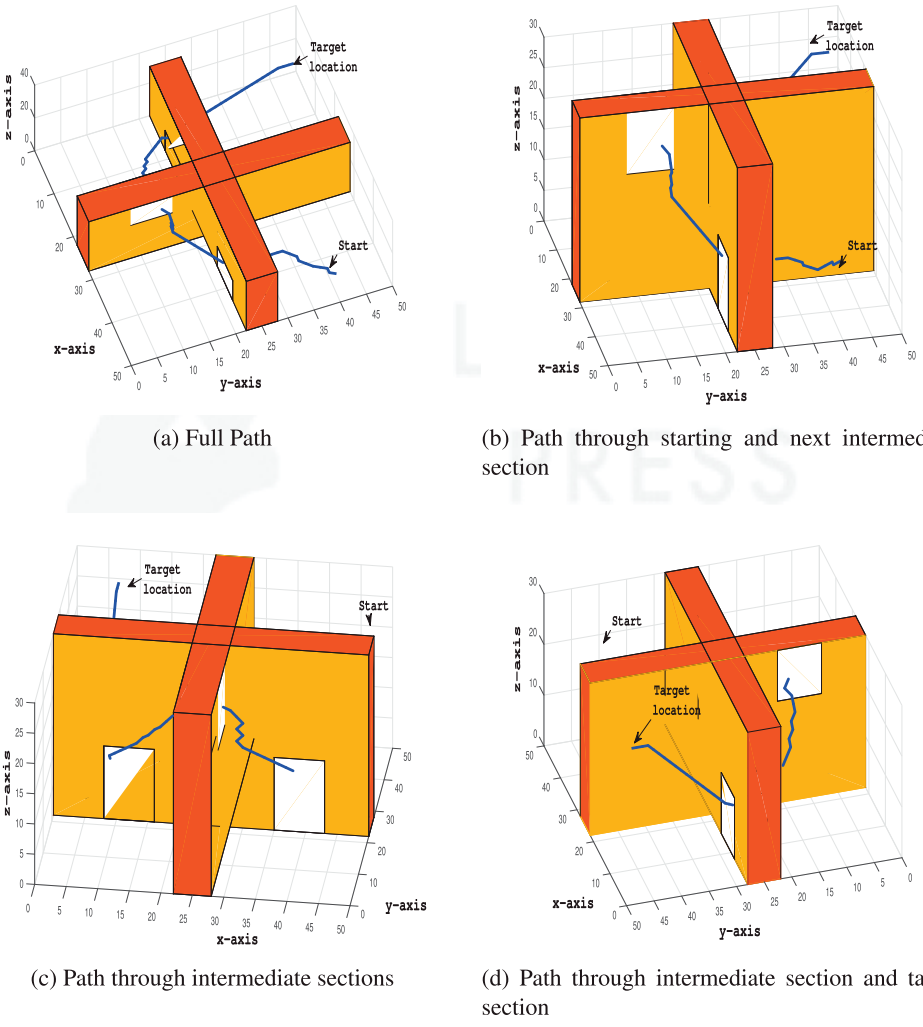
the seeker shows that the proposed method is feasible for static and dynamic environments.

## 3.4. Path Planning in Three Dimensions

M2W is easily extended from two-dimensional mapping to three-dimensional mapping. The new dimension's addition may result in an increased number of neighboring states and the total number of states in the workspace. However, the algorithm performs in the same manner, even in three dimensions, and generates safe paths from source to target location. Figure 8 shows the three-dimensional workspace of size (50 × 50 × 30), composed of four rooms/sections. To make it to the target section from the current position, the agent must go through all rooms as the wall between the source and target section lacks any opening. The intermediate room's walls have windows at either at the lower or the top edge of the wall, making agents change its elevation Figures 8b and 8c.

## 3.5. Configuration Space Planning

The path planning for a rigid body robot whose dimensions and orientation are not compatible with that of the discretized space cells is a complex problem for planner algorithms. There are many considerations to be kept in mind, such as the constraints on the motion, the selection of the orientation that best fits in a narrow path, etc. These problems are commonly known as the piano movers problem. One such problem's simulation for an L-shaped object is presented in Figure 9. Here the workspace has 400 states with 20 rows and 20 columns with the obstacles shown as solid black spaces. The $c_d$ is set to be 1.5, and the unit cost is set to be one for each step farther from the target state. Most discretized space planners require the object to be equal to or smaller in size than that of the individual cell. In the following simulation, the object is bigger than the unit cell as it spans up to 4 cells in length and two cells in width, and it can rotate to any angle. However, as the object has to obey the constraints of rigid body transforms, given the angle of orientation $\alpha$




(a) Full Path

(b) Path through starting and next intermediate section

(c) Path through intermediate sections

(d) Path through intermediate section and target section

**Figure 8** | Three-dimensional mapping using Modified 2-Way Wavefront (M2W) with workspace divided into four sections. The starting location's path to the target location has to go through the small windows in separating walls.

of the object, then $cos^2\alpha + sin^2\alpha$ should be equal to one. The second constraint on the object's motion under consideration is set to be for the consecutive steps. For two consecutive steps, the angle of the rotation should not be more than $\pm 25$ degrees.

A simple modification is performed to the M2W by introducing a matrix that keeps a record of all admissible angles at a specific state. Simultaneously, while the matrix is estimated, care is taken that if all orientations of the object, i.e., all $360^o$, overlap the obstacle space, then that state is also marked as an obstacle state. At the first iteration, all admissible angles are calculated for each state. When the state is processed the second time, e.g., in the inward wave, the second constraint's angles are marked. If none of the angles in a given state fulfills the second criteria with respect to any of the neighboring states, it is also set as an unapproachable state.

It is evident from Figure 9 that even with these simple modifications, the M2W produces a valid smooth path for the rigid body with an initial location at the top left of the figure to the target location at the bottom.

## 4. CONCLUSIONS

A new and efficient method, M2W, for robotic path planning in the discretized workspace is presented. The proposed work is inspired by the navigational method, APF, and Glasius model-based path planning. It combines the salient features of all three schemes but does not suffer from any of the shortcomings.

M2W has linear complexity in terms of the number of states in the workspace. Furthermore, the linearity depends on the shape and orientation of the obstacles regarding the target location. In the case of simple obstacles, it outperforms SOM planners as it does not require any time for neurons' activity to settle. It is more efficient than crude navigational planners in the absence of the obstacles as it does not maintain any queues. M2W does not suffer from local minima. The workspace in M2W is mapped in terms of the strictly positive and increasing function of the free space metric with respect to target location acting as a stationary point/global minima.

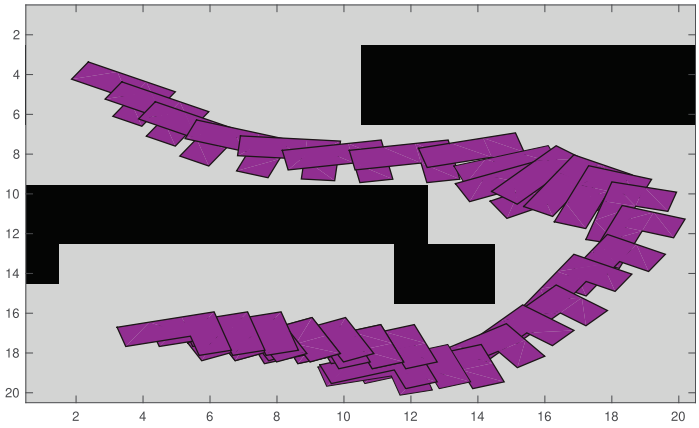



**Figure 9** | Planning for L-shaped object.

The paths generated by M2W are safe and do not lead dangerously near to the obstacles. M2W can be used in many scenarios (2D, 3D, and configuration planning) with suitable modification. It's suitable for static as well as the dynamic environment. For future work the M2W is extended for multiple co-operative agents by incorporating collective performance indicators, e.g., communication, task decomposition, uncertainty, and multiple targets.

## CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest to report regarding the present study.

## Funding Statement

The author(s) received no specific funding for this study.

## AUTHORS' CONTRIBUTIONS

All authors contributed equally.

## REFERENCES

[1] S.M. Lavalle, Planning Algorithms, vol. 1, chap. 5, Cambridge University Press, 2006, pp. 153–205.
[2] O. Takahashi, R.J. Schilling, Motion planning in a plane using generalized Voronoi diagrams, IEEE Trans. Robot. Automat. 5 (1989), 143–150.
[3] R.C. Arkin, Navigational path planning for a vision-based mobile robot, Robotica. 7 (1989), 49–63.
[4] Y. Sun, A reliability-based approach of fastest routes planning in dynamic traffic network under emergency management situation, Int. J. Comput. Intell. Syst. 4 (2011), 1224–1236.
[5] C.H. Cai, S. Ferrari, Information-driven sensor path planning by approximate cell decomposition, IEEE Trans. Syst. Man Cybern. Part B Cybern. 39 (2009), 672–689.
[6] S. Shojaeipour, S.M. Haris, M.I. Khairir, Vision-based mobile robot navigation using image processing and cell decomposition, in: H. Badioze Zaman, P. Robinson, M. Petrou, P. Olivier, H. Schröder, T.K. Shih (Eds.), Visual Informatics: Bridging Research And Practice, vol. 5857, Springer, Berlin, Heidelberg, Germany, 2009, pp. 90–96.
[7] A. Lazarowska, Discrete artificial potential field approach to mobile robot path planning, IFAC-PapersOnLine. 52 (2019), 277–282.
[8] J.O. Kim, P.K. Khosla, Real-time obstacle avoidance using harmonic potential functions, IEEE Trans. Robot. Automat. 8 (1992), 338–349.
[9] H. Chen, L. Xie, A novel artificial potential field-based reinforcement learning for mobile robotics in ambient intelligence, Int. J. Robot. Automat. 24 (2009), 245–254.
[10] F.A. Cosio, M.A.P. Castaneda, Autonomous robot navigation using adaptive potential fields, Math. Comput. Model. 40 (2004), 1141–1156.
[11] R. Kala, *et al.*, Robotic path planning using evolutionary momentum-based exploration, J. Exper. Theor. Artif. Intell. 23 (2011), 469–495.
[12] X. Chen, Y. Kong, X. Fang, Q. Wu, A fast two-stage aco algorithm for robotic path planning, Neural Comput. Appl. 22 (2013), 313–319.
[13] M. Likhachev, A. Stentz, Probabilistic planning with clear preferences on missing information, Artif. Intell. 173 (2009), 696–721.
[14] A. Stentz, Optimal and efficient path planning for unknown and dynamic environments, Int. J. Robot. Automat. 10 (1995), 89–100.
[15] S. Thrun, A. Bucken, Integrating grid-based and topological maps for mobile robot navigation, in Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, Portland, Oregon USA, 1996, vols. 1 and 2, pp. 944–950.
[16] N. Roy, S. Thrun, Motion planning through policy search, in 2002 IEEE/RJS International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 2002, vols. 1–3, pp. 2419–2424.
[17] X.C. Ji, H. Zhang, D. Hai, Z.Q. Zheng, A decision-theoretic active loop closing approach to autonomous robot exploration and mapping, in: L. Iocchi, H. Matsubara, A. Weitzenfeld, C. Zhou (Eds.), Robocup 2008: Robot Soccer World Cup XII, Lecture Notes in Computer Science, vol. 5399, Springer, Berlin, Heidelberg, Germany, 2009, pp. 507–518.
[18] G.L. Peterson, D.J. Cook, Incorporating decision-theoretic planning in a robot architecture, Rob. Auton. Syst. 42 (2003), 89–106.
[19] S. Seuken, S. Zilberstein, Formal models and algorithms for decentralized decision making under uncertainty, Auton. Agents Multi-Agent Syst. 17 (2008), 190–250.

[20] M. Tacke, T. Weigel, B. Nebel, Decision-theoretic planning for playing table soccer, in: S. Biundo, T. Frühwirth, G. Palm (Eds.), KI 2004: Advances in Artificial Intelligence, Proceedings, Lecture Notes in Computer Science, vol. 3238, Springer, Berlin, Heidelberg, Germany, 2004, pp. 213–225.
[21] A. Zelinsky, R.A. Jarvis, J.C. Byrne, S. Yuta, Planning paths of complete coverage of an unstructured environment by a mobile robot. In Proceedings of International Conference on Advanced Robotics, (1993), 533–538.
[22] E. Rimon, D.E. Koditschek, Exact robot navigation using artificial potential fields, IEEE Trans. Robot. Automat. 8 (1992), 501–518.
[23] X. Zhang, S. Xia, X. Li, Quantum behavior-based enhanced fruit fly optimization algorithm with application to uav path planning, Int. J. Comput. Intell. Syst. 13 (2020), 1315–1331.
[24] F. Liu, S. Liang, X. Xian, Optimal robot path planning for multiple goals visiting based on tailored genetic algorithm, Int. J. Comput. Intell. Syst. 7 (2014), 1109–1122.
[25] B. Xu, F. Zhou, A.M. Gates, Multi-objective particle swarm optimization algorithm for the minimum constraint removal problem, Int. J. Comput. Intell. Syst. 13 (2020), 291–299.
[26] R. Kala, K. Warwick, Intelligent transportation system with diverse semi-autonomous vehicles, Int. J. Comput. Intell. Syst. 8 (2015), 886–899.
[27] R. Glasius, A. Komoda, S. Gielen, Neural-network dynamics for path planning and obstacle avoidance, IEEE Trans. Neural Netw. 8 (1995), 125–133.
[28] M. Cohen, S. Grossberg, Neural dynamics of brightness perception: features, boundaries, diffusion, and resonance, Atten. Percept. Psychophys. 36 (1984), 428–456.
[29] K. Rahul, *et al.*, Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness, Neurocomputing. 74 (2011), 2314–2335.
[30] A.R. Willms, S.X. Yang, An efficient dynamic system for real-time robot-path planning, IEEE Trans. Syst. Man Cybern. Part B Cybern. 36 (2006), 755–766.
[31] S.X. Yang, M.Q.H. Meng, Real-time collision-free motion planning of a mobile robot using a neural dynamics-based approach, IEEE Trans. Neural Netw. 14 (2003), 1541–1552.
[32] M. Ghatee, A. Mohades, Motion planning in order to optimize the length and clearance applying a Hopfield neural network, Expert Syst. Appl. 36 (2009), 4688–4695.
[33] J.-C. Latombe, Robot Motion Planning, chap. 1, Kluwer Academic Publishers, Boston, MA, USA, 1991, pp. 3–36.
[34] J.B. Clempner, A shortest-path Lyapunov approach for forward decision processes, Int. J. Comput. Games Technol. 2009 (2009), 12–18.
[35] J.-C. Latombe, Robot Motion Planning, chap. 7, Kluwer Academic Publishers, Boston, MA, USA, 1991, pp. 289–303.