INTERNATIONAL JOURNAL OF ROBOTICS
(THEORY AND APPLICATIONS)
E-mail: IJR@kntu.ac.ir

A Novel Path Planning and Obstacle Avoidance Method for Considerable Localizing Error

S.S. Kazemi [a,*], S.F. Hosseini [a]

[a] Department of Mechanical Engineering, University of Tehran, Tehran, Iran

ARTICLE INFO

Article history:
Submit: 2018-07-17
Revise: 2025-01-04
Accept: 2025-01-04

Keywords:
Obstacle avoidance
Path planning
Occupancy grid map
Artificial neural network

ABSTRACT

Mobile robots hold significant potential for a broad spectrum of applications in industry and various service domains. Consequently, extensive research efforts have been devoted to addressing deficiencies and improving their performance. Among the critical challenges in robotics is obstacle avoidance, which enables the robot to navigate around unexpected objects encountered along a planned path. Numerous methods and algorithms have been proposed to prevent collisions between robots and detected obstacles. These approaches commonly rely on the crucial assumption of having precise knowledge of the robot's position at every step. This paper introduces a novel method for obstacle avoidance in indoor environments, leveraging an occupancy grid map of a partially known space and the A* algorithm. The proposed method addresses scenarios with imprecise information about the robot's state. Initially, a preliminary occupancy grid map is refined and transformed into an enhanced map using an artificial neural network. Subsequently, the A* algorithm is applied to the modified map. Additionally, an algorithm is developed to guide the robot from a starting point to a target endpoint. When encountering a newly emerged obstacle, the robot dynamically adapts its path to reach the goal while avoiding the obstacle. The proposed method's efficacy is validated through simulations of a two-wheeled robot in three distinct scenarios. Results demonstrate the method's capability to navigate the robot effectively within an indoor environment, even with imprecise state information. The algorithm ensures the robot maintains a safe distance from obstacles, showcasing its potential for practical applications.

* Corresponding address: University of Tehran, Tehran, Iran. E-mail address: ssk.kazemi@ut.ac.ir.

1. Introduction

In recent years, extensive research has focused on mobile robots and their diverse applications. Unlike stationary robots, mobile robots pose unique challenges, primarily due to the uncertainty associated with their position within the environment. Among these challenges, path planning and obstacle avoidance stand out as critical issues in most automated implementations of mobile robots. Specifically, when a robot encounters an unexpected object during task execution, it must adapt by formulating a new approach that accounts for the obstacle while ensuring the task continues seamlessly. This challenge becomes even more pronounced in dynamic environments, such as homes or offices, where obstacles frequently change.

To address these complexities, numerous methods and algorithms have been developed, tailored to different operational settings and performance requirements of mobile robots. These approaches aim to enhance the robot's ability to navigate effectively and maintain task efficiency, even in unpredictable scenarios.

The first proposed method was APF (artificial potential field) introduced by Khatib [1] which consider a goal as a point which attracts the robot and an obstacle as a point which repulses the robot in an environment. Other primary methods are VFH (vector field histogram) by Borenstein and Koren in [2] and DWM (Dynamic Window Method) by Fox et al in [3]. These methods are based on searching for an optimum point for an objective function. The main disadvantage of these methods is being incomplete to find a path for any arbitrary obstacle shape. FLC (fuzzy logic controllers) and ANN (artificial neural networks) are used in different papers for doing obstacle avoidance. Ganapathy et al in [4] designed an algorithm for path planning and obstacle avoidance implementing ANN and FLC in different steps of it. Samsudin et al [5] utilize an Ordinal structure FLC optimized by GA (genetic algorithm) to find a short smooth path between two points. A three-layer neuro-fuzzy network is implemented by Duta in [6] and this network is trained by PS (particle swarm) algorithm to find the shortest possible path. Mothlagh et al [7], Li and Choi [8] and Faisal et al in [9] also employ fuzzy logic controllers in order to solve the obstacle avoidance problem. Wen et al [10] proposed a virtual force between the robot and each {Ye, 2007 #11} obstacle to maintain the distance between them and a combination of ENN (Elman neural network) and the fuzzy controller is used for tracking the trajectory.

In addition, various approaches are introduced. Hwang and Chang used H_2/H_∞ controller and two CCD cameras in their paper [11]. Sgorbissa and Zaccaria [12] introduced a two-level cognitive architecture defining a roaming trail instead of a path to the goal. To clarify, a diamond is introduced to determine the boundaries of the path. APF method is improved by Tang et al in [13] and E.Shi et al in [14] by using gravity chain and changing the force function respectively. C.Shi et al improved the BCM method by adding a collision prediction model to it [15]. Jung et al [16] designed an algorithm for marathoner follower robot. In the obstacle avoidance step, each new obstacle has a radius which is calculated according to the relative speed of the robot to the robot.

In this paper, the desired purpose is to lead a two-wheeled robot from a starting point to an endpoint in a partially known environment by a 2D occupancy grid map and a laser range finder. In [17] Stachniss and Burgard proposed a method which defined a five-dimensional (x,y,θ,v,ω) configuration space and searches through its occupancy grid map. A* algorithm is utilized to reduce the search space by creating a channel of valid x-y positions. A group of VFH [2] based methods uses occupancy grid maps. Ulrich and Borenstein implement A* with VFH+ [18] method to gain a better prediction of probable collision between robot and obstacles which leads to VFH* [19]. Moreover, different derivatives are proposed based on VFH+ including VFH∗TDT [20] and TFH [21]. In [22] Seder and Petrovic represented a moving object in an occupancy grid map by moving cells. The moving object trajectory and robot possible trajectory are predicted for finding an obstacle-free map. In [23] a two-stage approach is proposed by Arras et al: Model stage (reduced DWM) and Planning stage (distance transform). Also, a method for calculation of distance function for polygonal robots is proposed.

The previous efforts for proposing a solution for path planning and obstacle avoidance didn't consider the noisy or inexact information about the robot state. Modeling the behavior of a robot is a conventional method for dealing with a system and it always brings on errors due to some primary assumptions about the system. For instance, Khorram and Moosavian in [24], Sadedel et al in [25] and Lavaei et al in [26] propose a dynamic model for three different dynamic systems. Moreover, most of the robots suffer from encoder error or sensor errors {Jung, 2014 #6}which leads to implementing different filters for achieving a better localizing output. These localizing methods still bring on errors that can't be used along with the named obstacle avoidance methods. Also, localizing takes a share of calculation time during

each iteration that may affect the robot performance while doing obstacle avoidance.

In the following, a new method is introduced to navigate a robot in a partially known environment in the presence of a considerable localizing error. In section 2 the grid map transformation is explained. Section 3 is dedicated to the details of the proposed path planning algorithm. In section 4 a controlling algorithm is proposed to keep the robot in the safe region. Sections 5, 6 and 7 include the simulation results, discussion on them and conclusion respectively.

2. Transforming the Initial Map

Occupancy grid maps are designed to represent the probability of an obstacle's presence within a given environment. However, conventional high-resolution occupancy grid maps often lack the consistency required for effective implementation in decision-making algorithms. For instance, errors introduced by sensors or encoders can result in free cells being interspersed with occupied cells, creating ambiguity in interpreting the region. This inconsistency makes it challenging to determine whether the area should be classified as occupied or free, thereby complicating subsequent decision-making processes.

For any decision-making algorithm utilizing an occupancy grid map, a reliable representation of regions is essential. Accurate classification of map cells as free or occupied is particularly critical for path planning and obstacle avoidance. To achieve this, a threshold value is employed to segregate grid cells into two categories: valid and invalid points. In this approach, the occupancy probability of each cell is represented as a value within the range [0,1]. Cells with a probability less than 0.1 are classified as valid (free), while those with a probability equal to or greater than 0.1 are considered invalid (occupied).

However, despite the high resolution of the map, the distribution of binary cells remains inconsistent and challenging to utilize effectively. For instance, valid cells may appear surrounded by invalid cells, or vice versa, leading to ambiguity. To address this, a relational framework between map cells is needed to enhance the map's representation and bring it closer to reality. In this study, a Hopfield neural network is employed to refine the occupancy grid map. This network transforms each square block of cells into one of six predefined valid states: fully filled, fully empty, top half filled, bottom half filled, right half filled, or left half filled. For clarification, if every 10 pixels of the map correspond to the robot's maximum aspect size, each 10×10 block of the map is provided as input to the network. The Hopfield network iteratively adjusts the cells within the block until the configuration converges to one of the specified patterns. The function governing this network is represented in Equation (1):

$$x_k(t) = fact(\sum_{j \in k} w_{j,k} x_j(t-1)) \qquad (1)$$

According to this function, the neuron values are changed at each step by the interaction between all neurons. The activation function is a binary threshold function or Heaviside (Fig. 1) which keeps the state of any neuron in one of two main states (-1 and 1). Network weights ($w_{j,k}$) are trained by introducing each defined pattern to the network. In [27] the details of implementing these networks are explained.

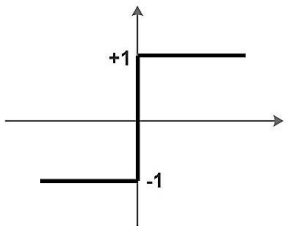

Figure 1. The activation function of the neural network

The outcome of this transformation is a more consistent and reliable map, comprising two distinct sets of cells: valid or free cells (denoted as -1) and invalid or occupied cells (denoted as +1). This refined map is generated prior to deploying the robot in the environment, ensuring an accurate representation of the workspace. The algorithms proposed in the following sections are designed to operate using this transformed map, leveraging its improved consistency for more effective decision-making and navigation.

3. Path Planning

As highlighted in the introduction, well-known obstacle avoidance methods, such as those based on the Vector Field Histogram (VFH), rely on deterministic and accurate information about the robot and its environment. However, in real-world and general applications, this assumption often does not hold, as the robot's state is typically represented by a probabilistic distribution. Consequently, the user must select a single point within this distribution to serve as the "true" state for subsequent calculations.

In methods like Dynamic Window Approach (DWA) and VFH, the robot searches within the velocity space $(v,)$ to determine an optimal motion for the next step. However, the inherent variability in the robot's belief about its state can lead to unstable motion commands. Furthermore, the computational time required for these methods is

substantial, which negatively impacts the robot's overall performance and responsiveness.

In our approach, the robot's motion is restricted to two primary actions: moving forward and twisting. Using the A* algorithm, a sequence of path points is generated from the starting point to the goal. The navigation algorithm then guides the robot through these points within a safe "tunnel" created using the specified motions, effectively ignoring the variations in the robot's state.

To ensure safe navigation of the robot, even when dealing with noisy or unsmooth states, it is essential to define a secure tunnel for its movement. This process can be visualized as placing a tube along the path points and gradually inflating it until it encounters an obstacle. This approach identifies the safe region around the path, ensuring obstacle avoidance while maintaining smooth navigation.

Constructing a suitable set of valid nodes (i.e., the graph) from the starting point to the goal depends on several parameters, including localization precision and variance, as well as desired performance metrics such as distance to obstacles and movement speed. In this method, a parameter called the precision factor (P) is introduced to represent the desired level of precision. This factor, defined in the range (0,1], influences key aspects of motion. A value of P closer to 1 indicates a lower precision requirement for motion, affecting the robot's speed and allowable proximity to obstacles.

To accommodate various mobile setups, two additional parameters are defined by the user:

1. **Robot Size (R):** The robot is modeled as a circle with radius R, simplifying obstacle avoidance calculations.
2. **Maximum Rotation:** This parameter specifies the robot's allowable angular movement during twisting motions.

The following section details the process of determining the optimal path between the starting and goal points, incorporating the precision factor and the other two parameters to achieve effective navigation.

3.1. Searching Process

To implement the A* algorithm, a graph of potential points or nodes must first be constructed. A* is a best-first search algorithm, meaning it systematically explores the nodes of a tree graph, using a heuristic function and a cost function at each step to select the node with the lowest cost for expansion. Therefore, a graph of valid nodes must be provided. In this method, candidate child nodes are generated around a parent node by placing them on two concentric circles centered on the parent. The configuration of these circles, along with other relevant details, is shown in Fig. 2. To determine whether a child node is valid, two conditions must be satisfied: the first condition ensures that the child node does not collide with obstacles, and the second condition verifies that the node lies within the allowable movement range based on the robot's capabilities.

1- There is no cell with value +1 in a circle with the center of the candidate node and the radius of PR.

2- There is no cell with value +1 in a circle with the center of the midpoint and the radius of PR.

To implement the A* algorithm, a graph of potential nodes must be created. A* explores nodes using a heuristic and cost function, selecting the lowest-cost node at each step. In this method, child nodes are generated around a parent node on two concentric circles, as shown in Fig. 2. For a child node to be valid, it must satisfy two conditions: it must not collide with obstacles, and it must lie within the robot's allowable movement range. The second condition involves a midpoint, which is the point halfway between the parent and child nodes. This is particularly important when navigating corners or narrow obstacles, especially if the distance between nodes is large, to avoid potential collisions.

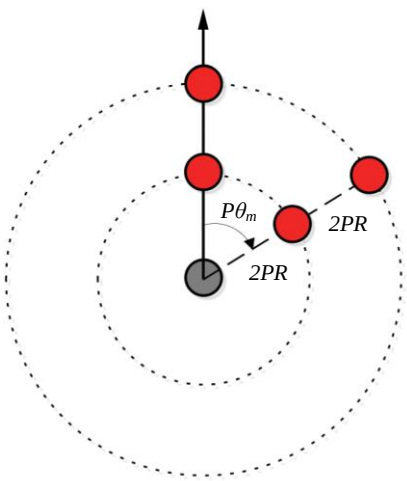

Figure 2. Expanding the graph from a parent node (black spot) to a child node (red spot)

In Fig. 2, the black vertical arrow indicates the robot orientation at the parent node. If the precision factor (P) decreases, the number of candidate nodes increases and the graph covers a vaster region.

The main determining agents in A* algorithm are the heuristic and cost functions. The cost function (Eq. 2) of this implementation of A* has four parts supposed to take into account three significant purposes. If $D(a,b)$ denotes the Euclidean distance,

and $R(a,b)$ denotes the orientation difference between two positions a and b, then the cost function for any node child node (c) of the parent (p_c) is defined as follows:

$$f(x) = -K_d \frac{D(c,p_c)}{R} + K_r \frac{R(c,p_c)}{\theta_m} - K_o \frac{O(c)}{R} + f(p_c) \quad (2)$$

The first term prioritizes longer distances between nodes, while the second term seeks a path that minimizes the robot's rotation. Both of these terms aim to reduce pauses and, consequently, increase the robot's speed. As mentioned earlier, the robot alternates between twisting and forward motion at each step, so a smoother path enables faster progress toward the goal. The third term considers the proximity of the nearest obstacle at each candidate node, represented by the function *O(c)*, which indicates the distance from the robot to the closest object. The greater the distance between the robot and obstacles, the lower the cost. The final term accounts for the parent node's cost function, which is crucial for selecting the most appropriate node.

The heuristic function is designed to evaluate each candidate node in relation to the goal position. Given the known position of the goal (g), the distance from the robot's current position to the goal provides an appropriate criterion for the heuristic function. Equation 3 represents the heuristic function used to calculate the global cost.

$$h(c) = K_h \frac{D(c,g) - D(p_c,g)}{R} \quad (3)$$

All the terms are normalized to robot size R and θ_m to make them comparable. In these functions, all gains are positive and similar to other objective functions the ratio between the coefficients determines the performance of the algorithm. The general cost function is the summation of two functions *h(c)* and *f(c)*:

$$g(c) = f(c) + h(c) \quad (4)$$

The final step is to define a condition for identifying the goal node. Since it is unlikely to find a node that precisely matches the goal position, an acceptable region around the goal is established. Any node within this region is considered a valid goal node. Specifically, if the distance between a node and the goal point is less than a predefined threshold (PR), the node is deemed valid.

The implementation of the A* algorithm follows the standard approach. Briefly, there are two groups of nodes: the open set and the closed set. At each step, the node with the lowest overall cost function in the open set is selected for expansion, and then it is moved to the closed set. Once a node satisfies the goal condition, the search terminates, and the parent nodes are used to construct the final path.

3.2. Path Planning Algorithm

In order to find the best path between each two points, a path planning algorithm is suggested that tries different precision factors and then chooses the best one. Then all path points are inflated to improve the robot performance. *P* determines the robot behavior while approaching each path node. The width of the safe tunnel and the robot speed are determined by the corresponding precision factor. The Fig. 3 demonstrates the first part. If N_i denotes the set of nodes belonging to i_{th} path, the Eq. 5 is utilized to compare the paths.

$$S(N_i) = \sum_{n \in N_i} D(n, p_n) \quad (5)$$

This function calculates the sum of distances between consecutive path points. The optimal path minimizes the distance from the starting point to the goal while also avoiding known obstacles in the environment.

The robot's speed while traversing the planned path is influenced by the precision factor. When the robot moves close to an obstacle, especially with localization errors, it is necessary to reduce its speed for safer navigation. The user defines the maximum robot speed ($S_{\max}$), and at each step, the corresponding precision factor *P* is multiplied by this value to adjust the speed, slowing it down near obstacles.

Consequently, the closest obstacle to each node must be identified, and the robot's speed is adjusted accordingly. Specifically, when the path passes through a narrow doorway or hallway, the robot should slow down only in those areas, while maintaining a higher speed in other locations. Finally, the inflation process assigns the maximum possible P_n value to each node *n*. *C(n,P)* represents a circle centered at node *n* with a radius P_R . The inflation of path nodes is summarized in Table 1.

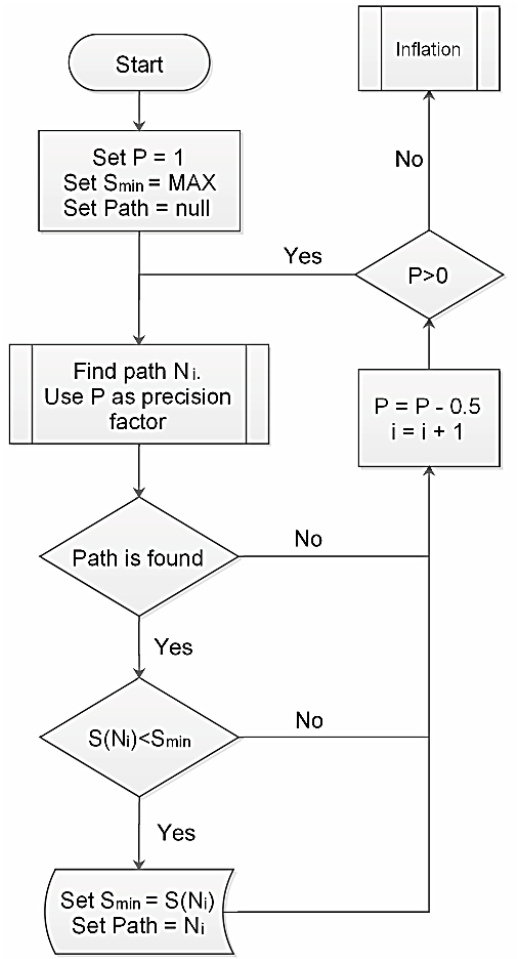

Figure 3. Flowchart of the first step of path planning

Table 1. Inflation of path nodes (Path) algorithm

No.	
1	*For all n ϵ Path*
2	*P=0.5*
3	*While (P<1) & (no obstalce in C(n,P))*
4	*P=P+0.5*
5	*End*
6	*P_n=P*
7	*End*

The value of 0.5 used to adjust the precision factor can be modified by the user for different applications. However, for a conventional robot size, this adjustment typically increases the search time without providing significant benefits. The final outcome of this step is a set of points and corresponding precision factors that define the robot's motion behavior. Nonetheless, the controller and obstacle avoidance methods remain crucial factors that must be addressed.

4. Controlling Method

To ensure the robot's safety while moving from node to node, the commands sent to the wheels should not be based solely on the instantaneous output of the localization system, as the resulting *(v, ω)* commands can fluctuate significantly due to measurement errors. Therefore, in our proposed method, the commands do not change with each measurement. Instead, they are updated only when the robot exceeds the safe region or begins the next step. To reach each path node, the robot first rotates to face the node and then starts moving toward it. Due to localization errors, it is possible that the robot will not reach the node after rotation and forward movement. To address this issue, a tunnel between each pair of consecutive nodes is defined as a safe region, and whenever the robot moves outside this region, a correction process is initiated.

If the robot position in the global coordinate system is represented by (x_r, y_r, θ_r) and goal node by (x_n, y_n) then the orientation error is calculated between the robot and node by following equations:

$$\theta_n = \begin{cases} tan^{-1}\left(\frac{y_n - y_r}{x_n - x_r}\right) & x_n \geq x_r, y_n \geq \\ 2\pi + tan^{-1}\left(\frac{y_n - y_r}{x_n - x_r}\right) & x_n \geq x_r, y_n < \\ \pi + tan^{-1}\left(\frac{y_n - y_r}{x_n - x_r}\right) & x_n < x_r \end{cases} \quad (6)$$

$$error_\theta = \begin{cases} \theta_n - \theta_r & -180 \leq \theta_r - \theta_n \leq 180 \\ 2\pi + \theta_n - \theta_r & \theta_r - \theta_n > 180 \\ 2\pi + \theta_r - \theta_n & \theta_r - \theta_n < -180 \end{cases} \quad (7)$$

The variables of these two functions are illustrated in Fig. 4. The robot orientation should be mapped into the range of $[0,2\pi]$. The positive sign of $error_\theta$ is the same as θ_n and θ_r. If the *error*، is positive, the robot twists to right and for negative value the robot twists to left for decreasing the error.

Robot twisting stops when the orientation error falls down a specified valid error. This value depends on the localizing system variance and is denoted by E_θ. Even if the user determines inappropriate parameters for a mobile robot or the system variance is too high, the following method is able to keep the robot in the safe region and steers it toward the goal but it takes much more

time to do the task. The following algorithm takes the next node position and corresponding P and steers the robot toward the node as presented in Table 2.

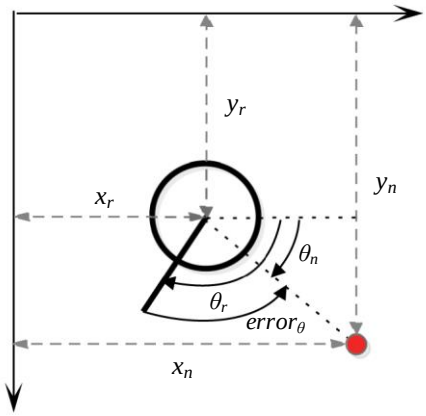

Figure 4. Robot twisting heading toward a node

Table 2. Going toward the node (*node*,*P*) algorithm

No.	
1	Pos_R = *localizing(t)*
2	CP_1 = ((Pos_R+ *node*)) / 2
3	CP_2 = ((Pos_R+ CP_1)) / 2
4	CP_3= (CP_1+Pos_R) / 2
5	*Twist (node,* E_θ*, PR)*
6	*While (* D (Pos_R, *node*) > 0.5PR)
7	*ForwardMove(PR)*
8	D_g = D (Pos_R, *node*)
9	D_1 = D (Pos_R, CP_1)
10	D_2 = D (Pos_R, CP_2)
11	D_3 = D (Pos_R, CP_3)
12	D_{min} = *Min* (D_g, D_1, D_2, D_3)
13	*If* (D_{min} > PR)
14	*Twist (node,* E_θ*, PR)*
15	*ForwardMove (PR)*
16	*For* $i \in \{1,2,3,\ldots,N_c\}$
17	*Count = 0*
18	*ranges = Sensor()*
19	*For all* $r \in$ *ranges*
20	*If* (D (Pos_R , r) < PR)
21	*count = count+1*
22	*If* (count > N_s)
23	*STOP*
24	*End*
25	*End*
26	Pos_R = *localizing(t)*
27	*End*

As shown in line 1, localizing(t) represents the output of the localization system. In lines 2-4, three checkpoints are created, and the distances from the robot's current position to each of these points, as well as the goal (lines 8-11), are compared in line 12 to determine if the robot is within the safe region. In Fig. 5, the three midpoints are marked by gray spots, while the red spot indicates the next node. Whenever the robot moves outside the black circles, it stops, and its heading is reset (lines 13-15). Lines 16-25 check the distances to nearby obstacles. The parameter N_c determines the number of checks per iteration. If the number of nearby obstacles exceeds N_s, the robot will stop and a new path will be planned. The robot's motion will also stop if it enters the red circle.

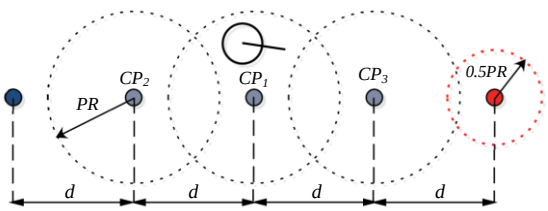

Figure 5. Demonstration of approaching the next node by defined checkpoints

This algorithm is executed for each consecutive path node until the robot reaches the goal. If the execution is halted due to obstacle detection, the map is updated, and the path planning algorithm recalculates a new path from the robot's current position to the goal node.

5. Simulation Test

The proposed method is evaluated in three simulated scenarios. A two-wheeled robot, modeled as a square with a side length of 50 cm, is used for two tests. A laser rangefinder, simulated as a sensor on the robot, detects a variable number of points (ranging from 120 to 140 points per revolution) to better simulate real-world conditions. The first test involves moving from a starting point to a destination without encountering unexpected obstacles, while the second test introduces a new obstacle along the path. These three environments are depicted in Fig. 6.

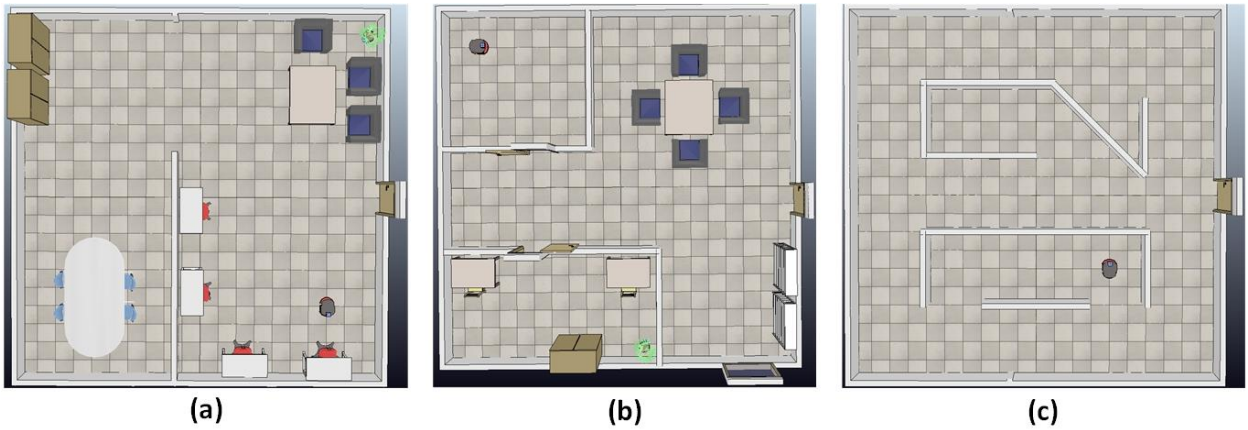

Figure 6. Three simulation environments. (a) Scene 1. (b) Scene 2. (c) Scene 3

The main advantage of the proposed method is dealing with significant localizing error. Therefore, the implemented localizing method for following tests is a particle filter. If $M_{(x,y)}$ denotes the cell value at the point (x, y) and S^i denotes the set of scanned points corresponding to candidate position (x^i, y^i, θ^i), then the weight of each particle is calculated as follows:

$$W(x^i, y^i, \theta^i) \triangleq \frac{|S^i|}{\sum_{(x,y,\theta)\in S^i} M_{x,y}} \qquad (8)$$

According to this weight function, the best position or the highest weight belongs to the particle which locates the most number of scanned points on the occupied regions of the map. This localizing method brings on significant error and variance in outputs and the path planning method should overcome this issue and lead the robot to the goal.

5.1. Path Planning Without New Obstacle

In the first test, two points are defined as the starting and goal positions, and the robot uses the proposed method to plan an optimal path. The control algorithm then guides the robot along the path until it reaches the goal. The parameters required for the subsequent tests are specified in Table 3.

Table 3. Values of different parameters for simulation test

Parameter	Value
R	*0.5 m*
θ_m	*45 deg*
K_d	*1*
K_r	*0.05*
K_o	*1*
K_h	*0.1*
S_{max}	*0.5 m/s*
N_s	*4*
N_c	*1*

In Fig. 7, the starting and goal points are shown. Also, the planned path, robot trajectory and localizing error are plotted.

5.2. The Existence of New Obstacle

With the parameters set the same as in the first test, a new obstacle is introduced along the robot's path. The robot then attempts to find a new path after encountering the obstacle. Fig. 8 illustrates the initial planned path and the updated path after accounting for the new obstacle.

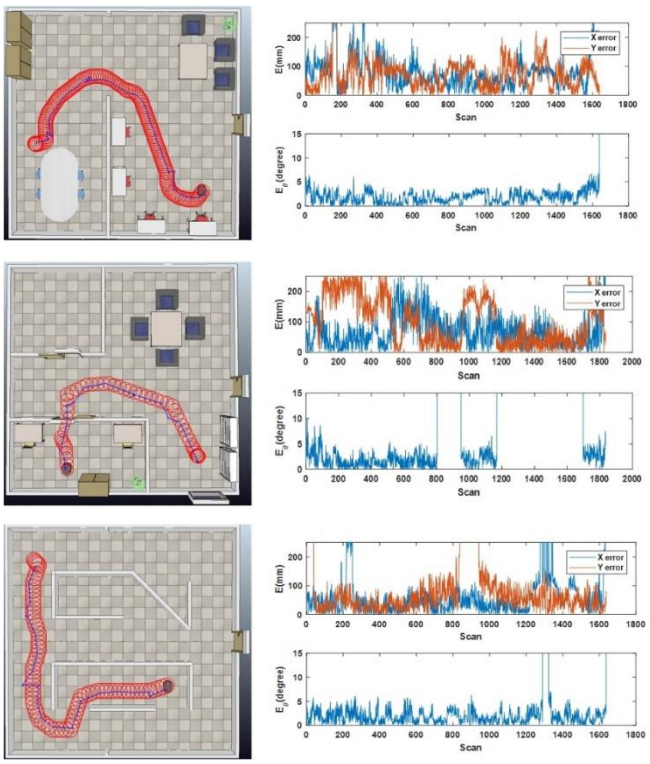

Figure 7. The results of the first test. Left column: robot trajectory (blue line) and allowed distance to obstacles at each path point and controller checkpoints (red circles). Right column: localizing error while moving along the path

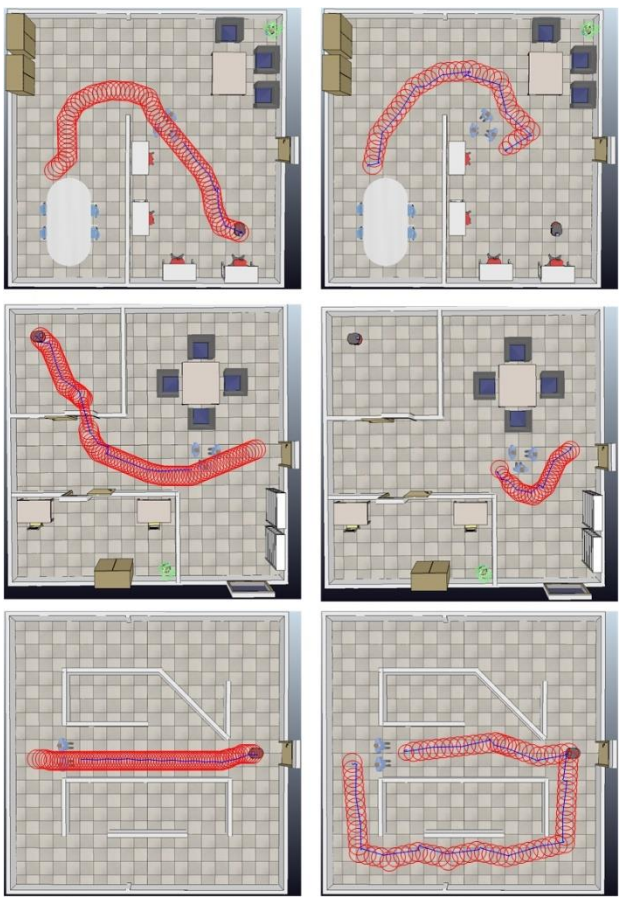

Figure 8. The results of the second test. Left column: primary planned path (red circles) and robot trajectory until reaching the new goal. Right column: the new planned path to the goal

6. Discussion on Results

In the first test, the robot's localization error is significant, as shown in Fig. 7. However, the proposed method successfully guides the robot to the goal, while other path planning methods fail to handle this error. Based on the planned paths in different scenarios, the robot's trajectory (represented by the blue line) is smooth (in comparison to the localization error) and primarily stays within the safe region (indicated by the red circles). Therefore, the A* algorithm, along with the proposed cost function and the defined parameters (Table 3), effectively identifies a short and safe path from the robot's current position to the goal.

7. Conclusion

All mobile robots require a reliable path planning method for autonomous applications. Most conventional path planning solutions are unable to handle significant localization and sensor errors, which can lead to oscillations during robot movement. In this paper, a novel path planning approach is presented that safely guides the robot to its goal in a known environment. This approach is based on the A* search algorithm and restricts the robot's motion to two types: forward motion and rotation. The A* algorithm is applied to a transformed occupancy grid map. According to simulation results, even with inaccurate localization output, the robot successfully navigates toward the goal while avoiding obstacles. The planned path, determined by the defined cost function, represents the shortest safe route from the starting position to the goal, and the proposed control algorithm ensures the robot stays on this path.

References

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," pp. 500–505, 1985.

[2] J. Borenstein and Y. Koren, "The Vector Field Histogram - Fast obstacle avoidance for mobile robots," *IEEE Journal of Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[3] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," no. March, pp. 1–23, 1997.

[4] V. Ganapathy, C. Y. Soh, and J. Ng, "Fuzzy and neural controllers for acute obstacle avoidance in mobile robot navigation," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pp. 1236–1241, 2009.

[5] K. Samsudin, F. A. Ahmad, and S. Mashohor, "A highly interpretable fuzzy rule base using ordinal structure for obstacle avoidance of mobile robot," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1631–1637, 2011.

[6] S. Dutta, "Obstacle Avoidance of mobile robot using PSO based Neuro Fuzzy Technique," vol. 02, no. 02, pp. 301–304, 2010.

[7] O. R. E. Motlagh, T. S. Hong, and N. Ismail, "Development of a new minimum avoidance system for a behavior-based mobile robot," *Fuzzy Sets and Systems*, vol. 160, no. 13, pp. 1929–1946, 2009.

[8] X. Li and B. Choi, "Design of Obstacle Avoidance System for Mobile Robot using Fuzzy Logic Systems," *International Journal of Smart Home*, vol. 7, no. 3, pp. 321–328, 2013.

[9] M. Faisal, R. Hedjar, M. Al Sulaiman, and K. Al-Mutib, "Fuzzy logic navigation and obstacle avoidance by a mobile robot in an unknown dynamic environment," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[10] S. Wen, W. Zheng, J. Zhu, X. Li, and S. Chen, "Elman fuzzy adaptive control for obstacle avoidance of mobile robots using hybrid force/position incorporation," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 4, pp. 603–608, 2012.

[11] C.-L. Hwang and L.-J. Chang, "Trajectory Tracking and Obstacle Avoidance of Car-Like Mobile Robots in an Intelligent Space Using Mixed H_2/H_infinity Decentralized Control," *IEEE/ASME Transactions on Mechatronics*, vol. 12, no. 3, pp. 345–352, 2007.

[12] A. Sgorbissa and R. Zaccaria, "Planning and obstacle avoidance in mobile robotics," *Robotics and Autonomous Systems*, vol. 60, no. 4, pp. 628–638, 2012.

[13] L. Tang, S. Dian, G. Gu, K. Zhou, S. Wang, and X. Feng, "A Novel Potential Field Method for Obstacle Avoidance and Path Planning of Mobile Robot -," *IEEE International Conference on Computer Science and Information Technology*, pp. 633–637, 2010.

[14] E. Shi, T. Cai, C. He, and J. Guo, "Study of

the new method for improving artifical potential field in mobile robot obstacle avoidance," *Automation and Logistics, 2007 IEEE International Conference on*, no. 50375119, pp. 282–286, 2007.

[15] C. Shi, Y. Wang, and J. Yang, "A local obstacle avoidance method for mobile robots in partially known environment," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 425–434, 2010.

[16] E. J. Jung, J. H. Lee, B. J. Yi, J. Park, S. Yuta, and S. T. Noh, "Development of a laser-range-finder-based human tracking and control algorithm for a marathoner service robot," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 6, pp. 1963–1975, 2014.

[17] C. Stachniss and W. Burgard, "An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments," *IEEE International Conference on Intelligent Robots and Systems*, vol. 1, no. October, pp. 508–513, 2002.

[18] I. Ulrich and J. Borenstein, "VFH+: reliable obstacle avoidance for fast mobile robots," *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 2, no. May, pp. 1572–1577, 1998.

[19] I. Ulrich and J. Borenstein, "{VFH*: local obstacle avoidance with look-aheadverification}," *Proceedings of the 2000 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, no. April, pp. 2505–2511, 2000.

[20] A. Babinec, F. Duchoň, M. Dekan, P. Pásztó, and M. Kelemen, "V FH*TDT (V FH*with Time Dependent Tree): A new laser rangefinder based obstacle avoidance method designed for environment with non-static obstacles," *Robotics and Autonomous Systems*, vol. 62, no. 8, pp. 1098–1115, 2014.

[21] C. Ye, "Navigating a mobile robot by a traversability field histogram," *IEEE* Transactions *on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 2, pp. 361–372, 2007.

[22] M. Seder and I. Petrović, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," *IEEE International Conference on Robotics and Automation*, no. April, pp. 1986–1991, 2007.

[23] K. O. Arras, J. Persson, N. Tomatis, and R. Siegwart, "Real-time obstacle avoidance for polygonal robots with a reduced dynamic window," *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 3, no. May, pp. 3050–3055, 2002.

[24] M. Khorram and S. A. A. Moosavian, "Dynamics modeling and stable gait planning of a quadruped robot in walking over uneven terrains," vol. 46, no. 2, pp. 205–220, 2015.

[25] M. Sadedel, A. Yousefi-koma, and F. Iranmanesh, "Analytical Dynamic Modelling of Heel-off and Toe-off Motions for a 2D Humanoid Robot," vol. 46, no. 2, pp. 243–256, 2015.

[26] M. R. Lavaei, M. Mahjoob, and A. Behjat, "Stiffness control of a legged robot equipped with a serial manipulator in stance phase," *Jcamech*, vol. 48, no. 1, pp. 27–38, 2017.

[27] D. Kriesel, *A Brief Introduction to Neural Networks*. available at http://www.dkriesel.com, 2007.

Biography

Seyed Foad Hosseini Musa is a graduate student of Mechatronics at *University of Tehran*. He has also M.Sc. graduated from The *École Nationale Supérieure d'Arts et Métiers* in the field of knowledge integration of mechanical production. His areas of interest include robotics, mechatronics and mobile robots.

Seyyed Saied Kazemi received his B.Sc. and M.Sc. degrees in Mechanical Engineering from *University of Tehran*, Tehran, Iran in 2011 and 2015 respectively. He was the member of Intelligence Based Experimental Mechanics (IBEM) since 2015. His research interests include structral design of structres like robots, fatigue and damage behavoiur of materials.