Proceedings of the 10th RSI International Conference on Robotics and Mechatronics (ICRoM 2022),
Nov. 15-18, 2022, Tehran, Iran

2022 10th RSI International Conference on Robotics and Mechatronics (ICRoM) | 978-1-6654-5452-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRoM57054.2022.10025051

Dynamic reconfiguration of costmap parameters with Fuzzy controllers to enhance path planning

Navid Zarrabi
Amirkabir University of Technology
Tehran, Iran
navidz@aut.ac.ir

Rasul Fesharakifard*
Amirkabir University of Technology
Tehran, Iran
fesharaki@aut.ac.ir

Mohammad Bagher Menhaj
Amirkabir University of Technology
Tehran, Iran
menhaj@aut.ac.ir

***Abstract*—Autonomous mobile robots are widely used in industries, and these robots utilize various methods to find their path in complex environments. Path planning methods aim to choose the shortest and fastest route to the target. However, working alongside humans needs a better intuition of the surrounding environment. In this paper, Fuzzy controllers have been designed to reconfigure the costmap parameters of mobile robots, considering the danger of collision in their environment. The Fuzzy controller has been tuned by tracking the behavior of mobile robots in several experiments, and it is observed that conveying the result of these observations to robots would significantly improve their navigation behavior. The results of this work are applicable to all path planning methods working based on costmaps.**

***Index Terms*—Fuzzy controller, Navigation, Path planning, Costmap, Autonomous mobile robots**

I. Introduction

Mobile robot applications are rapidly growing all around the world. Meanwhile, robots are getting smarter every day and reaching a better understanding of their surrounding environment. Using costmaps to model the environment is very common. Sensors perceive the data from the robot's world and build a 2-dimensional or 3-dimensional occupancy grid of the data, and the costs are inflated in 2D costmaps based on the occupancy grids and a user-specified inflation radius [1]. Each cell in a costmap is free, occupied, or unknown, except for the inflation layer.

Graph-based algorithms are frequently used for finding the shortest path in costmaps. Numerous graph search algorithms developed over the last decades have been tested for path planning of autonomous robots, for instance, Astar (A*), Dijkstra, breadth-first search (BFS), and depth-first search (DFS) [2]. These algorithms can find a path between two locations on a static global costmap. However, the output is merely an array of points, and local planners are needed to generate velocity commands for motors.

Local planners are used for tracking global paths generated by global planners and have to consider some constraints. For instance, the famous Dynamic Window Approach (DWA) introduced in [3], generates a space of feasible velocities and controls the mobile robot successfully. This algorithm will never generate a speed that is impossible to achieve from the current state. Time Elastic Band (TEB) local planner is another example of practical local planners [4]. Each of these algorithms has its own setting and parameters.

In [5] a Fuzzy controller has been deployed to dynamically adjust the weights of the terms in the Dynamic Window cost function. The main goal of fuzzy calculations, especially fuzzy controllers, is modeling an expert to enhance the automation of a complicated system [6]. It is not convenient to model an expert's knowledge using classic controllers, and knowledge-based controllers are a better fit for this purpose.

There are two main groups of knowledge-based controllers. The first group has a supervisory role and tries to improve the performance of a closed-loop classical controller. The second knowledge-based controllers work independently and substitute the classic controllers. The proposed Fuzzy controller aims to enhance the performance of a mobile robot and thus is of the first type.

In fact, many papers have presented a planner improved by Fuzzy controllers. In [7], Fuzzy logic is adopted to evaluate the danger of moving obstacles by estimating risk and relative distance. To adapt Dynamic Window Approach in

*Corresponding author.
E-mail: fesharaki@aut.ac.ir; Tel.: +982164545579;
Fax: +982164542670;

more complex environment, reference [8] has introduced a modified algorithm in which its coefficients are adapted by Fuzzy controllers.

Although there always exists a planning algorithm or a combination of algorithms that can navigate a mobile robot from a starting point to the target, there are still many parameters to re-tune when the robot is tested in a new environment. Even in a single complex environment, it would be difficult to reach the correct settings. A detailed guide for tuning the navigation parameters in ROS is presented in [9]. Some papers suggest learning these parameters using teleoperated demonstration, corrective interventions, evaluative feedback, or reinforcement learning in simulation [10]. Nevertheless, this process would need a considerable amount of time and effort.

In our previous work, we analyzed localization performance using different sensors on a ROS-based transport mobile robot [11]. Testing robots in real-world scenarios have shown that the navigation parameters should be tuned according to the local settings of the environment. In [12], a room segmentation approach has been introduced, which can be inspiring to make the robot re-calibrate its parameters automatically using a knowledge-based method. Based on our experiments about mobile robots and their performance in real-world scenarios, Fuzzy controllers have been designed to re-tune the costmap parameters wherever needed.

II. Test platform

To test and verify the proposed method, Gazebo Simulator [13] is used with the open source turtlebot3 chosen as a differential drive test platform. A large simulation environment, called the house (Figure 1), is chosen as the test environment since it offers a house with rooms and challenging path planning scenarios for a mobile robot. The Robot Operating system is linked to Gazebo to implement the codes in Python and C++.

The software platform is based on ROS Noetic in Ubuntu 20.04. The map of the test zone, shown in Figure 2, is known by the robot as prior knowledge and is called a static map. The simulated robot uses Odometry data, and the well-known Adaptive Monte Carlo Localization (AMCL) is used to reduce the accumulated error. This package uses a particle filter to track a robot's pose against the static map.

The proposed method is applicable to any path planning algorithm that uses two-dimensional costmaps. There are several choices for Global path planning algorithms, e.g., Dijkstra and A* modified versions, and several choices for local planners, e.g., Dynamic-Window Approach (DWA) [3] and Time Elastic Band (TEB). However, the following settings have been used in all tests discussed in this paper.

In our experiments, modified A* is used to find the static map's shortest path. The robot applies the Dynamic Window Approach (DWA) as a local planner to stick to the global path. DWA minimizes the cost function G in Equation (1). $Heading$ is the alignment of the robot with the target direction, $Dist$ is the distance to the closest obstacle that intersects with the curvature, and $Velocity$ is the robot's progress on the corresponding trajectory.

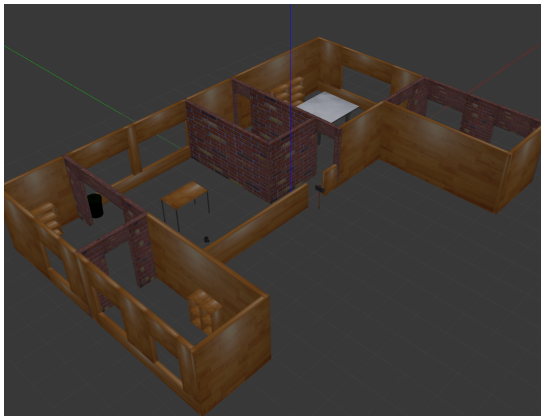


Fig. 1. Test simulation environment

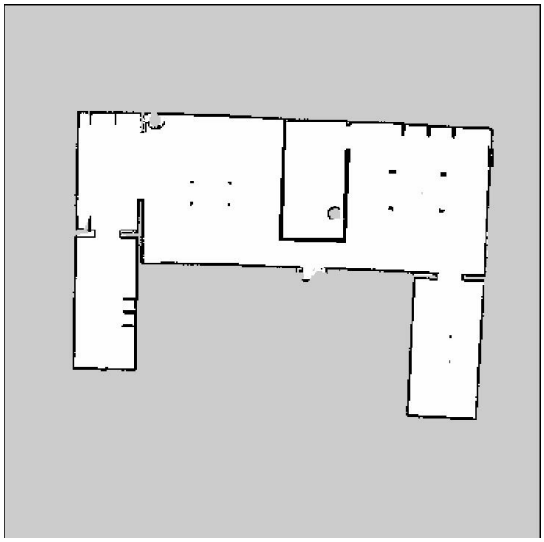


Fig. 2. Static(Global) map of the environment

$$G(v,\omega) = \sigma(\alpha\,\mathrm{heading}(v,\omega)+\beta\cdot\mathrm{dist}(v,\omega)+\gamma\cdot\mathrm{velocity}(v,\omega)) \quad (1)$$

The α, β, and γ are cost function variables and σ smoothens the cost function. The presented combination of packages have been tested on multiple systems and are a reliable software base for mobile robots.

III. Problem Statement

A 2-d laser scanner is sometimes sufficient to avoid obstacles. For instance, uniform cross-section obstacles are detected and avoided accurately using 2-d Laser scanners. On the other hand, 3-d sensors, like lidars and depth cameras, are expensive and computationally costly. This is why 2-d lidars are still popular among autonomous robots and cars.

One solution to overcome the risk of collision is inflating obstacles in the costmap. This might provide a safe margin by paying attention to robot dimensions, obstacle dimensions, and

a safe margin. Nevertheless, different situations require varied settings. In our experiments on mobile robots in real-world environments, we have noticed that large margin troubles robot in narrow environments while small margins make the robot less cautious when it faces an obstacle with a non-uniform cross-section, like an office chair.

Let us define Inflation radius as the radius in meters to which the map inflates obstacle cost values. It is common to mask an inflation layer on the whole costmap, each cell ranging from 0 to 255. A scaling factor is used to create an exponential decay curve as follows.

$$v = \exp(-1.0 * csf * (d_o - r_i)) * c) \tag{2}$$

where v is the value for each cell, the csf variable is the cost scaling factor, d_o is the distance from the obstacle, and c is assumed to be a constant coefficient. The negative sign in Equation (2) increases the v value as the robot approaches the obstacle. The r_i parameter is the robot's inscribed radius, which will be explained in what follows.

Therefore, inflation is propagating cost values out from occupied cells that decrease with distance. Obstacle inflation cells have five different cell types [1]: Lethal, Inscribed, Possibly circumscribed, Freespace, and Unknown.

The Lethal regions are occupied cells, and a collision will happen if the robot enters the Lethal zone. The inscribed radius is the radius of the largest circle that can be contained inside the robot's cross-section polygon. Similarly, the circumscribed radius is the radius of the largest circle that passes all (or most) of the robot's cross-section polygon vertices.

When an occupied cell is in an inscribed region, collision is certain. On the contrary, when the occupied cell is between inscribed and circumscribed zones, collision depends on the robot's orientation, and the zone is called possibly circumscribed. Finally, the region between the circumscribed and inflation radius is set according to user preference. The steepness of the decay curve of Equation (2) depends on the *csf* in this region. In [9], some suggestions have been made about setting proper values to *csf* and inflation radius. The following lines show the challenges of setting static values to the cost scaling factor (csf).

Figure 3 shows different paths planned by same algorithms but different cost scaling factors. The global algorithm only cares about minimizing distance and reaching the target. However, passing from cluttered and narrow spaces is undesirable in many real-world case. In this specific case, the red path is shorter but planned to pass beneath a table in the room, which is not preferable in an office. In fact, a robot with high scaling factor might choose to pass from anywhere larger than its curcuscribed radius.

Figure 4 shows the effect of *csf* on robot turns. The red path with larger *csf* is shorter and faster, but in many cases, it is not safe in uncertain environments. On the other hand, choosing smaller *csf* might confuse robots in corridors and cause more unnecessary maneuvers and, thus, more traversed distance, energy consumption, and time.

Hence, changing the *csf* coefficient dynamically will help the robot to adapt its behavior considering its surrounding environment. Robots usually use a global (static) map and a local map of the environment simultaneously. Cost scaling factor and inflation are defined separately for each of these global and local maps. Since it is assumed that the robot has a static map of the environment, it is possible to let the robot know about some critical points on its map.

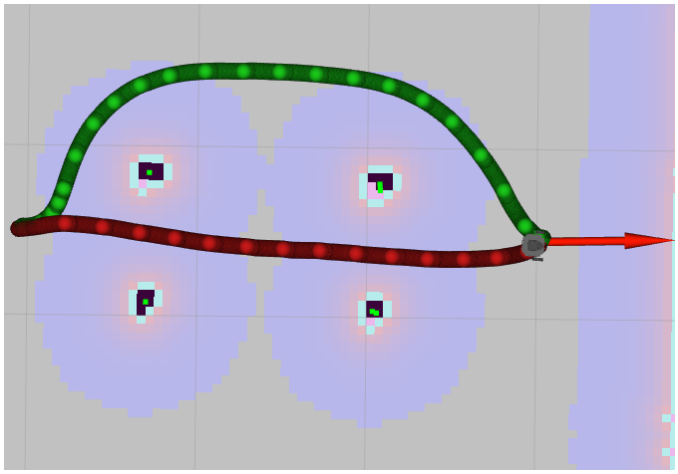


Fig. 3. Red path shows the csf=10 and green path shows csf=3

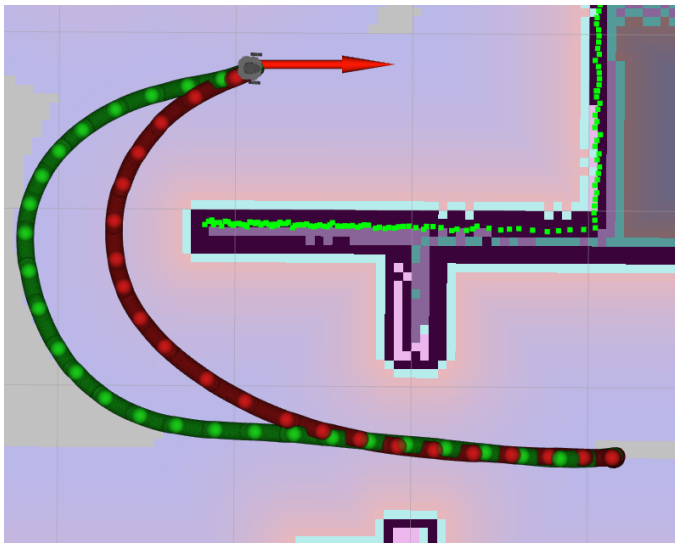


Fig. 4. Red path shows the csf=5 and green path shows csf=2

IV. Fuzzy controller nodes

The general idea is to put some points as landmarks on the static map, which makes the robot aware of its global state. So, the robot knows how far its current position is from a room full of office chairs or a safe corridor with no challenging obstacles to avoid. The first input of the Fuzzy controller is the Euclidean distance of the robot from these landmarks. The second input is the first derivative of the distance from landmarks. This input will help the robot be aware of the risk of collision since being

near an obstacle while it is getting closer is more dangerous than getting farther.

The Fuzzy controller's output is the *csf* mentioned in Equation (2). The inflation radius has been set to 1 meter, and in our tested environment, it works appropriately. However, one might choose to update the inflation radius dynamically. As mentioned earlier, *csf* depends on the inflation radius, and the range of *csf* should be defined according to the inflation radius. After several experiments, the proper *csf* range for this scenario is estimated between 0 to 10 for inflation.

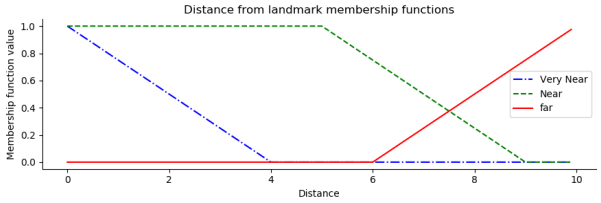
To address the problems stated in section III, Fuzzy nodes are added to the ROS environment. For each landmark point, a Fuzzy node should be defined. Although it is possible to use the same node for similar segments(regions), the rules and membership functions must be defined by an expert for each case. First, the problem explained in Figure 3 is discussed, and a Fuzzy node is added to prevent the robot from passing below tables and narrow pathways.

Figure 5 demonstrates our suggestion for input and output membership functions. Distance is assumed to be between 0 to 10, and the code is written to neglect landmarks with distances more than 10 meters. Three membership functions are defined for the distance in Figure 5(a). The robot is prone to collision when the "Very near" membership function is activated. On the other hand, the more the distance, the less the risk of collision will be. As the distance to the landmark gets smaller and it enters the "Very near" and then "Near" membership functions, the cost scaling factor increases gradually. For the second input, only one membership function is defined, activated when the robot is getting far from the landmarks (Figure 5(b)).

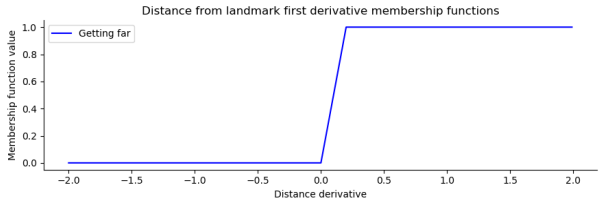
The "Getting far" membership function is defined with a steep slope to have sufficient influence on the output and is not defined as a step function to avoid fluctuations near the target point. According to Figure 5(c), the output has three membership functions. The default value for *csf* is 3, and the membership functions of output are defined such that this value remains constant until robot distance gets close to the "Near" region.

Fuzzy rules are defined as shown in table I. Rule one attributes the "Near" membership function to a "Low" value for *csf*, regardless of the robot's orientations. The reason is that when the robot is "Very Near" to the obstacle, a slight movement of the robot might culminate in negating the distance derivative. In addition, when the robot is near the obstacle and getting far, a collision is unlikely (Rule 2). On the other hand, if the robot is getting close to an obstacle, the Fuzzy controller should do its best to repel the planned path from the crowded region (Rule 3). If the distance is "Far", the robot can use the default value of 3 regardless of the derivative.

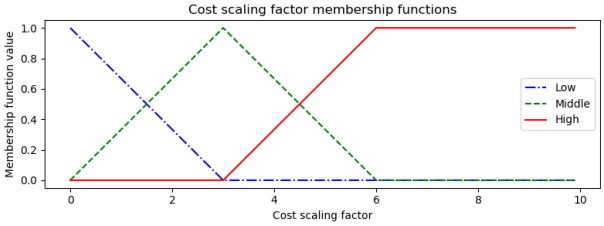
The Fuzzy *And* method is set to "minimum" in our Fuzzy system. Mamdani minimum inference engine (Equation (3)) has been used, which is an individual-rule-based approach. The Mamdani implication is also deployed. This inference engine is computationally simple and intuitively appealing



(a) Input membership functions for distance



(b) Input membership functions for distance derivative



(c) Output membership functions

Fig. 5. Fuzzy input and output membership functions

for many practical problems [6].

$$\mu_{B'}(y) = \max_{I=1}^{M} \left[\sup_{x \in U} \min \left(\mu_{A'}(x), \mu_{A_1^I}(x_i), \ldots, \mu_{A_n^I}(x_i), \mu_{B^I}(y) \right) \right] \tag{3}$$

where $A' \in U$ is the input Fuzzy set, $B' \in V$ is the output Fuzzy set, μ shows membership functions, and M is the number of rules. Figure 6 shows the cost scaling factor for different input values.

As mentioned earlier, with the controller explained above, the planner is more likely to choose a safe path and avoid risky regions. All the code has been implemented in a node in the ROS environment, and by putting purple sphere Markers on the static map (Figure 7(b)), the user can choose the areas that the robot should avoid.

The problem illustrated in Figure 4 can be solved with minor modifications in the aforementioned Fuzzy node. Table II shows the slight changes in Fuzzy rules. Should the expert prefer sharp turns, a shorter path, and a smaller margin from the walls, this modified node could be utilized at any desired location. In the following experiments, this modified Fuzzy node is shown with a blue sphere on the static map (Figure 7(b)).

V. Experiment Results

Figure 7 compares the results of experiments in a more challenging scenario where the robot is faced with both situations discussed in this paper. The test starts from the top right corner of Figures 7(a) and 7(b), and ends in the room

TABLE I
FUZZY RULES FOR CONTROLLER NODE

Rule number	*If statement*	*then statement*
1	(Distance is Very Near)	(csf is Low)
2	(Distance is Near) and (Ditance derivative is Getting Far)	(csf is Middle)
3	(Distance is Near)	(csf is Low)
4	(Distance is Far)	(csf is Middle)

TABLE II
FUZZY RULES FOR MODIFIED CONTROLLER NODE

Rule number	*If statement*	*then statement*
1	(Distance is Very Near)	(csf is High)
2	(Distance is Near) and (Ditance derivative is Getting Far)	(csf is Middle)
3	(Distance is Near)	(csf is High)
4	(Distance is Far)	(csf is Middle)

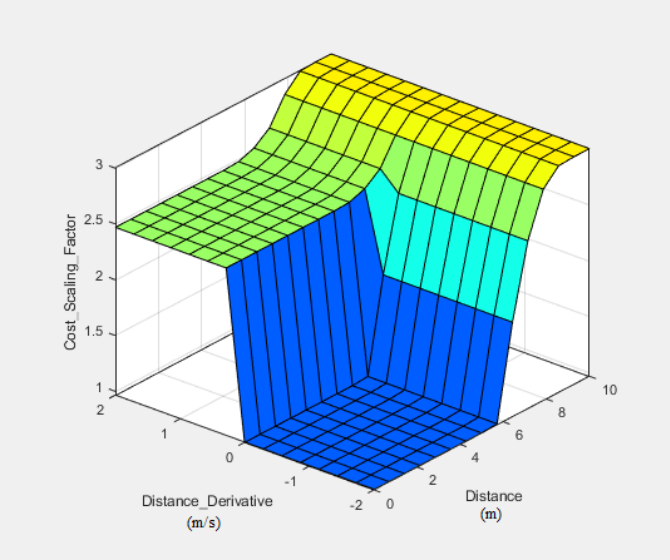


Fig. 6. Cost scaling factor diagram of Fuzzy rules

on the left bottom corner of these figures. As the robot starts navigation, it will face a table in the first room, a door to the second room, a table in the second room, and a door to the third room, respectively (Figure 1).

The default value (*csf*=3) is used in the first test, and the robot navigates successfully to reach the target. Still, it chooses to pass through the table in the second room. The reason is that table legs are tiny, and the robot observes them as separate small obstacles which are easy to avoid. The first Fuzzy controller introduced in section IV and the purple sphere in Figure 7(b) is added for this purpose. Therefore, our Fuzzy node will make the robot aware that it should pass more expansive areas, if possible.

In the second test, the *csf* value is decreased to 1, and the robot is expected to avoid cluttered regions. The green path in Figure 7(a) shows that the robot chooses a dangerous path through the table in the first room since it tends to move with abrupt rotations. In the second room, the robot chooses to avoid the table as it perceives the decayed costmap curve as soon as entering the room. However, while entering the third room, the robot gets troubled by its sudden turns and moves back and forth a few times to find the path to the last room. The dense green circles by the door in Figure 7(a) illustrate the shortcomings of choosing a small constant value for *csf*. Hence, the blue circle Marker in Figure 7(b) is added in the third experiment to make the robot aware that sharp turns are probable while entering the third room.

Two Fuzzy nodes are added in the third experiment, as shown in 7(b). The grey path shows similar behavior to the red path in the first room, where the Fuzzy nodes are not updating *csf*, and the default value is maintained. While entering the second room, the first Fuzzy controller is activated and deviates the path curve by decreasing the value of *csf*. Finally, when the robot gets closer to the blue Marker, it increases the *csf* again, causing a safe turn while entering the third room.

One important notice is that these controllers should not conflict with each other. In other words, updating the *csf* value from two distinct Fuzzy nodes confuses the robot and results in several path replanning in overlapping regions. The membership functions in Figure 5 are defined in a 0 to 10-meter range, but considering the marker regions, these membership functions could be scaled conveniently, and there is no need to change any other settings. In our experiments, the purple and blue nodes are scaled by a factor of 0.3 and 0.2, respectively. In other words, the effective regions for these nodes are 3 meters and 2 meters, and outside these regions, the *csf* values are reset to default. The traversed distance and required time to reach the goal have been recorded and summarized in Table III. The third experiment reports the shortest path, although the robot is forced to avoid the table. The first experiment is the fastest since a sharp turn is not

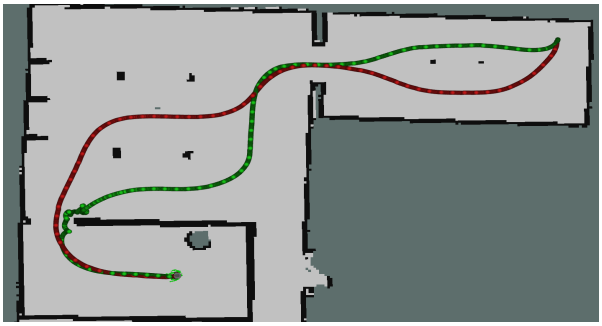
TABLE III
COMPARING RESULTS OF EXPERIMENTS BY MEASURING TIME AND DISTANCE

Test Number	*Test settings*	*Traversed distance (m)*	*Time (s)*
1	Inflation= 1 (m) , Cost Scaling Factor=3	15.072	75.097
2	Inflation= 1 (m) , Cost Scaling Factor=1	16.039	99.535
3	Fuzzy nodes used	15.040	75.615

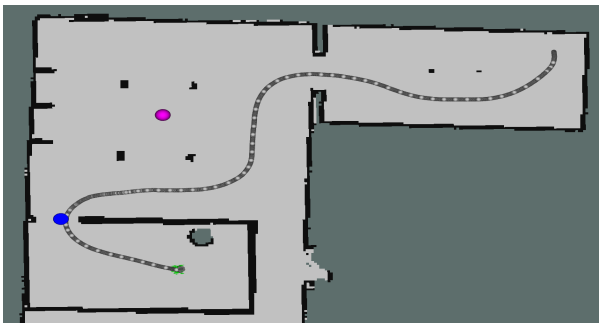
necessary to reach the third room. However, experiment 3 still reports an adequate time to reach the target. The distance and time numbers are the worst of all in the second experiment since the robot took some time to find a proper path to room 3. As a result, the explained Fuzzy nodes enhance the time of navigation, reduce the risk of collision, plan the robot's path from desired locations, and culminate in a reasonable path to the goal.

VI. CONCLUSION

In this paper, two Fuzzy controllers are introduced to enhance mobile robot safety and navigation behavior in real-world environments. Fuzzy controllers are implemented as two ROS nodes that can be added to any desired points on the environment's static map based on an expert's knowledge. Two challenges with different constant cost scaling factors are discussed, and the Fuzzy nodes are added to overcome these challenges. The results demonstrate that the modified path passes from the regions with a lower risk of collision. In addition, the Fuzzy controllers prevent the robot from getting stuck in narrow spaces and room entrances. At last, the time and distance of navigation prove the effectiveness of this approach.



(a) Planned path with static csf: Red for csf=3 and green for csf=1



(b) Generated path while using Fuzzy nodes

Fig. 7. Comparing constant csf and Fuzzy dynamic reconfiguration of csf

REFERENCES

[1] "ROS: an open-source Robot Operating System.", 27 October 2020, Online: https://ros.org/.

[2] "Koubaa, A. et al. (2018). Introduction to Mobile Robot Path Planning." Robot Path Planning and Cooperation. Studies in Computational Intelligence, vol 772. Springer, Cham. https://doi.org/10.1007/978-3-319-77042-0_1

[3] Fox, Dieter, Wolfram Burgard, and Sebastian Thrun. "The dynamic window approach to collision avoidance." IEEE Robotics & Automation Magazine 4, no. 1 (1997): 23-33.

[4] Rösmann, Christoph, Frank Hoffmann, and Torsten Bertram. "Integrated online trajectory planning and optimization in distinctive topologies." Robotics and Autonomous Systems 88 (2017): 142-153.

[5] Akka, Khaled, and Farid Khaber. "Optimal tracking control of a trajectory planned via Fuzzy reactive approach for an autonomous mobile robot." International journal of advanced robotic systems 15, no. 1 (2018): 1729881418760624.

[6] Mohammad Bagher Menhaj, Yaser Shokri Kalandaragh. *" Fuzzy Control"* Tehran, Iran: Amirkabir University of Technology, 2015.

[7] Lin, Zenan, Ming Yue, Guangyi Chen, and Jianzhong Sun. "Path planning of mobile robot with PSO-based APF and Fuzzy-based DWA subject to moving obstacles." Transactions of the Institute of Measurement and Control 44, no. 1 (2022): 121-132.

[8] L. Xiang, X. Li, H. Liu and P. Li, "Parameter Fuzzy Self-Adaptive Dynamic Window Approach for Local Path Planning of Wheeled Robot," in IEEE Open Journal of Intelligent Transportation Systems, vol. 3, pp. 1-6, 2022, doi: 10.1109/OJITS.2021.3137931.

[9] Zheng, Kaiyu. "Ros navigation tuning guide." arXiv preprint arXiv:1706.09068 (2017).

[10] Xiao, Xuesu, Zizhao Wang, Zifan Xu, Bo Liu, Garrett Warnell, Gauraang Dhamankar, Anirudh Nair, and Peter Stone. "Appl: Adaptive planner parameter learning." Robotics and Autonomous Systems 154 (2022): 104132.

[11] N. Zarrabi, R. Fesharakifard and M. B. Menhaj, "Robot localization performance using different SLAM approaches in a homogeneous indoor environment," 2019 7th International Conference on Robotics and Mechatronics (ICRoM), Tehran, Iran, 2019, pp. 338-344, doi: 10.1109/ICRoM48714.2019.9071902.

[12] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele. Room Segmentation: Survey, Implementation, and Analysis. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2016.

[13] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), 2004, pp. 2149-2154 vol.3, doi: 10.1109/IROS.2004.1389727.