Fuzzy Motion Planning for Nonholonomic Mobile Robot Navigation in Unknown Indoor Environments

Weria Khaksar, Md Zia Uddin, and Jim Torresen
Robotics and Intelligent Systems Group,
Department of Informatics, University of Oslo, Oslo, Norway
Email: {weriak, mdzu, jimtoer}@ifi.uio.no

***Abstract*—An important expectation from a mobile robot is to be able to navigate in unexplored areas in the presence of unknown obstacles. Despite the great deal of work in this field, this problem is still a challenge as the planning domain is highly complex. This challenge can be seen in the form of high-cost solutions, high computational cost or increasing rate of failure. In this paper, a fuzzy motion planning approach is presented for guiding a nonholonomic mobile robot in unknown environments. The fuzzy controller uses the readings of the robot's sensor(s) in terms of total travelled distance, distance to the goal and the presence of obstacles in the vicinity of the robot; and computes the linear and angular velocity of the robot with a pre-defined frequency. The proposed method was tested through simulation as well as experimental studies on a TurtleBot in indoor environments. The simulation and experimental studies show the effective performance of the proposed approach in terms of solution cost, computational cost and failure rate.**

***Index Terms*—motion planning, nonholonomic systems, fuzzy logic, unknown environments**

I. Introduction

Path planning for a mobile robot is the procedure of moving the robot from an initial position to a goal configuration inside an environment filled by arbitrary shaped obstacles, while avoiding any collision with the obstacles as well as the environment boundaries. It has been proved that path planning problem is NP-Complete [1]. One of the most simplified versions of the motion planning problem is planning a collision-free path for a robot among an arbitrary number of randomly-shaped obstacles, between two collision free configurations of the robot. Complexity analysis has shown this instance of the problem to be PSPACE-complete [2, 3]. In cases where the problem gets more complex by considering the physical properties, and actuator limitations of a real robot, it is not known if the problem is even decidable except for some particular cases [4]. Some of the well-known conventional motion planning algorithms are cell decomposition and visibility roadmaps [5, 6]. These algorithms turned out to be computationally expensive and hard to implement in a practical planning setup and additional restrictions to the problem are required to use a conventional algorithm in a practical complex motion planning case [7, 8].

In most of the path planning applications, there is no prior information about the environment like positions of the obstacles and surrounding boundary. This class of path planning problems is called sensor-based or online path planning which in general mean path planning in unknown environment. There are a variety of works done in this field resulting in different approaches with their specific characteristics, advantages and drawbacks [9-12]. In this class of path planning, the motion decisions are made simultaneously while the robot moves and obtains new data from the environment. High complexity of motion planning application domains such as computer animation, planning in uncertainty and self-driving vehicles, requires intelligent and robust tools and methods. In general, a robot can perform well if it can act as similar as possible to human behavior. Particularly in navigation, following human strategy to maneuver unknown obstacles and still finding the shortest possible distance can lead to an efficient performance.

The introduction of recent powerful computational methods inspired their potential positive application in path planning. Methods such as Fuzzy Logic Control [13-15], Neural Networks [16], Genetic Algorithms [17, 18], Ant Colony Optimization [19] and Simulated Annealing [20] have all been applied in robot path planning. Khatib [21] proposed a potential field method such that artificial forces repelled the robot away from the obstacles and attracted it towards the goal position. Potential fields were also applied for mobile robots in [22], however they suffered from falling into local minima and performed poorly in narrow regions [23]. Sensor based reactive planning methods have been proposed [24-26]. Control based methods require formulating accurate models for the robot and the environment [27, 28], which can be a rather daunting task. One of the best intelligent tools for this purpose if fuzzy logic [29].

Fuzzy logic is a powerful tool to include reasoning in the decision-making process of a machine. In this paper, a fuzzy logic-based motion planner is proposed which receives environmental information from the robot's

Manuscript received April 7, 2018; revised November 1, 2018.

doi: 10.18178/ijmerr.8.1.6-11

sensor(s) and generates linear and angular velocity of the robot. The proposed fuzzy controller uses the readings of a simple range finder sensor such as LiDar or a 3D camera and computes the values of three basic variables measuring the distance from the start position, the distance to the goal position and the distance to the closest obstacle. Next, the linear and angular velocity values are calculated according to the output of the fuzzy controller. This process repeats with a pre-defined frequency until the robot reaches the goal or concludes that no feasible path exists. The main advantage of the proposed algorithm is the optimality of the generated results. Furthermore, the failure rate of the planner is significantly reduced due to the input variable selection of the fuzzy system.

The remainder of this paper is organized as follows: Section II describes the proposed algorithm and the details of the mobile robot and the fuzzy controller. The simulation and experimental results are provided in section III and finally the work is concluded, and potential future work is discussed in section IV.

II. The Proposed Planner

A. Robot Specifications

In the framework of this work, a robot explores an environment with a finite number of obstacles. The environment is a connected subset of R^2, filled with arbitrary shaped obstacles. The environment (Φ) is formed by the union of two subset namely free space (Φ_{free}) which is the part of the environment that is not occupied by obstacles, and obstacle space (Φ_{obs}), locates the regions filled by obstacles.

$$\Phi = \Phi_{\text{free}} \cup \Phi_{\text{obs}} \tag{1}$$

For simplicity, we assume the robot is a disk which always knows its current and the goal positions. This assumption is close to the actual robot that has been used in the experimental studies as presented in section III. However, the robot can also be treated as a point. For making this consideration, we need to expand the environment with the size equal to the radius of the robot, as represents in Fig. 1.

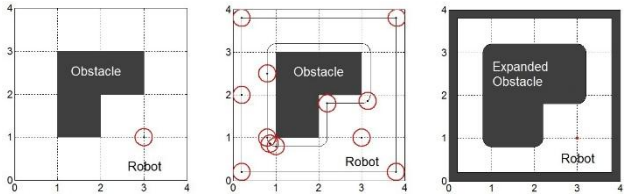


Figure 1. Environment expansion. (a) The actual environment. (b) Expanding the environment with the size of robot's radius. (c) Expanded environment.

Based on the performance of the robot sensory system, we formulate the robot's vision as follows. At each repetition of the algorithm, the robot can perform a scan of its surrounding area to determine the free space. Consider the robot is placed at $C \in \mathbb{R}^2$, with rays radially emanating from it. For each $\theta_j \in \Theta$, the value $\psi(C, \theta_j)$ is the distance to the closest obstacle along the ray from C at the angle θ_j. If there is no visible obstacle in the direction of θ_j, then $\psi(C, \theta_j)$ is equal to the Range of the sensors.

$$\psi(C, \theta): \mathbb{R}^2 \times \Theta^1 \rightarrow \mathbb{R}, \tag{2}$$

where:

$$\Theta = [0, 2\pi) \quad \text{and} \quad 0 < \psi(C, \theta) \leq Range \tag{3}$$

$$\psi(C, \theta) = max_\lambda \left\{ \lambda \times \left[cos(\theta_j), sin(\theta_j) \right]^{\mathrm{T}} \right\} \tag{4}$$

such that:

$$0 < \lambda \leq Range \quad \text{and} \quad \left[C + \psi(C, \theta_j) \right] \in \Phi_{\text{free}} \tag{5}$$

Fig. 2 illustrates the performance of the sensory system. The environmental scan process is evoked frequently with a predefined frequency. Each time the robot scans the environment, the distance to the surrounding obstacles is measured. Further, the distance of the robot to the initial and final positions is measure. In Fig. 2a, the robot scans its surrounding environment completely. The value of $\psi(C, \theta)$ is plotted for all possible amounts of θ in Fig. 3b, while in Fig. 3c, a difference function is used to point out the visible vertices of the obstacles.

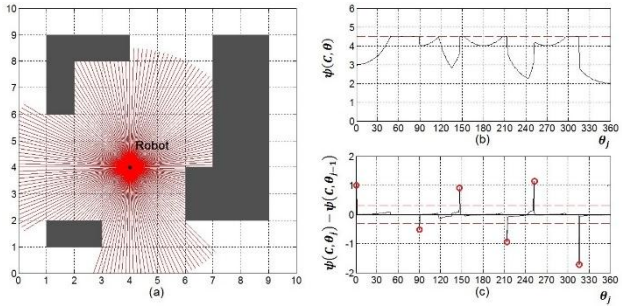


Figure 2. The performance of the sensory system. (a) A complete 360° scan of the surrounding environment. (b) The value of $\boldsymbol{\psi(C, \theta)}$ for different angles. (c) By subtracting the values of $\boldsymbol{\psi(C, \theta)}$ for any two adjacent angles, obstacles vertices are identified by sharp peaks which are shown by red circles.

B. Fuzzy Controller

As mentioned before, the main idea of the proposed approach is to guide the robot to choose the proper linear and angular velocities. A fuzzy controller was designed to check the readings of the sensor(s) and act accordingly. As the robot starts to move, the controller initiates the environment scan process with a given frequency, receives the reading from sensor(s) and calculates the value of three fuzzy variables including D_G, D_S, and D_{obs}.

The first fuzzy variable (D_G) is the robot's current distance to the goal. This part of the controller forces the robot to move closer to the goal continuously. This variable can be formulated as follows. Note that the value is normalized using the diagonal of the environment, δ. $D\{A, B\}$ is the Euclidean distance between points A and B.

$$D_G = \frac{1}{\delta} D\{C, G\} = \frac{1}{\delta}\sqrt{(x_c - x_g)^2 + (y_c - y_g)^2} \quad (6)$$

$$D_G \in [0, 1] \quad (7)$$

Three linguistic variables are defined for D_{NG} including "Close", "Far", and "Normal". As much as D_G gets closer to zero, the robot gets closer to the final position.

Next element of the controller guides the robot to move farther from the start position continuously by calculating the robot's distance to the start position as follows. Again, this value is normalized using the maximum reading range of the sensor(s), $Range$.

$$D_S = \frac{1}{\delta} D\{C, S\} = \frac{1}{\delta}\sqrt{(x_c - x_s)^2 + (y_c - y_s)^2} \quad (8)$$

$$D_S \in [0, 1] \quad (9)$$

Three linguistic values are considered for this variable including "Close", "Far", and "Normal".

The last element of the controller guides the robot to avoid surrounding obstacles. Based on the readings of the sensor(s), the distance to the closest obstacle, D_{obs}, is calculated as shown in Fig. 2(a).

$$D_{obs} = (\frac{1}{Range}) \min_{\theta} D\{robot, Obstacle\} \quad (10)$$

$$D_{obs} \in [0, 1] \quad (11)$$

Three linguistic values are considered for this variable including "Close", "Far", and "Normal". The membership functions of these variables are presented in Fig. 3. All three normalized input variables range from zero to one and have the same type of membership functions with similar parameters.

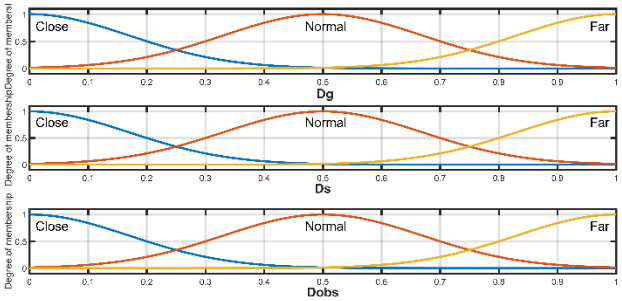


Figure 3. The corresponding membership functions to the input variables.

The output fuzzy variables include the linear velocity, $v(\frac{m}{\text{sec}})$, and the angular velocity, $\omega(\frac{deg}{\text{sec}})$, of the robot. The linear velocity unit is meters per second and it is bounded to one meters per second. The angular velocity unit is degrees per second and is bounded to 90 ° per second. This restrictions on v and ω are optional and does not affect the performance of the planner. The membership functions of the output variables are shown in Figure 4.

Again, similar membership functions have been used. All of the input and output variables have Gaussian membership functions with the following details:

$$f(x, \sigma, c) = e^{\frac{-(x-c)^2}{2\sigma^2}} \quad (12)$$

The value of σ is set to be 0.1677 for all functions and the value of c depends on the position of each function.

The next step is to design the fuzzy rules. Since there are three input variables which each can take one of the three different options including "Close", "Normal", and "Far", there will be $3^3 = 27$ fuzzy rules in the proposed system as presented in Table I. These rules have been crafted based on trial and error. For designing the fuzzy rules and membership functions, we tried to avoid using other available methods such as genetic algorithm [30] or ANFIS [31] to keep the simplicity of the work. Furthermore, the results of the proposed method were good enough to avoid any further improvement at the moment. However, it is possible to investigate the effect of using a more structured way for constructing and optimizing the proposed fuzzy system.

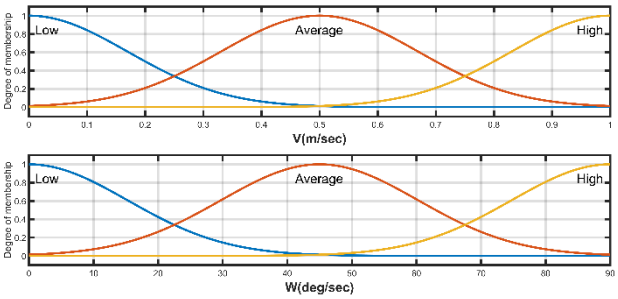


Figure 4. The corresponding membership functions to the output variables.

The value of σ is set to be 0.1677 for all functions and the value of c depends on the position of each function.

The next step is to design the fuzzy rules. Since there are three input variables which each can take one of the three different options including "Close", "Normal", and "Far", there will be $3^3 = 27$ fuzzy rules in the proposed system as presented in Table I. These rules have been crafted based on trial and error. For designing the fuzzy rules and membership functions, we tried to avoid using other available methods such as genetic algorithm [30] or ANFIS [31] to keep the simplicity of the work. Furthermore, the results of the proposed method were good enough to avoid any further improvement at the moment. However, it is possible to investigate the effect of using a more structured way for constructing and optimizing the proposed fuzzy system.

Figure 5 presents the decision surfaces for the given fuzzy logic controller. All possible arrangements of input and output variables are considered here to give an overall understanding of the behavior of the controller.

As explained before, the fuzzy planner takes the readings of the sensor(s) and calculates three fuzzy variables namely, $D_G, D_S,$ and D_{obs}. These values will be fed to the fuzzy system to get the outputs including v and ω. For instance, if the three input variables are $D_G = 0.8\ (m), D_S = 0.3\ (m)$ and $D_{obs} = 0.5\ (m)$ then the output of the fuzzy controller will be $v = 0.652\ m/sec$ and $\omega = 31.4\ deg/sec$. An important part of the

proposed algorithm is setting the value of the environmental scan frequency.

TABLE I. DESIGNED FUZZY RULES IN THE PROPOSED CONTROLLER.

#	D_g		D_S		D_{obs}	$v_{(m/s)}$	$\omega_{(deg/s)}$
1	Close	AND	Close	AND	Close	Low	High
2	Close	AND	Close	AND	Normal	Average	Average
3	Close	AND	Close	AND	Far	High	Average
4	Close	AND	Normal	AND	Close	Low	High
5	Close	AND	Normal	AND	Normal	High	Low
6	Close	AND	Normal	AND	Far	High	Low
7	Close	AND	Far	AND	Close	Low	High
8	Close	AND	Far	AND	Normal	High	Low
9	Close	AND	Far	AND	Far	High	Low
10	Normal	AND	Close	AND	Close	Low	High
11	Normal	AND	Close	AND	Normal	Average	Average
12	Normal	AND	Close	AND	Far	High	Low
13	Normal	AND	Normal	AND	Close	Low	High
14	Normal	AND	Normal	AND	Normal	High	Average
15	Normal	AND	Normal	AND	Far	High	Low
16	Normal	AND	Far	AND	Close	Low	High
17	Normal	AND	Far	AND	Normal	Average	Low
18	Normal	AND	Far	AND	Far	High	Low
19	Far	AND	Close	AND	Close	Low	High
20	Far	AND	Close	AND	Normal	High	Average
21	Far	AND	Close	AND	Far	High	Low
22	Far	AND	Normal	AND	Close	Low	High
23	Far	AND	Normal	AND	Normal	High	Low
24	Far	AND	Normal	AND	Far	High	Low
25	Far	AND	Far	AND	Close	Low	High
26	Far	AND	Far	AND	Normal	High	Low
27	Far	AND	Far	AND	Far	High	Low

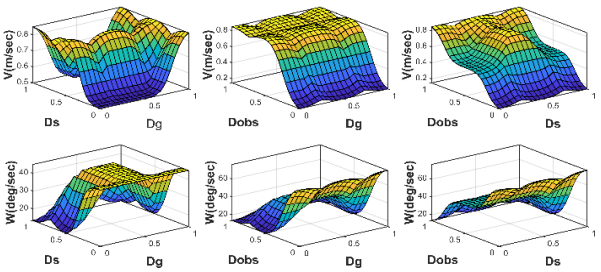


Figure 5. The corresponding decision surfaces of the proposed fuzzy controller. In each plot, the value of the missing input variable is set to 0.5.

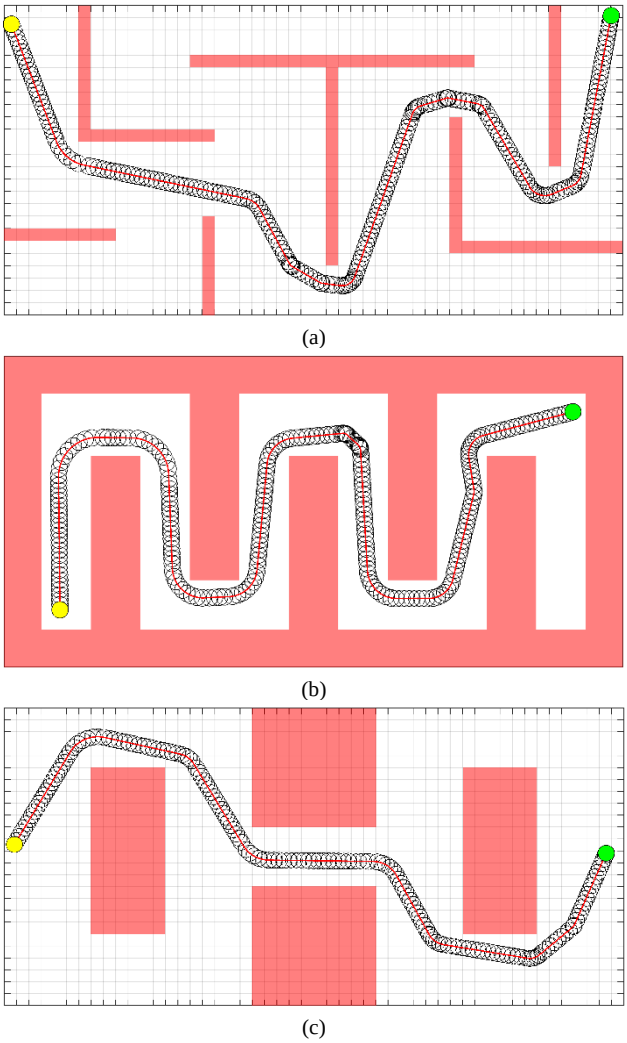
Having the planner to evoke the scan process more often increases the quality of the solution but also increases the runtime of the planner. On the other hand, setting a low frequency for the scan process reduces the quality of the solution but makes the planning computationally cheap. After a careful implementation, we decided to set the scan frequency to $f_{scan} = 0.5$ (sec). It means every 0.5 second; the planner evokes the scan process and changes the linear and angular velocity if it is recommended by the fuzzy system.

III. RESULTS AND DISCUSSION

In this section, the results of the simulation and experimental studies are presented.

A. Simulation Results

The planner was simulated in MatLab R2017a to perform in three different planning scenarios as presented in Fig. 6. All simulations were run on a desktop with a 3.40-GHz Intel Core i7 processor with 32 GB of memory. Only the initial and final positions were given to the planner. The planner managed to guide the robot through safe and efficient trajectories without any failure or collision with the obstacles.



(a)

(b)

(c)

Figure 6. An instance of the simulation results in three 2D indoor environments including (a) an office like environment with 6 separate obstacles, (b) a maze and (c) an environment with four obstacles which form a narrow passage. The dimension of all three problems was set to 25x50. Initial and final positions of the robot are marked by yellow and green, respectively.

The numerical results of the simulations are presented in Table II in terms of solution cost, computational cost, failure rate and the optimality. Solution cost is calculated as the total distance travelled by the robot, computational cost is the total time used for running the fuzzy controller without including the pure navigation time. Failure is when the robot is not able to reach the goal or conclude that there is no solution. Finally, optimality percentage was calculated using the optimal path generated by offline visibility graph approach [32].

TABLE II. NUMERICAL RESULTS OF THE SIMULATIONS IN THREE TEST PROBLEMS.

Problem	Solution Cost (m)	Computation Cost (sec)	Failure Rate (%)	Optimality (%)
Office	79.13	14.57	0	91.04
Maze	108.34	16.80	0	93.29
Narrow	72.63	15.32	0	93.20

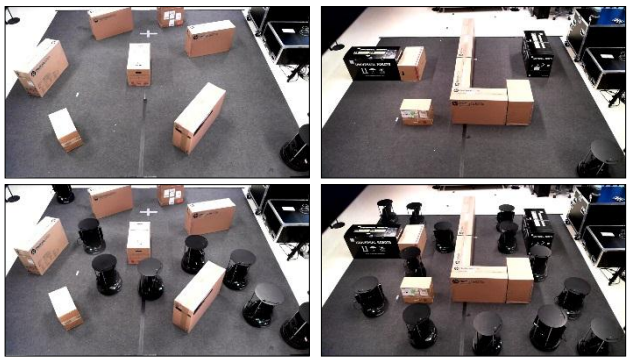
The failure rates are entirely zero while the optimality is above 90%. Specially in the office example, the level of optimality and failure rate is important since there are different corridors which may not end up to the desired position.

B. Experimental Results

To test the planner on a real robot, two different experiments was conducted on a Turtlebot2 with an Asus Xtion Pro Live camera, and an onboard computer with a 2.60-GHz Intel Core i5 processor with 8 GB of memory, as presented in Fig. 7, in two different planning problems as shown in Fig. 8. And Table III.

(a) (b)

Figure 7. The robotic system used in the experimental studies. (a) A TurtleBot2 with an Asus Xtion Pro Live camera, and an onboard computer with a 2.60-GHz Intel Core i5 processor with 8 GB of memory and (b) The Asus Xtion Po Live camera used for environmental scan and obstacle avoidance.



(a) (b)

Figure 8. (Top images) Two setups for the experimental studies with the results of the proposed planner. (Bottom images) Using multiple exposure images, the position of the robot is shown over time. (a) A scattered environment with six different obstacles in random positions, and (b) A maze where the robot should navigate throughout the maze in order to reach the final position. The environment is a $4.03m \times 3.78m$ polygon.

First, the robot is moving in a 2D bounded environment with six randomly scattered obstacles. Second, the robot is trapped in a maze and should travel the entire maze in order to reach the desired position. This is even more challenging as the robot gets lost in the maze for a few seconds and has to come back to the entrance of the maze and change the course of motion to reach the goal.

TABLE III. NUMERICAL ANALYSIS OF THE EXPERIMENTAL STUDIES.

Measure	Setup	
	Scattered	Maze
Path Length (m)	8.16	16.29
Runtime (sec)	3.22	5.73
v_{max} (m/sec)	0.93	0.88
v_{min} (m/sec)	0.37	0.00
ω_{max} (deg/sec)	82.17	90.00
ω_{min} (deg/sec)	0.00	0.00
D_{min} (m)	0.15	0.08

Fig. 9 shows the effect of each individual input variables on the linear and angular velocity of the robot. For examples, as the robot gets closer to an obstacle, the linear velocity decreases to avoid collision while the angular velocity increases so the robot can maneuver around the obstacle.

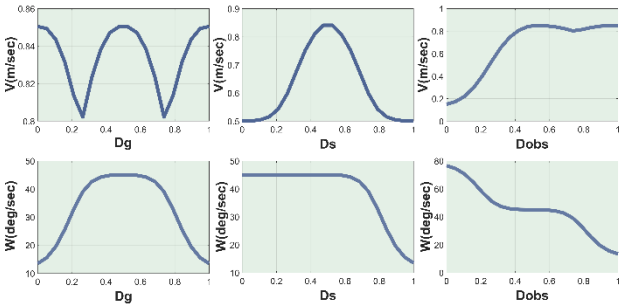


Figure 9. The effect of individual fuzzy input variables on the linear and angular velocity of the robot. For each graph, the value of other two input variables is set to 0.5.

IV. CONCLUSION

In this paper, a fuzzy logic-based motion planner is presented which controls a nonholonomic mobile robot in unknown environments. The fuzzy planner uses the readings of the robot's sensory system and calculates three fuzzy variables hoping to move the robot closer to the final position, farther from the initial position and away from the surrounding obstacles. The outputs of this fuzzy system are the linear and angular velocities of the robot and will be used to adjust the motion speed and direction.

Several simulation and experimental studies have been implemented to test the performance of the proposed planner in different typical navigation problems. The planner generates safe and stable results with low computational requirements while the failure rate is zero and the optimality rate is more than 90%. Implementing the proposed work on a TurtleBot shows the applicability of the planner in real implementations.

This work can be further improved by implementing more deterministic tools for designing the elements of the fuzzy controller. It can also be studied to see if using an optimization technique for optimizing the fuzzy membership functions and fuzzy rules has a notable effect on the results.

ACKNOWLEDGMENT

This work is supported by the Research Council of Norway as a part of the Multimodal Elderly Care Systems (MECS) project, under grant agreement 247697.

REFERENCES

[1] J. C. Latombe, *Robot Motion Planning*, Springer Science & Business Media, 2012.
[2] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.
[3] J. Canny, "Some algebraic and geometric computations in PSPACE," in *Proc. the Twentieth Annual ACM Symposium on Theory of Computing* - STOC '88, 1988.
[4] P. Cheng, G. Pappas, and V. Kumar, "Decidability of motion planning with differential constraints," in *Proc. 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007.
[5] H. M. Choset, Principles of robot motion: theory, algorithms, and implementation, 2005 MIT press.
[6] S. M. LaValle, "Planning algorithms," MIT Press, 2006.
[7] D. Halperin and C. K. Yap, "Combinatorial complexity of translating a box in polyhedral 3-space," *Computational Geometry*, vol. 9, no. 3, pp. 181–196, Feb. 1998.
[8] S. Hirsch and D. Halperin, "Hybrid motion planning: Coordinating Two discs moving among polygonal obstacles in the plane," *Algorithmic Foundations of Robotics V*, pp. 239–255, 2004.
[9] J. Ziegler, H. Gattringer, D. Kaserer, and A. Müller, "Automated, depth sensor based object detection and path planning for robot-aided 3D scanning," *Mechanisms and Machine Science*, pp. 336–343, July 2017.
[10] J. Minguez, F. Lamiraux, and J. P. Laumond, "Motion planning and obstacle avoidance," *Springer Handbook of Robotics*, pp. 1177–1202, 2016.
[11] H. Wang and S. J. Julier, "Path planning in partially known environments," *Biomechanics / 752: Robotics*, 2012.
[12] M. Cefalo, E. Magrini, and G. Oriolo, "Parallel collision check for sensor based real-time motion planning," in *Proc. 2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.
[13] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 3, pp. 464–477, Mar. 1995.
[14] H. Martínez-Alfaro and S. Gómez-García, "Mobile robot path planning and tracking using simulated annealing and fuzzy logic control," *Expert Systems with Applications*, vol. 15, no. 3–4, pp. 421–429, Oct. 1998.
[15] K. Park and N. Zhang, "Behavior-based autonomous robot navigation on challenging terrain: A dual fuzzy logic approach," in *Proc. 2007 IEEE Symposium on Foundations of Computational Intelligence*, Apr. 2007.
[16] D. Janglova, "Neural networks in mobile robot motion," *Cutting Edge Robotics*, July 2005.
[17] F. A. Afsar, M. Arif, and M. Hussain, "Genetic algorithm based path planning and optimization for autonomous mobile robots with morphological preprocessing," in *Proc.* 2006 IEEE International Multitopic Conference, Dec. 2006.
[18] Y. R. Hu, S. X. Yang, L. Z. Xu, and M. Q. H. Meng, "A knowledge based genetic algorithm for path planning in unstructured mobile robot environments," in *Proc.* 2004 IEEE International Conference on Robotics and Biomimetics.
[19] M. A. P. Garcia, O. Montiel, O. Castillo, R. Sepúlveda, and P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," *Applied Soft Computing*, vol. 9, no. 3, pp. 1102–1110, June 2009.
[20] T. Maekawa, T. Noda, S. Tamura, T. Ozaki, and K. Machida, "Curvature continuous path generation for autonomous vehicle using B-spline curves," *Computer-Aided Design*, vol. 42, no. 4, pp. 350–359, Apr. 2010.
[21] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, Mar. 1986.
[22] P. Nattharith, "Motor schema-based control of mobile robot navigation," *International Journal of Robotics and Automation*, vol. 31, no. 4, 2016.
[23] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. 1991 IEEE International Conference on Robotics and Automation*.
[24] V. Ayala-Ramirez, J. A., R. Lopez-Padilla, and R. E., "An artificial protection field approach for reactive obstacle avoidance in mobile robots," *Mobile Robots Navigation*, Mar. 2010.
[25] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proc. IEEE International Conference on Robotics and Automation*.
[26] F. Belkhouche and B. Bendjilali, "Reactive path planning for 3-D autonomous vehicles," *IEEE Transactions on Control Systems Technology*, 2011.
[27] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, Feb. 2007.
[28] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, Dec. 2011.
[29] T. Abbas, M. Arif, and W. Ahmed, "Measurement and correction of systematic odometry errors caused by kinematics imperfections in mobile robots," *2006 SICE-ICASE International Joint Conference*, 2006.
[30] R. AlcalÁ, M. J. Gacto, F. Herrera, and J. AlcalÁ-Fdez, "A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 15, no. 05, pp. 539–557, Oct. 2007.
[31] P. K. Mohanty and D. R. Parhi, "A new intelligent motion planning for mobile robot navigation using multiple adaptive neuro-fuzzy inference system," *Applied Mathematics & Information Sciences*, vol. 8, no. 5, pp. 2527–2535, Sep. 2014.
[32] Han-Pang Huang and Shu-Yun Chung, "Dynamic visibility graph for path planning," in *Proc. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE Cat. No.04CH37566).

Weria Khaksar received the Ph.D. degree in industrial engineering from University Putra Malaysia in 2013. He is currently a Post-Doctoral Research Fellow with the Department of Informatics, University of Oslo, Norway. His main research interests include robotics, AI, and machine learning with special focus on motion planning and probabilistic robotics.

Md. Zia Uddin received the Ph.D. degree in biomedical engineering in 2011. He is currently a Post-Doctoral Research Fellow with the Robotics and Intelligent Systems Research Group, Department of Informatics, University of Oslo, Norway. He authored or co-authored over 75 research publications, including international journals, conferences, and book chapters. His research is mainly focused on computer vision, image processing, artificial intelligence, and pattern recognition.

Jim Torresen received the Ph.D. degree in computer science from the Norwegian University of Science and Technology. He is currently a Professor of computer science with the University of Oslo, Norway. His research interests include nature-inspired computing, adaptive systems, reconfigurable hardware, and robotics and their use in complex real-world applications.