Multilayer Decision-Based Fuzzy Logic Model to Navigate Mobile Robot in Unknown Dynamic Environments

Farah Kamil a and Mohammed Yasser Moghrabiah b

aTechnical Institute of Al-Diwaniyah, Al-Furat Al-Awsat Technical University (ATU), Technical Institute of Al-Diwaniyah, Al-Diwaniyah, Iraq; bDepartment of Computer and Communication System Engineering, Universiti Putra Malaysia, Selangor, Malaysia

ABSTRACT
The investigation into mobile robot navigation under uncertain dynamic environments is of great significance. This paper seeks to solve the current problems which are the difficulty to plan in indeterminate ever-changing environments, the problem of optimality, failure in complex situations, and the problem of predicting the obstacle velocity vector. The objective of this study is to propose a multilayer decision-based fuzzy logic model to find the solution for robot navigation through a safe path while preventing any types of barriers and to understand the non-collision mobile robots’ movement in an unknown dynamic environment. In this study, the prediction and priority rules of a multilayer decision are used by the fuzzy logic controller to improve the quality of the next position with regard to its path length, safety, and runtime. The results of comparison studies revealed a considerable improvement in failure rate and path length. Outcomes show that the suggested method displays attractive features, for instance, great stability, great optimality, zero failure rates, and low running time. The average path length for all test environments is 13.11 with 0.47 a standard deviation that provides 89% of an average optimality rate. The average running time is about 5.31 s with a 0.25 standard deviation.

1. Introduction
In society today, robotics can be considered an important and necessary factor [1]. Recently, methods concerning robots have attracted the interest of many researchers due to the capability of robots to work in hostile and dangerous environments instead of humans [2]. Path planning, which deals with finding an optimal collision-free path from a start point to an endpoint target based on features such as time, distance and energy, is very significant and important for mobile robots [3]. The functions of mobile robot are: sense its environment, interpret these sensed information, improve the knowledge of its position, plan the route from the initial start point to the end position by avoiding obstacles and controlling the turning angle and linear velocity of the mobile robot to attain the target [4].

In static environments, path planning is a problem that has been studied comprehensively and can be solved efficiently. However, path planning is still a challenge when several
dynamic obstacles are involved. This is because path planning needs to add time as an extra dimension to the search-space examined by the planner [5]. Several methods have been reviewed in detail by [6]. The classic approaches to mobile robot planning are inadequate and not able to overcome the challenges of a dynamic environment or inadequate information on the environment [7, 8]. Therefore, many reactive approaches are introduced. These approaches enable the utilisation of artificial intelligence techniques in which learning, reasoning and problem-solving is the main concerns. In this scope, artificial intelligence techniques, such as fuzzy logic and neural networks, become the foundation of navigation systems in mobile robots [9–11].

Fuzzy Logic (FL) is a type of artificial intelligence that handles mimicking human thinking and decision-making that enables intermediate values to be interpreted among traditional values, such as (yes/no), (true/false), (high/low), etc. Many studies on mobile robot navigation have used fuzzy logic approach [12]. Fuzzy logic procedure with four modules by M. Faisal et al. (2013) [9] are used to steer an autonomous mobile robot in unstructured, varying and unidentified environments to arrive to the destination and obstacle avoidance acts amongst four modules and exchanges the control among modules. The suggested technique is active and robust under dynamic obstacle scenarios. However, because of the use of several sensors to predict the surroundings, M. Faisal et al. remarked that they do not have any evidence on the actual time process. As well as due to the absence of testing and simulation, it is difficult to judge to what level that this system is efficient. By V. Raudonis and R. Maskeliunas (2011) [13], an indoor navigation system which makes use of fuzzy logic control was presented. V. Raudonis and R. Maskeliunas of the aforementioned study utilised fuzzy logic and a camera to move the robot to its goal. However, there was no attention being paid to avoiding obstacles. There was only utilised FLC for navigation.

In another study by S. Junratanasiri et al. (2011) [14], it was suggested a navigation system in an unknown environment that concentrates on dynamic obstacles for a mobile robot. A fuzzy vector was used to model the future position of a dynamic obstacle. Moreover, an interval type-2 fuzzy logic system was used to calculate the velocity and angular velocity of a mobile robot. The robot waited for an obstacle and then moved towards the target. This increased the arrival time. A fuzzy control process was used in a test for obstacle avoidance, together with motion planning. It enabled the mobile robot to look for the target, avoid obstacles and preserve heading (See H. Chang and T. Jin (2013)) [15]. However, the velocity vector of the obstacles was not considered. P. K. Mohanty and D. R. Parhi (2012) [16] utilised an adaptive Neuro-Fuzzy inference system (ANFIS) to generate a route and avoid obstacles for a self-directed mobile robot in an unknown and changing environment, but the velocity vector of the obstacles was not considered.

Gaussian noise was utilised by Y. Gao and S. Sun (2009) [17] to model the uncertainty that existed in an improved local path planning method. On the basis of compressed information, fuzzy logic controller was utilised to compute command velocity to better understand its capability and for more intelligent control. The position of the hazardous region (potential collision region) was predicted on the basis of the velocity of obstacles and their distribution. The hazardous region will be expanded according to its position’s distribution to guarantee security. The time cost involved in tracking one obstacle is another issue. This causes inefficiency in a highly dynamic complicated environment. Fuzzy logic controller with potential field method was used in another study by A. A. Ahmed et al. (2015) [18] to solve the disadvantages of the artificial potential field method, such as local minima
problems, making an effective motion planner and improving the quality of the mobile robot’s path. This method failed in a crowded dynamic environment, in which there were obstacles with different relative velocities.

C. Chinag and C. Ding (2014) [19] presented a robot navigation method for a dynamic environment with static and dynamic obstacles. The robot avoids the static obstacles by utilising a logic controller. Moreover, it avoids mobile obstacles by applying a Trajectory Prediction Table to predict the future trajectory of obstacles. There are some issues to this method. Firstly, it regards the obstacles as a point without dimensions. Secondly, it consists of a complete map regarding the velocity and track of mobile obstacles before avoiding them. Finally, it does not work in some cases, such as when another dynamic obstacle appears suddenly. An improved plan of fuzzy controllers for differential mobile robots to move in outdoor environments over a predetermined path with no human interference was designed by O. Montiel et al. (2014) [20]. It was effective at reducing the accumulated mistakes that arose from intervals when an odometer system was utilised to the minimum level. However, the velocity vector of obstacles was not considered. It just worked on a complete map.

W. Khaksar et al. (2014) [21] suggested a fuzzy controller which uses the heuristic rules of Tabu search to enhance the superiority of generated samples before any further processing. However, this approach was only focused on static obstacles. Fuzzy logic controller is suggested by W. Dongshu (2011) [4] to resolve the robot routing and priority-based behaviour control so as to get the non-collision path of mobile robots in dynamic unidentified surrounding. This system can lead the mobile robot successfully, but it did not take into account the speed of obstacle. N. Wang et al. (2018) [22] proposed the BCTTC scheme for path tracking of a QUAV through constrained actuator dynamics and difficult unknowns. The whole QUAV system has been divided into five cascade subsystems linked by intermediate nonlinearities. A later study by N. S. Ahmad et al. (2018) [23] solved the challenging issue regarding accurate path tracking control of an AUSV suffering from difficult uncertainties without using the persistently required PE condition. FUOs have been developed to precisely recognise disturbed uncertainties stemming from transformed rotational and translational subsystems of the AUSV.

Zagradjanin, Novak, et al. (2021) [24] designed a self-directed cloud-based multi-robot system to perform extremely repetitive jobs in a changing environment for example a new megastore. Cloud level is proposed for execution of the utmost demanding processes to unload the robots which are users of cloud services in this design. D Lite method is applied for motion planning on a global level, considering its great efficiency in changing environments. So as to present an intelligent cost map for additional enhancement of motion planning in difficult and crowded surroundings, the application of a fuzzy inference system (FIS) and learning algorithm is suggested. This technique requires a complete map to implement while in our approach, the planner does not create a map of the environment or retain the examined region totally. The only things it needs to know are the current position and the goal position. Ayub, Shahanaz, et al. (2021) [25] designed and tested a hybrid neuro-fuzzy along with PSO optimisation for navigation of multi-robot. Automated modification of membership value variables in addition to fuzzy rule suggestions is essential to accomplish extra effective control regarding travelling period and distance traversed. The neuro-fuzzy switching method can professionally combine their characteristics; it is verified via the simulated results. Such design permits the robots to use action that matches
to the environments condition (the presence or absence of obstacles, in addition to if the surrounding is known or unknown). However, these techniques did not take into account the velocity vector of obstacles.

Sangeetha, Viswanathan, et al. (2021) [26] proposed a fuzzy gain-based dynamic ant colony optimisation (FGDACO) for dynamic path planning to effectively plan collision-free and smooth paths, with feasible path length and the minimum time. The ant colony system’s pheromone update mechanism was enhanced with a sigmoid gain function for effective exploitation during path planning. Collision avoidance was achieved through the proposed fuzzy logic control. However, this technique did not take into account the velocity vector of obstacles. The problem happens when the robot and obstacle transfer at the same speed and direction. Subsequently, the robot cannot pass the obstacle nor reach its target without predicting the obstacle’s velocity vector and changing the direction to the free direction.

The current problems that this paper seeks to solve are the difficulty to plan in indeterminate ever-changing environments, the problem of optimality, failure in complex situations, and the problem of predicting the obstacle velocity vector. The study gap is that there is no method from the previous methods that solve the above-mentioned problems simultaneously.

It has been revealed that a mixture of motion planning approaches, with other intelligent and heuristic methods, is an effective way of solving the problems mentioned earlier in this study and to act efficiently in complicated situations. The combination of our original approach with fuzzy logic is one of the most effective kinds of these hybrid systems. A fuzzy logic controller makes use of the priority and prediction rules of the multilayers decision approach for improving the quality of next position with regards to runtime, safety and path length.

The authors use fuzzy systems because planning is still challenging when there are some dynamic obstacles. That is because planning needs adding time as an extra dimension to the search-space examined by the planner. The obstacles in which are moving and their speed profiles are not pre-identified. Other approaches on mobile robot planning are not strong enough and not capable of overcoming the challenges of crowded complex situations where the environment is dynamic and unknown.

The objective of this study is to propose and use a Multilayer Decision-Based Fuzzy Logic Approach for avoiding static and dynamic obstacles on the basis of the velocity vector of the obstacles and priority behaviour. It is capable of planning navigation in a complicated dynamic environment and handling sensor-based planning tasks. In order to make the suggested approach more effective and intelligent, a fuzzy logic controller is used. This controller assesses the prediction procedure and decisions regarding the next position in the visible region and chooses the best next step based on several criteria to solve sensor-based planning problems in both hazardous and complicated dynamic uncertain environments. These criteria are prepared by the rules of the suggested approach and are being transformed into fuzzy variables to create the fuzzy reasoning system.

The following sections provide details of the proposed planner. Then, the planner is simulated in a complicated dynamic environment to assess its performance. The compared findings will be discussed to support the assertion of the superiority of the proposed planner.
Figure 1. Robot navigation issue in complex dynamic environments.

2. System Specifications

Figure 1 clarifies the common settings of the navigation problem. The movement of obstacles ($v$) and limited sensing ranges ($R_s$) can cause potential environmental changes. Finding a proper collision-free path that connects the initial position $p_0$ at $t_0$ to its final position $p_f$ at $t_f$ can be considered a navigation problem, where

$$p_0 = (x_0, y_0, \phi_0)$$
$$p_f = (x_f, y_f, \phi_f)$$

In dynamic obstacle scenario, determining the exact speed vector of each obstacle especially in different directions’ obstacles is toward to moving robot is a big issue.

The main problem occurs when the robot decides to move inside a dangerous region where three moving obstacles are moving toward each other and will collide with this next position. The robot cannot chose how to escape from them because the robot has a different solution for each moving obstacle. For instance, the decision about the obstacle that is moving towards the right direction is to move left and is different from the decisions about two other obstacles which are moving left and down. Therefore, a collision will happen unless it is able to predict the dangerous region and change its next position to another position, which has a lower risk of collision and unobstructed direction.

The robot is regarded as a Wheeled Mobile Robot (WMR). It is square-shaped, centred at $(r_x, r_y)$. As shown in Figure 2, it has two rear wheels that are autonomously-driven. Moreover, it has a castor front wheel. $r_c(t) = (r_x(t), r_y(t), r_\phi(t))$ shows the configuration of a square robot at time $t$. $r_x(t), r_y(t)$ determines the coordinates of the robot’s centre that the robot rotates around the origin. Moreover, $r_\phi(t)$ shows the orientation of the robot, as measured by its angle in relation to the positive X-axis.
The formulation of the kinematic model of the WMR with two independently driven rear wheels and a castor front wheel is:

\[ \dot{x} = v \cos \varphi \] (1)
\[ \dot{y} = v \sin \varphi \] (2)
\[ \dot{\varphi} = \omega \] (3)

In our situation, to attain a straight-line trajectory to achieve the shortest path length, it is presumed that:

\[ v_r(t) = v_l(t) = v(t) \] (4)
\[ \omega(t) = \dot{\varphi}(t) = 0 \] (5)

It is also presumed that the robot’s wheels do not slip. This is specified by an algebraic expression as follow:

\[ \dot{x} \sin \varphi - \dot{y} \cos \varphi = 0 \] (6)

The obstacles are specified through arbitrary shapes. Obstacle’s velocity is \((v_x, v_y)\) in which the elements on the X and Y axes are indicated by subscripts \(x\) and \(y\), respectively. Obstacles are either stationary or dynamic. The speeds of the obstacles are set randomly. Obstacle’s velocities are less than or equal to the velocity of the robots to give the robot enough time to detect the moving direction of the moving obstacles. The location of obstacles and their velocity vectors (orientation and speed) are not identifiable to the robot. It is assumed that obstacles move along arbitrary trajectories and are recognisable by the robot. Recognisable does not mean that the obstacles are pre-known to the robot, but it means that the robot can recognise the obstacles status through its sensors to determine their status.

The location and speed of the obstacles are not recognised by the robot; therefore, it must be made ready with range sensors or detectors to acquire the necessary information. The robot becomes ready with range sensors of a finite direction of 360 degrees and acquires information from its surroundings. Its detecting range is a circle centred at \((x, y)\) with radius \(R_s\), through which it senses the positions of obstacles and performs a visibility scan. When it arrives at a new position in the configuration space, the robot first computes its distance to neighbouring obstacles using its radial sensor readings. It then stores the
result in a visibility matrix, which includes visible obstacle points’ position in two sequential repetitions (time intervals) to evaluate the speed vector of each obstacle.

3. Multilayer Decision Algorithm

In this part of the study, the concept of the original approach will be explained using several processes [27]. The first behaviour is goal-seeking, which causes the robot to move from its source to a destination position with angle \( \varphi \) and distance \( d \). This happens through steering direction and the Euclidean equations among the start and goal positions \((x_1, y_1), (x_2, y_2)\), respectively, as shown below:

\[
\varphi = \tan^{-1}\left(\frac{|y_2 - y_1|}{|x_2 - x_1|}\right) \quad (7)
\]

\[
d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (8)
\]

Similar to other navigation algorithms, the approach of this study starts by checking the reachability of the destination. When the destination is reachable, the robot starts moving straight towards the target and the algorithm is ended. Otherwise, the next behaviour, known as predicting trajectories of obstacles, is evoked by the algorithm. In this behaviour, the surrounding region is sensed by the robot based on the range sensor to check whether there is any obstacle or not. When the sensor detects an obstacle, the robot examines it and discovers whether it is a dynamic or static obstacle.

When the prediction reveals that obstacle’s position does not alter, this signifies that the obstacle is static; otherwise, the obstacle is dynamic. Then, the direction of the moving obstacle is evaluated. Next, the robot selects the next stage based on the obstacle’s future velocity vector and priority limitations. Therefore, the robot moves from its original path when the future trajectory of the obstacle crosses its path.

The following concepts lead to the achievement of the predicting behaviour. The range sensor consists of four circles with various radii, including \( R_1, R_2, R_3, \) and \( R_4 \). The smallest circle is the least safe threshold between the robot and the obstacle, while the biggest circle is the maximum range of the sensor. Moreover, there are two other medium circles (between the smallest and biggest circles) (refer to Figure 3). We can formulate this as:

\[
\text{Circle}_i = \text{Check around } (R_x, R_y, R_i); \text{ where } i = 1, 2, 3, 4, \text{ and } R = 0.2, 0.4, 0.6, 0.8 \quad (9)
\]

The reasons to split the sensor range into ‘four’ circles are:

1. Give the precious description about the exact distance between the robot and the obstacles.
2. Give the robot enough time to detect the moving obstacle to take the decision about the next step.
3. Determine the exact direction about the movement of the obstacle.
4. Cover the actual distance of the rangefinder that is 80 cm [28].

These circles are split into four regions to evaluate the direction of the moving obstacle together with its position. The first region is between 0 and 90°, second 90° to 180°,
third $180^\circ$ and $-90^\circ$ and fourth $-90^\circ$ and $0$ (refer to Figure 4). The robot should record the intersection points between the positions of the obstacle and these circles. Moreover, it determines which one of these regions has the intersection points. The sensor reading records 1 for any position inside the obstacle and 0 for any position outside the obstacle to estimate the future trajectory of the moving obstacle. It then stores the result in a visibility matrix, which includes visible obstacle points’ position in two-time intervals to evaluate the speed vector of each obstacle. The visibility matrix is as follow:

| Time | Static obstacle | Dynamic obstacle | Right obstacle | Left obstacle | Front obstacle |
|------|-----------------|------------------|---------------|--------------|---------------|
| $t$  | 0               | 1                | 0             | 0            | 1             |
| $t + T$ | 0       | 1                | 1             | 1            | 1             |

Intersection Points of Circle,

$$= \text{Check in Polygon} \left( \text{No. of polygon}, R_x, R_y, x_{\text{polygon}}, y_{\text{polygon}} \right)$$

When the robot detects an obstacle, it stops to determine the status of the obstacle. The obstacle is static when the prediction shows that the number of intersection points does not alter. However, if it reveals that this number changes, it means that the obstacle is dynamic. Thereafter, the measurement of the intersection points’ number is used as a means to determine the direction of the moving obstacle. When the number of the intersection points increases, this signifies that the direction of the moving obstacle is towards the robot. However, if this number decreases, the obstacle is moving away from the robot. For example, when the direction of the obstacle is towards the first region, the robot moves...
away from this region. We can formulate the number of intersection points at each quarter as:

\[
\text{No. Intersection Points in Quarter}_i = \sum_{i=1}^{4} (\text{Intersection Points of Circles in Quarter}_i) \quad (11)
\]

The density of intersection point does not decide the performance of runtime and path length because it only focuses on if there is any obstacle around to decide the status and the location of obstacles to avoid failure.

Succeeding that, priority behaviour is provoked by finding a proper position. When the obstacle is static and intersects with the trajectory of the robot, the robot will digress from its original path based on being near to the goal priority. However, when the obstacle is dynamic, the future velocity vector of the moving obstacle can be predicted by the robot to determine which one of the directions is best. The robot then evaluates its next position and direction based on three priorities:

1. The distance of the robot to the destination should be decreased ceaselessly in every part of the navigation. As a result, when there are 2 positions to the target, the next robot’s position should be nearer to the destination \((N_1)\) compared to the other location of the robot \((N_2)\). We can express this constraint as:

\[
\text{Next position}(N_i) = \text{Current position}(C_{i-1}) + \text{Step} \times [\cos \varphi_i, \sin \varphi_i]^T \quad (12)
\]

\[
\text{Best next position}(N_{bi}) = \begin{cases} 
N_1 & \text{if } \text{Dis}_{N1,G} < \text{Dis}_{N2,G} \\
N_2 & \text{otherwise}
\end{cases} \quad (13)
\]

2. The next location of the robot should not intersect with the moving obstacles’ future predicted trajectory points in the environment. We can formulate this constraint as:

\[
\text{Next position}(Nfi) = \text{Current position}(C_{i-1}) + \text{Step} \times [\cos \varphi_i, \sin \varphi_i]^T \notin \text{Future obstacle trajectory} \quad (14)
\]

3. After the robot has passed the obstacle, it should go back to its original path and arrive at its destination in order to maintain a time and keep its shortest path based on distance \((d)\) and direction equations \((\varphi)\) which were presented in Formulas (7) and (8), respectively.

Note: we suggested that the robot should go back to its original path after passing the obstacle because the environment is unknown to the robot and the robot only knows the current position and the goal position. Consequently, the robot should follow the original path to prevent missing the goal position.

When the algorithm discovers a new place, it will be robot’s next location. The program computes the arrival time and path length when the robot arrives at the destination as follows:

\[
d_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \quad i = 1, 2, \ldots, n \quad (15)
\]
where \( n \) is the number of points along the route. The total length of the path is the summation of the distances. In the equation, \( L \) is the total length of the path.

\[
\text{fl}L = L = \min \sum_{i=1}^{n-1} (d_i) i = 1, 2, \ldots, n
\]

\[
r(t_f) = D; \; r(t) \notin \bigcup_{i=1}^{n} (O_i(t)) \forall t \in [0, t_f]
\]

where \( r(t) \) is the position of the robot at \( t \) period, \( D \) is the destination, and \( O(t) \) is the location of the obstacle at \( t \) period. \( \bigcup_{i=1}^{n} (O_i(t)) \) is the union of the location of the obstacle at \( t \) period for all number of points along the route.

The relationship between \( O \) and \( n \) is that in sometimes, the \( n \) is corresponding to \( O \) when there is an obstacle on the path and other times is not when the path is empty.

No promising solution exists for this navigation problem, when the planner is not capable of finding a location. For the original proposed algorithm, the pseudo-code is as follows:

**Multilayer Decision Algorithm:** The pseudo-code for the algorithm

- **Input:** initial position and end position
- **Output:** a safe path from start to goal

**While** (the robot does not reach the target),

1. Divide the range sensor into four circles with various radii
2. Detect whether exist any obstacle around the robot;
   - If no obstacle around the robot then
     - Goal seeking behavior;
   - End if
   - If an obstacle is detected by sensor then
     - Record obstacle's position in two successive iterations;
     - Record the intersection points between the positions of the obstacle and these circles
     - If the number of intersection points does not alter then
       - The obstacle is static
       - Determine the obstacle’s position by determines which one of these regions has the intersection points
       - If the position of the obstacle crosses its path
         - Do the static avoidance behavior;
         - Find best next position depend upon the priority limitations;
       - End if
     - End if
   - End if
   - Else if the number of intersection points alters then
     - The obstacle is dynamic
     - If the number of intersection points increases then
       - The direction of the moving obstacle is towards the robot.
       - If the future trajectory of the obstacle crosses its path then
         - Do dynamic avoidance behavior using predicting velocity vector of moving obstacle;
       - End if
     - Else if the number decreases then
       - The obstacle is moving away from the robot.
       - Find best next position far from dangerous region depend upon velocity vector rules;
     - End if
   - End if
- End if
End while
4. Fuzzy Logic Controller

As mentioned earlier, the main objective of this approach is to direct the robot to select the best next step between the collision free positions. For the purpose of checking the prediction process and decisions about the next position, and selecting better positions with regards to minimum risk, a fuzzy logic controller was designed. This controller is invoked after the next step locates a collision-free position. The values of six fuzzy variables, including right static (RS), left dynamic (LD), front dynamic (FD), left static (LS), front static (FS) and right dynamic (RD), are calculated by the controller at each iteration.

The first fuzzy variable (LS) is the distance to the left static obstacle, which is defined as the intersection points (IP) between the sensor range and the obstacle. The robot learns to identify the safe distance to the left static obstacle when found through this part of the controller. The controller measures the intersection points between the sensor layers and the obstacle. If the obstacle intersects the maximum layer of the sensor, this means the obstacle is far from the robot. Meanwhile, if the obstacle intersects the middle sensor’s layer, this shows that the distance between the obstacle and the robot is safe. However, if the intersection occurs with the minimum layer, the robot is near to the obstacle. By subtracting these values, we have (LS), which will be used in future fuzzy evaluations. This variable can be formulated as follows.

Sensor Layers \( (R_i = 0.2, 0.4, 0.6, 0.8) \), Range quarters \( (q_j = 1^{st}, 2^{nd}, 3^{rd}, 4^{th}) \) \( (18) \)

\[ IP = (n, R_x(q_j), R_y(q_j), x_{polygon}, y_{polygon}) \] \( (19) \)

Function[distance] = get obstacle distance from array(intersect array, first layer radius, second layer radius, third layer radius, fourth layer radius) \( (20) \)

Function[direction] = get obstacle direction from \( q_j \)(first \( q_j \), second \( q_j \)) \( (21) \)

\[ LS(R_i, q_j) = (\text{distance}, \text{direction}) \] \( (22) \)

\[ LS \in [0, 1] \] \( (23) \)

The minimum distance of LS occurs when the new position is placed at \( R_i(\text{min}) \) and its maximum distance happens at \( R_i(\text{max}) \). All other values of LS lie between these maximum and minimum levels. A zero value for IP indicates that the obstacle is out of the sensor’s range and a positive value shows that the obstacle is inside the sensor’s range. To determine the exact position and direction of obstacles, two rangefinder quarters regions will be used. For example, if the robot detects that the obstacle lies in the first and second quarters, this means the obstacle is in front of the robot. The second and third quarters indicate the left side of the robot, while the third and fourth quarters refer to obstacles that are behind the robot. Finally, the fourth and first quarters refer to the right side of the robot. Sometimes, the obstacle lies in one-quarter of the range finder i.e. in quarter one. In this case, the sensor gives information that the obstacle is in the right corner of the robot, and so on for all other quarters. Four linguistic variables are defined for (LS), including Close, Medium, Far and Very far.

Next, fuzzy operators \( (RS) \) and \( (FS) \) represent the distance to the right static obstacle and the front static obstacle, respectively. These operators are determined using the same
equations that were applied for the (LS) operator. However, the number of selected quarters to determine the position of obstacles is different. For example, the exact regions for (FS) are quarters 1 and 2, and for (RS) are quarters 1 and 4.

Next, the elements of the controller ((LD), (RD), and (FD)) represent the distance to the left, right, and front dynamic obstacles, respectively. These elements are determined using the same equations that were applied for the static obstacles.

The membership functions of input variables left static (LS), right static (RS), front static (FS), left dynamic (LD), right dynamic (RD), and front dynamic (FD), are the same and are designed as shown in Figure 5.

After the controller detects the obstacle’s distance to the next position and the obstacle’s status, if it is static or dynamic, it arranges these obstacles in a matrix. If the controller detects that the next position faces four static obstacles or more, it sorts the nearest three static obstacles. The same process occurs if the controller detects four dynamic obstacles or more.

This process’s output fuzzy variable is the risk for the next location. In order to assess the efficiency of the next location, the fuzzy rules are designed cautiously. This is due to the fact that the risk between static obstacles is different to those of dynamic obstacles. A reasoning system was designed with five linguistic values for this output, including Very High (VH), High (H), Normal (N), Low (L), and Very Low (VL). The membership functions and decision surfaces are shown in Figure 6. The proposed fuzzy controller’s output variable can be interpreted as follows:

\[ \text{Risk} = \text{Function} \left( \text{LS}, \text{RS}, \text{FS}, \text{LD}, \text{RD}, \text{FD} \right) \text{ and Risk } \in \left[ 0, 100 \right] \]  

We have selected trapezoidal membership functions (MF) in fuzzy logic for many reasons depending on the reference of [29]:

(1) Have simpler analytical structures.
(2) Gives the user more freedom in MF construction.
(3) Preserves the adaptiveness.
(4) Preserves the novelty.
We also have selected 5 intervals in order to (1) assess the efficiency of the next location by determining the exact dangerous situations, (2) design cautiously the fuzzy rules because the risk between static obstacles is different to those of dynamic obstacles.

To see the whole output surface of our system, that is the whole span of the output set based on the whole span of the input set, you should open up the Surface Viewer by selecting Surface from the View menu.

Upon opening the Surface Viewer, a three-dimensional curve represents the mapping from obstacle position (LD, RD, FD, LS, RS, and FS) to Risk amount. Since this is a two-input one-output case, the whole mapping can be seen in one plot. Moving beyond three dimensions overall, we start to encounter trouble displaying the results. Accordingly, the Surface Viewer is equipped with pop-up menus that let you select any two inputs and any one output for plotting.
If the risk of the next position does not exceed the maximum permitted risk ($R_{\text{max}}$), the next location will be selected as the next position for the robot and the robot navigates directly to this new configuration. Otherwise, the controller decides to change the next location depending upon the alternative solutions available.

The maximum permitted risk ($R_{\text{max}}$) is the maximum value of the risky region that the robot should avoid to prevent failure. In this paper, this value is assumed to be 76% because based on the experiments, using more than 76%, the robot could not enter into the narrow passages.

The fuzzy rules for the proposed fuzzy controller are stated in Table 1.

From the above table, we found that the risk condition is ‘very high’ of item 1 but they are ‘normal’ for items 2 and 4 of If–Then fuzzy rule. We can explain that the risk condition is very high because the robot faced three close obstacles, one of them is dynamic and the two others are static. Therefore, the robot will trap inside the dangerous region. While the risk condition of items 2 and 4 is normal because the robot faced two dynamic obstacles and other obstacles are far. Therefore, the robot can find its path safely.

According to Figure 7, the fuzzy logic controller checks the generated next step and determines the risk of choosing it as the robot’s next position based on the aforementioned fuzzy variables.

![Figure 7](image_url)  
*Figure 7.* The flowchart of the proposed algorithm.
### Table 1. If–Then fuzzy rules.

| IF | LS Operator | FS Operator | RS Operator | LD Operator | FD Operator | RD Operator | Risk       |
|----|-------------|-------------|-------------|-------------|-------------|-------------|------------|
| 1  | Very Far    | AND         | Close       | AND         | Very Far    | AND         | Close      | THEN        | Very High   |
| 2  | Very Far    | AND         | Very Far    | AND         | Close       | AND         | Far        | THEN        | Normal      |
| 3  | Very Far    | AND         | Medium      | AND         | Very Far    | AND         | Close      | THEN        | High        |
| 4  | Very Far    | AND         | Far         | AND         | Very Far    | AND         | Close      | THEN        | Normal      |
| 5  | Very Far    | AND         | Very Far    | AND         | Close       | AND         | Very Far   | AND         | Close      | THEN        | Normal      |
| 6  | Medium      | AND         | Very Far    | AND         | Very Far    | AND         | Close      | AND         | Medium      | THEN        | Very High   |
| 7  | Medium      | AND         | Very Far    | AND         | Very Far    | AND         | Close      | AND         | Very Far    | THEN        | Very High   |
| 8  | Close       | AND         | Very Far    | AND         | Close       | AND         | Close      | THEN        | Very Far    | THEN        | Very High   |
| 9  | Medium      | AND         | Very Far    | AND         | Close       | AND         | Medium     | THEN        | Very Far    | THEN        | Very High   |
| 10 | Very Far    | AND         | Very Far    | AND         | Close       | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 11 | Very Far    | AND         | Very Far    | AND         | Medium      | AND         | Very Far   | AND         | Very Far    | THEN        | Low         |
| 12 | Very Far    | AND         | Very Far    | AND         | Very Far    | AND         | Far        | AND         | Very Far    | THEN        | Very Low    |
| 13 | Very Far    | AND         | Very Far    | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | Very Low    |
| 14 | Close       | AND         | Very Far    | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | High        |
| 15 | Medium      | AND         | Very Far    | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 16 | Far         | AND         | Very Far    | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | Low         |
| 17 | Close       | AND         | Close       | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 18 | Medium      | AND         | Medium      | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 19 | Far         | AND         | Far         | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 20 | Close       | AND         | Close       | AND         | Close       | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 21 | Medium      | AND         | Medium      | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 22 | Far         | AND         | Far         | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 23 | Close       | AND         | Medium      | AND         | Close       | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 24 | Close       | AND         | Medium      | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 25 | Close       | AND         | Medium      | AND         | Far         | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 26 | Close       | AND         | Medium      | AND         | Very Far    | AND         | Very Far   | AND         | Very Far    | THEN        | Normal      |
| 27 | Very Far    | AND         | Very Far    | AND         | Medium      | AND         | Very Far   | AND         | Close      | THEN        | Normal      |
| 28 | Close       | AND         | Very Far    | AND         | Medium      | AND         | Very Far   | AND         | Far        | THEN        | High        |
5. Simulation Studies

In order to analyse the performance of the proposed algorithm, and compare it to the other related algorithms in the path planning’s field, a simulation framework was designed using Matlab R2013a with a 2.30-GHz Intel Core 7 Duo Processor. Convex, maze, narrow and concave are the four arbitrary unknown dynamic environments that were included in the simulation. As mentioned earlier, the algorithm results in different paths every time it runs, due to an arbitrary environment. Therefore, for each class of environment, the algorithm carries out 100 iterations to compute the average values for performance variables, such as path length and run time. The stimulation results in test environment 1 are shown in Figure 8. Start configurations are shown by yellow squares and goal configurations are shown by green squares. Moreover, obstacles are shown using red objects, the positions of the robot are shown using blue squares and the generated trajectory is shown using blue lines.

Figure 8. Simulation results of concave environment.
Figure 8(a) shows that the robot navigates towards the goal based on the optimal path between the start and goal positions. After each step, the robot checks the environment to ensure the safety of the route. As illustrated in Figure 8(b), which is a convex edge, when an obstacle is detected by the sensor, it records 1. Meanwhile, when no obstacle is detected by the sensor, it records 0 in the priority matrix. Then, the robot will assess whether the obstacle is dynamic or static.

Four range sensor circles are emitted from the robot to compute the intersection points between sensor circles and obstacles for two-time intervals. The obstacle is static if both the old and new positions of an obstacle are the same. As a result, the robot looks for the next step based on being near to its goal priority. The quality of the prediction procedure is checked by the fuzzy logic controller before any further processing takes place. Moreover, the fuzzy logic controller checks the next step decision by computing the collision risk for the next position. As shown in part (c), the robot will go to the next position when the controller shows that the risk of the next position is less than that permitted.

As illustrated in parts d, e and f, when the robot arrives at the local minima, in spite of the fact that the concave edge of the environment influences the created path, the robot will be successfully guided by the planner to run away from the local minima and follow the obstacle wall.

Although the risk of the next point turns to be higher in the narrow passage age part (g), but it is less than the permitted risk. This is because the controller shows that these two obstacles are static. As a result, the robot is easily guided by the algorithm between these narrow passages.

The risk value is more than the permitted risk when a dynamic obstacle part (h) is detected by the robot. This is because the three obstacles facing the robot are detected by the controller. Two of these obstacles are static and the third is dynamic. As a result, these obstacles are stored in a matrix. Moreover, the risk for the next position is computed based on these obstacles. The findings show that the risk value is more than the permitted risk. Therefore, the controller determines changing the next position based on the alternative solutions. Then, as shown in parts i and j, the robot alters its path and exits from the narrow environment.

Finally, the robot continues with its navigated route towards the destination using the same concept (parts k and l) until it arrives at the target. Figure 9 shows the risk of the next step.

Maze, concave, narrow and convex are the other four environments included in the simulation (See Figure 10).

Average performance parameter values are shown in Table 2 for the proposed approach, for 100 iterations in each arbitrary environment.

| Parameters   | Concave | Convex | Maze  | Narrow |
|--------------|---------|--------|-------|--------|
| Runtime      | Average | 4.44   | 4.36  | 8.95   | 3.50   |
|              | Std     | 0.33   | 0.35  | 0.20   | 0.13   |
| Path length  | Average | 13.14  | 13.20 | 16.96  | 9.14   |
|              | Std     | 0.60   | 0.57  | 0.49   | 0.23   |
| Failure      | 0       | 0      | 0     | 0      |
Figure 9. The changes of the output variable of the fuzzy controller (the risk of the next step) and the corresponding membership graphs.

Figure 10. Test environments were included in the simulation including convex, maze, and narrow.

All the above-mentioned simulations assumed that the obstacle’s velocities are less than or equal to the velocity of the robots. However, if the obstacle’s velocities are faster than the robot, then we should add another constraint to the algorithm of this work such as enlarge the minimum distance between the robot and the obstacles. This constraint helps the robot to avoid the obstacle at the suitable time before striking.
Based on the study’s simulation results, the proposed Multilayer Decision-Based Fuzzy Logic Model provides effective solutions for navigation problems in complex and hazardous environments (as shown in Table 2 and Figure 11). Furthermore, the resulted path lengths and runtimes show the success of the planner to navigate the robot in unknown dynamic environments. As shown in Table 3 and Figure 11, in order to support the assertion of superiority of the algorithm, a set of 4 different navigation algorithms was selected for comparison studies. Each of these algorithms was chosen cautiously to consider the different features of the proposed planner.

The multilayer decision algorithm is sensor-based; hence, its performance requires comparison with a bug algorithm because this algorithm is one of the earliest and yet
most powerful sensor-based motion planning methods [30]. A dynamic window approach [31] was selected, because it is a well-known real-time collision avoidance approach for mobile robots. To evaluate the performance of the proposed algorithm in slow time, RRT was selected [32]. Meanwhile, PRM was selected as it reduces the resulted route [33].

Figure 12 shows the performance of the proposed FLC and compare it with the other algorithms in terms of optimality.

The proposed multilayer decision-based fuzzy logic model efficiently solved the planning queries in all test environments and provides better optimality rates than other algorithms so far. However, as mentioned before, the proposed algorithm needs to evaluate the multilayer decision-based fuzzy logic controller each time the planner finds a new collision-free configuration, and therefore, the runtime of this planner is slightly higher than the original multilayer decision approach.

In the proposed algorithm, there is a significant variable that influences the total performance of the planner; including the risk. This variable will be analysed in detail in this part of the study. The risk of the created step will be used to determine whether a position should be selected as the next destination or not. This variable can be used to observe the algorithm’s performance. Based on Figure 11, all of the risks of the next positions are less than the minimum accepted risk, except for step number 80, where the risk is around 93% (maximum permitted risk is 76%). At this point, the robot is closer to the risk region for dynamic and static obstacles. As a result, the fuzzy controller chooses to alter this next step from the alternative solutions and carries on with its navigation.

Selecting the best alternative solutions’ position is based on the priority of these alternatives and the analysis of each of them. The status and position of each obstacle, concerning each alternative solution, are predicted by the fuzzy controller. The risk of all alternatives will then be analysed and calculated by the fuzzy controller based on the fuzzy rules. Next, the solutions that encompass the lowest risk from all other solutions will be selected by the fuzzy controller.
6. Conclusion

For robot path planning in unknown dynamic environments, a new sensor-based algorithm was proposed. A fuzzy controller was utilised to assess the positions generated by the planner and select better steps that decrease the overall path length, together with the failure rate of the algorithm. Moreover, it keeps the robot away from possible local minima. This controller uses the priority rules and prediction of the multilayers approach to improve the quality of the next position, with regard to safety and path length, in order to develop six fuzzy control variables.

Each time the planner creates a new collision-free configuration, fuzzy variables will be computed. They will be served as the fuzzy reasoning system’s input to decide the total risk of the next step. Dynamic path planning locates a new path from the alternative solutions to avoid risky regions through priority behaviour; when the prediction shows that the risk rate of collision with many obstacles is very high. This algorithm uses the benefits of fuzzy logic to make the multi-layer decision process more efficient and effective.

Another advantage of this approach, along with short paths, low runtimes and a zero failure rate of the proposed algorithm, is that the planner does not create a map of the environment or retain the examined region totally. The only things it needs to know are the current position and the goal position. The robot is prepared with range sensors with a 360° finite direction that obtains information from its surroundings. After an obstacle enters into the robot range, the distance between it and the robot is determined, and the orientation of the moving obstacle is predicted. Then the robot will decide to select the next step depending on the future velocity vector of the obstacle and priority constraints. The robot departs from its original path if the future trajectory of the obstacle crosses its path.

As discussed previously, the algorithm’s performance was compared against several well-known path planning approaches to decide its efficiency. The safest, smoothest and shortest paths are produced by the algorithm while any local minima are avoided.

It is recommended that for future work, the results of the simulation platform should be validated against an experimental platform to extend the field of sensor-based path planning and result in more efficient algorithms. However, this application is a more complex and difficult procedure that needs further study. There is another recommendation about using type-2 fuzzy logic for designing the planner. Type-2 FL is originally planned for systems with greater uncertainty, and using this type may result in improved solutions. We use type-1 FL because Type-2 FL is a more complex procedure that needs additional study for the localisation of the fuzzy model. Finally, our study should be enhanced to handle more difficult environments, for example avoiding smart obstacles.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributors

Farah Kamil received her B.S. and Ph.D. degrees from University of Technology, Iraq, and University Putra Malaysia, respectively. Her research areas include path planning, artificial intelligence and optimization.
Mohammed Yasser Moghrabiah received his first Bachelor Degree in the field of Informatics Engineering in 2011 from Arab International University. He pursued his Master of Science degree in the field of Intelligent Systems Engineering from University Putra Malaysia in 2015.

ORCID

Farah Kamil http://orcid.org/0000-0002-2178-4751

References

[1] Purian FK, Sadeghian E. Mobile robots path planning using ant colony optimization and Fuzzy Logic algorithms in unknown dynamic environments. Control, Automation, Robotics and Embedded Systems (CARE), 2013 International Conference on; IEEE; 2013.
[2] Dunwei G, Na G. Local robot path planning with predicting band trajectories of obstacles. Natural Computation (ICNC), 2011 Seventh International Conference on; IEEE; 2011.
[3] Zeng Z, Sammut K, Lian L, et al. A comparison of optimization techniques for AUV path planning in environments with ocean currents. Rob Auton Syst. 2016; Aug 1;82:61–72.
[4] Dongshu W, Yusheng Z, Wenjie S. Behavior-based hierarchical fuzzy control for mobile robot navigation in dynamic environment. Control and Decision Conference (CCDC), 2011 Chinese; IEEE; 2011.
[5] Phillips M, Likhachev M. Sipp: Safe interval path planning for dynamic environments. Robotics and Automation (ICRA), 2011 IEEE International Conference on; IEEE; 2011.
[6] Kamil F, Tang S, Khaksar W, et al. Obstacles avoidance mobile robot system in uncertain and ever-changing surroundings. Pert J Schol Res Rev. 2016;2:2.
[7] Mingxin Y, Sun’an W, Canyang W, et al. Hybrid ant colony and immune network algorithm based on improved APF for optimal motion planning. Robotica. 2010;28(06):833–846.
[8] Li G, Yamashita A, Asama H, et al. An efficient improved artificial potential field based regression search method for robot path planning. Mechatronics and Automation (ICMA), 2012 International Conference on; IEEE; 2012.
[9] Faisal M, Al-Mutib K, Hedjar R, et al. Multi modules fuzzy logic for mobile robots navigation and obstacle avoidance in unknown indoor dynamic environment. Proceedings of the 2013 International Conference on Systems, Control and Informatics; 2013.
[10] Masehian E, Sedighizadeh D. Classic and heuristic approaches in robot motion planning – a chronological review. World Acad Sci Eng Technol. 2007;23:101–106.
[11] Weerakoon T, Ishii K, Nassiraei AAF. An artificial potential field based mobile robot navigation method to prevent from deadlock. J Artif Intell Soft Comput Res. 2015;5(3):189–203.
[12] Algabri M, Mathkour H, Hedjar R, et al. Self-learning mobile robot navigation in unknown environment using evolutionary learning. JUCS. 2014;20(10):1459–1468.
[13] Raudonis V, Maskeliunas R. Trajectory based fuzzy controller for indoor navigation. Computational Intelligence and Informatics (CINTI), 2011 IEEE 12th International Symposium on; IEEE; 2011.
[14] Junratanasiri S, Auephanwiriyakul S, Theera-Umpon N. Navigation system of mobile robot in an uncertain environment using type-2 fuzzy modelling. Fuzzy Systems (FUZZ), 2011 IEEE International Conference on; IEEE; 2011.
[15] Chang H, Jin T. Command fusion based fuzzy controller design for moving obstacle avoidance of mobile robot. In: Future information communication technology and applications. Dordrecht: Springer; 2013. p. 905-913
[16] Mohanty PK, Parhi DR. Path generation and obstacle avoidance of an autonomous mobile robot using intelligent hybrid controller. In: Swarm, evolutionary, and memetic computing. Berlin, Heidelberg: Springer; 2012. 20; p. 240-247
[17] Gao Y, Sun S. A collision predict based local path planning of mobile robots. Informatics in Control, Automation and Robotics, 2009. CAR’09. International Asia Conference on; IEEE; 2009.
[18] Ahmed AA, Abdalla TY, Abed AA. Path planning of mobile robot using fuzzy-potential field method. Iraqi J Electr Electr Eng. 2015;11:1.

[19] Chinag C, Ding C. Robot navigation in dynamic environments using fuzzy logic and trajectory prediction table. Fuzzy Theory and Its Applications (iFUZZY), 2014 International Conference on; IEEE; 2014.

[20] O, Sepúlveda R, MurcioI, Orozco-Rosas U. Geo-navigation for a mobile robot and obstacle avoidance using fuzzy controllers. In: Recent advances on hybrid approaches for designing intelligent systems. Cham: Springer; 2014. p. 647–669.

[21] Khaksar W, Hong TS, Khaksar M, et al. A fuzzy-tabu real time controller for sampling-based motion planning in unknown environment. Appl Intell. 2014;41(3):870–886.

[22] Wang N, Su S, Han M, et al. Backpropagating constraints-based trajectory tracking control of a quadrotor with constrained actuator dynamics and complex unknowns. IEEE Trans Syst Man Cybern Syst. 2018;49(7):1322–1337.

[23] Wang N, Su S, Pan X, et al. Yaw-guided trajectory tracking control of an asymmetric underactuated surface vehicle. IEEE Trans Ind Inf. 2018;15(6):3502–3513.

[24] Zagradjanin N, Rodic A, Pamucar D, et al. Cloud-based multi-robot path planning in complex and crowded environment using fuzzy logic and online learning. Inf Technol Control . 2021;50(2):357–374.

[25] Ayub S, Singh N, Hussain MZ, et al. Hybrid approach to implement multi-robotic navigation system using neural network, fuzzy logic and bio-inspired optimization methodologies. Navigation. 2021; 1:1–19.

[26] Sangeetha V, Krishankumar R, Ravichandran KS, et al. A fuzzy gain-based dynamic ant colony optimization for path planning in dynamic environments. Symmetry. 2021;13(2):280.

[27] Kamil F, Hong TS, Khaksar W, et al. New robot navigation algorithm for arbitrary unknown dynamic environments based on future prediction and priority behavior. Expert Syst Appl. 2017;86:274–291.

[28] Ahmad NS, Boon NL, Goh P. Multi-sensor obstacle detection system via model-based state-feedback control in smart cane design for the visually challenged. IEEE Access. 2018;6: 64182–64192.

[29] Wu D. Twelve considerations in choosing between Gaussian and trapezoidal membership functions in interval type-2 fuzzy logic controllers. 2012 IEEE International Conference on Fuzzy Systems; IEEE; 2012.

[30] Lumelsky VJ, Stepanov AA. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica. 1987;2(1):403–430.

[31] Fox D, Burgard W, Thrun S. The dynamic window approach to collision avoidance. IEEE Robot Autom Mag. 1997;4(1):23–33.

[32] LaValle SM, Kuffner Jr JJ. Randomized kinodynamic planning. Int J Rob Res. 2001;20(5):378–400.

[33] Fleury S, Soueres P, Laumond J, et al. Primitives for smoothing mobile robot trajectories. IEEE Trans Rob Autom. 1995;11(3):441–448.