Novel Fuzzy Reinforcement Algorithm for Mobile Robot Navigation in Automated Storage

Chadi F. Riman * and Pierre E. Abi-Char

College of Engineering and Technology, American University of the Middle East, Kuwait
Email: chadi.riman@aum.edu.kw (C.F.R.); pierre.abichar@aum.edu.kw (P.E.A-C.)
*Corresponding author

***Abstract*—Robots are used to move stored items in Automated Storage (AS) to shelve and pickup items in warehouse depicted shelves. In this situation and many others, it is important to follow the shortest way so that the smallest time possible is taken to achieve the task. In this research, a Fuzzy Logic system with Q-Learning Reinforcement in order to achieve a better overall system. While Fuzzy Logic can be used alone for robot navigation, Reinforcement learning helps to adjust the fuzzy rules and refine them towards two main purposes: reach the final goal, while avoiding difficult obstacles such as traps. This is done as an enhancement on our previous work where Fuzzy Logic system was used alone. Simulation results are added to support the work done. It proved that this new system is much better than the previous one. Highlighting key parameters or features of simulation results show that the system achieved 33% more optimized time in addition to avoid stalled/unsuccessful navigation in some difficult situations, thus demonstrating the system's success.**

***Keywords*—fuzzy logic, fuzzy reinforcement learning, path planning, Q-Learning, robotics**

I. Introduction

As part of the Industry 4.0 standard, Automated Storage (AS) retrieval systems are used in many warehouses in big companies around the world. They use robots for handling and fetching items because they are faster, more practical, and save human resources. The mobile robot navigation in AS is very important. It is the base of an automated storage. Furthermore, it is crucial in robotics to navigate through the minimum possible path because it saves time and money. Many shortest-path algorithms are used such as Dijkstra, A*, BFS, and even reinforcement learning. A* is an enhanced version of Dijkstra, using a heuristic function H to have a better assumption for the current situation [1]. According to Permana *et al.* [1], A* proved better than Dijkstra and Breadth First Search (BFS) when tested on Maze Runner Game.

Reinforcement Learning (RL) is a way to improve navigation by training data of a known environment. It is mainly used in gaming and control problems.

Fuzzy logic is used for decision systems when the outcome is uncertain. It is used in many areas such as control, image processing, AI, and also in robotics. Fuzzy logic better reflects real-world problems than classical logic and fuzzy algorithms can produce accurate results with imprecise data. However, they require validation, and depend on human knowledge.

Fuzzy reinforcement learning can be added to fuzzy systems in order to improve with training and time the performance of implemented fuzzy rules. Through the perception and interpretation of its environment, it adds extra weight to each fuzzy rule positively acting to reach the goal, and hinders rules that deviate from the optimum path. The reinforcement helps the robot to adjust its navigation in order to improve performance. RL is an automatic modification of the robot behavior in its navigation environment [2]. Therefore, RL is a way for optimizing control, when the system starts from an deficient solution which slowly improves according to the training done to solve the navigation problem [2]. By observing the results of every action, the agent is able to maximize the rewards and achieve a better performance [2].

In this work, we suggest a model to include fuzzy Q-Learning reinforcement to our existing fuzzy model. This reinforcement will improve the operation of our fuzzy rules in order to better achieve reaching the goal while avoiding obstacles, and escaping traps on the way. Our suggested model will enhance robot's navigation in AS and reduce time necessary to fetch and store goods in the storage.

The rest of this paper is arranged as follows. In Section II, we briefly survey the relevant existing work and compare the existing systems in terms of number of fuzzy systems, usage of reinforcement learning and others. In Section III, the mathematical background of Fuzzy Logic and Fuzzy Reinforcement Learning are defined. In Section IV, the reinforcement learning algorithm requirements with the training/testing phases are explained. In Section V, we show the implementation and testing of the reinforcement model. In Section VI, we compare the fuzzy reinforcement model with our previous work with multi-system fuzzy model without reinforcement. In Section VII, we compare our model with another existing multi-system fuzzy reinforcement model. Finally in

Manuscript received November 8, 2023; revised December 17, 2023; accepted January 3, 2024; published April 17, 2024.

Section VIII, we end the paper and present the potential future work.

II. Literature Review

The main goal in mobile robots is to arrive at the goal using the smallest possible track, and at the same time not be trapped concave obstacles (called cul-de-sac), or hit walls, or be stuck in a certain location not able to decide on a direction. Obstacles can be either static or dynamic such as other robots on the way. Several solutions using fuzzy logic systems exist: from single inference systems, or multiple systems that are run according to the current situation. There can up to more than 3 fuzzy controllers at the same time. A few of this use reinforcement learning in a way to enhance the overall system's capability, after a training period. Although reinforcement learning enhances these rules, however the fuzzy sets should be selected in a wise way from the beginning in order to achieve a good overall system. Systems not using reinforcement, were selected because of their importance handling situations where training is not possible or the overall environment is not known in advance.

Several papers were studied and compared together. Among these systems, several used one fuzzy controller that handles all situations [2–16], others used two fuzzy controllers [17–27], and a few used three fuzzy controllers [28–39]. Furthermore, a few of the papers [2, 3, 17, 21, 24, 26, 29, 37, 39] used reinforcement learning in order to improve on the fuzzy system. A list of the studied papers is shown in Table I next, ordered by year of publication.

TABLE I. Reviewed Systems with Fuzzy Controllers

Reference	Year	Number of Fuzzy Controllers	Includes Reinforcement
[37]	≥ 2020	1	Yes
[39]	≥ 2020	3	Yes
[6, 12, 13]	≥ 2020	1	No
[38]	≥ 2020	2	No
[36]	≥ 2020	3	No
[5, 9, 10, 14–16]	2017–2019	1	No
[25]	2017–2019	2	No
[18–23, 27]	2010–2016	2	No
[21, 24]	2010–2016	2	Yes
[30]	2010–2016	3	No
[29]	2010–2016	3	Yes
[28, 31]	2010–2016	4	No
[4, 7, 8]	< 2010	1	No
[2, 3]	< 2010	1	Yes
[17, 26]	< 2010	2	Yes

In our previous work [36], we presented a Fuzzy control system for robotics' movement in AS. Our system detected neighboring obstacles using sensors, and included 3 sub-controllers: Reach Goal, Avoid Walls, and Escape Cul-De-Sac. The fuzzy system proved to be effective on a small map, and for simple navigations with few obstacles on the way. But it did not give an optimal path in several situations.

Comparing the existing systems with reinforcement learning, in Table II a list of the explained fuzzy with reinforcement learning systems is shown with indication whether they are using single, dual, or triple fuzzy controllers, including a short description of each of the controllers.

TABLE II. Fuzzy Systems with Reinforcement Learning

Reference	First Controller	Second Controller	Third Controller
[2]	Target locate and safe move	–	–
[3]	Path navigation and obstacle avoidance	–	–
[17]	Follow Target and Avoid Obstacles	Escape concave trap	–
[21]	Goal seeking	Wall following	–
[24, 26]	Goal seeking	Obstacle avoidance	–
[29]	Goal seeking	Wall following	Obstacle avoidance
[37]	Obstacle avoidance	–	–
[39]	Goal seeking	Obstacle avoidance	Wall following

The reviewed systems using dual fuzzy controllers are listed in Table III. Table III is divided into 3 columns, the first showing the work reference, the second listing the name of the first used controller, and the third showing the name of the second used controller.

TABLE III. Systems with Dual Fuzzy Controllers

Reference	First Controller	Second Controller
[17]	Follow Target and Avoid Obstacles	Escape concave trap
[18]	Angular velocity	Linear velocity
[19–26]	Goal reaching / seeking	Obstacle avoidance
[21]	Goal seeking	Wall following
[22]	Orientation	Obstacle avoidance
[27]	Obstacle avoidance	Join Virtual Target

Furthermore, the checked systems using three or more controllers are shown in Table IV. Table IV has similar divisions to Table I, with an extra column showing the name of the third used controller. The last column shows the name of the fourth controller, if it exists.

TABLE IV. Systems with Triple or More Fuzzy Controllers

Reference	First Controller	Second Controller	Third Controller	Fourth Controller
[28]	Reach target	Avoid obstacle	Escape trap: right wall follow	Escape trap: left wall follow
[29]	Goal seeking	Wall following	Obstacle avoidance	–
[30]	Linear velocity	Angular velocity	Obstacle avoidance	–
[31]	Go to target	Avoid obstacle	Wall follow	Wander
[36]	Reach Goal	Avoid Walls	Escape Cul-De-Sac	–
[39]	Goal seeking	Obstacle avoidance	Wall following	–

III. Mathematical Background

A. Fuzzy Logic Definition

Fuzzy logic is a mathematical model built on the notion of “degree of membership” rather on the usual true/false (1 or 0) binary logic on which modern computers are based. A fuzzy model or set is a computational way to show fuzzy or vague data. It is similar to how human minds work. Fuzzy logic is in a way how thinking works. It has a degree of truth going from 0 to 1 as a mathematical model of vagueness [32]. Fuzzy Systems are composed of three parts [33]: Fuzzification, Fuzzy Rules Evaluation (or Inference Mechanism), and Defuzzification. A basic fuzzy system is shown in Fig. 1.

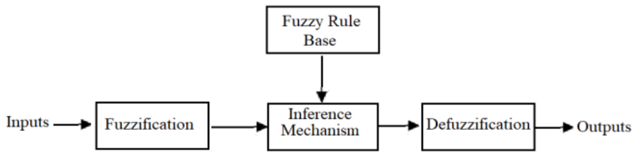


Fig. 1. Basic fuzzy system.

The inference mechanism contains a number of If-Then rules which applies a fuzzy logic quantification of the description of how to achieve a good output. These If-Then rules are aggregated together for each output variable, as shown in Fig. 2.

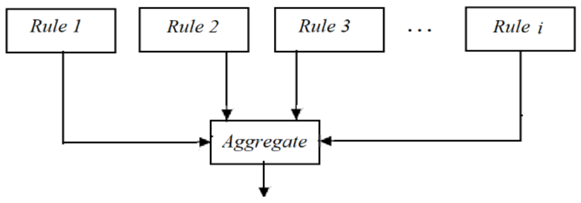


Fig. 2. Fuzzy rules aggregation.

B. Reinforcement Learning Definition

Reinforcement Learning contains an environment and its agent working within that environment to achieve some goals such as reach a target for a robot. The agent reads the environment, and executes an action. The state the environment then goes to a new state, and the agent gets a reward that is an indicator on the correctness of the followed action. The agent’s goal is to get the most possible rewards during an experiment. An experiment starts from a certain zero stage, and keeps changing until the agent reaches a final stage [34]. The Reinforcement Learning is based on the Markov Decision Process.

An infinite horizon, discounted Markov Decision Process (MDP) is defined by M = $(S, A, P, r, \gamma, \mu)$ [35]: where

- S: a state space.
- A: a discrete action space.
- P: a transition function $S \times A \rightarrow \Delta(S)$
- r: reward function $S \times A \rightarrow [0, 1]$
- γ: discount factor $\in [0,1]$
- μ: initial state distribution $\in \Delta(S)$

Q-Learning is a reinforcement learning technique to learn the value of an action $\mathbf{a_t}$ in a particular state $\mathbf{S_t}$. It is used for solving MDP that models realistic problems. It forecasts the overall future discounted reward that will be received from current action $\mathbf{a_t}$. It has three functions: an evaluation function, a reinforcement function and an update function. The update function is based on the equation: [32]

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_t + \gamma.max_a Q(s_{t+1},a_t) - Q(s_t,a_t)] \quad (1)$$

where alpha $\alpha \in [0,1]$ is a small learning rate constant. Gamma γ is the discount rate applied to the maximum Q-value of next state $\max_a Q(s_{t+1},a_t)$. r_t is the immediate reward. The Q-function has a Q-table, where each cell corresponds to a state-action value pair value. This Q-table contains the result of the learning reinforcement after several episodes.

C. Fuzzy Reinforcement Learning

A Fuzzy Reinforcement Learning applies the training reward as a factor to be multiplied by the fuzzy rules contributing to the decision made by the agent. Initially the factor is filled with an initial training iteration (explained in next section), which applies a direct reward reinforcement. The reward can slightly increase or

decrease the factor after each cell movement action made by the agent during the training period. After finishing the training period, each fuzzy rule will end up with a fixed multiplying factor ***M*** that affects its contribution to the end result. ***M*** is the accumulation of all the rewards for rules applied for a particular action in a specific situation. This relies on the Q-table which contains the state/action/reward combinations. The state being the position and the direction.

$$\begin{gathered}(1+M_1)/2 \times \textit{Action } 1\\ (1+M_2)/2 \times \textit{Action } 2\\ \ldots\\ (1+Mi)/2 \times \textit{Action } i \end{gathered} \quad (2)$$

IV. Suggested Fuzzy Reinforcement Model

A. Model Assumptions/Requirements

Path planning solution can be thought of as a connected graph (undirected) G = (N, E, L), N being the number of vertices, E the edges, and L the actual length between 2 nodes. As an example, in Fig. 3, the starting point is a red circle, the target node is a violet circle, and the obstacles are noted as green squares. The assumptions/requirements are the following:

- Only one robot can fit on every block.
- All mobile robots move at the same speed of 1m/s (assumed).
- Every square is 1 meter wide by length. One second is needed by the robot to move for one square.
- A robot can move on straight lines only, that are marked with numbers as in Fig. 3. Check Fig. 4 for the movement directions.

- Mobile robots direction is not considered. The robot can move in any direction.
- The main robot has a knowledge of the environment with the obstacle locations, which is needed to apply learning reinforcement technique.

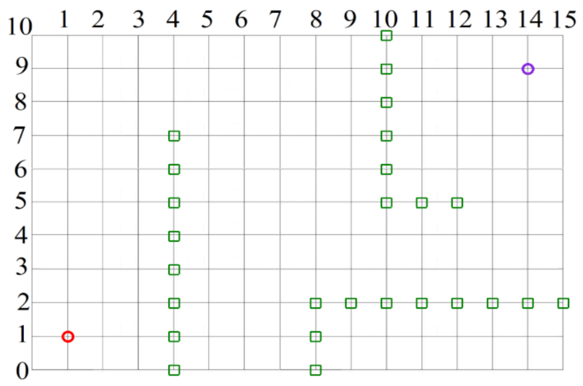


Fig. 3. Environment grid used.

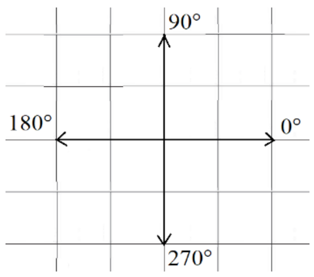


Fig. 4. Robot's movement directions.

B. *Initial Fuzzy Model before Reinforcement*

The proposed initial Fuzzy model is a single fuzzy system to reach goal while avoiding obstacle. It guides the robot to its goal, avoiding obstacles on the way until success or getting stuck in a specific position. The current and target positions are both taken into consideration. The system has two axes *x* and *y*. The inputs are called *delta(x)* and delta(y), where:

delta(x)= DisX = current(x) – goal(x)
delta(y)= DisY = current(y) – goal(y)

There is also the distance from the robot to its surrounding obstacles.

Obstacle(x) – current(x) = ObX
Obstacle(y) – current(y) = ObY

For each loop run, the controller calculates the above values, then runs the fuzzy model three parts: fuzzification part, checking inference rules, and then the defuzzification part. The defuzzification is implemented using Center of Gravity method. The output is the new motion by the robot. The action is validated and then, if no issues will occur, meaning that it will not hit a wall, then it is followed. The controller stops when the robot arrives to the goal or get stuck in a position where it cannot decide on a move.

The inputs distances *DisX* and *DisY* have a fuzzy set containing 3 membership values: Negative, Zero, and Positive. The Zero is a singleton value that is on the goal. The Negative is on a shape showing how distant is the target is in the reverse direction of the robot. The Positive is on a shape showing how distant is the target from robot but on the same direction. Fig. 5 displays the explained fuzzy set.

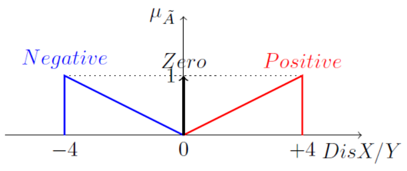


Fig. 5. Robot's movement directions.

A fuzzy set defined inputs *ObX* and *ObY*, having 3 membership values Negative/Zero/Positive. All values are singletons: −1, 0, +1. Negative means that the obstacle is one step behind the robot. Positive means that the obstacle is one step in front of the robot. Zero means that the obstacle is on the same level as the robot, obviously in only of the *X*/*Y* axes. Fig. 6 displays the explained fuzzy set.

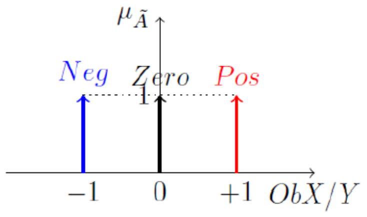


Fig. 6. Fuzzy set for inputs *ObX* and *ObY* (distance from robot to wall in *X* or *Y*).

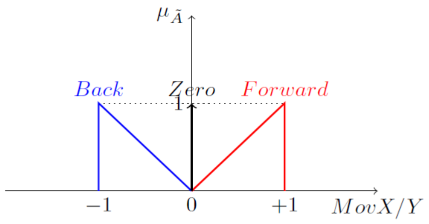


Fig. 7. Robot's movement directions.

The system outputs are *MovX*/*MovY*. *MovX*/*Y* is the decided motion in *X*/*Y* direction. These outputs are outlined with a fuzzy set having 3 memberships: Forward, Zero, and Backward. Forward is a move on the robot's direction. Backward is a move in reverse to the robot's direction. Zero is no move at all in any direction. It is a singleton of value 1. Fig. 7 reveals the output movement's set in *X* and *Y* ways.

C. *Q-Learning Reinforcement Training Phase*

For every scenario, there should be a training phase. In the scenario, the target position is fixed along with all obstacles. The training phase consists of testing all the current possible positions for the robot along with all possible actions. The possible positions along the *X* axis are from 0 to 15, and for y axis from 0 to 10. The possible actions for each position are just 4:

- Move forward along the *X* axis.
- Move backward along the *X* axis.
- Move forward along the *Y* axis.
- Move backward along the *Y* axis.

The Q-Table will be of 3 dimensions. It consists of all the *x* positions, the *y* positions, and the possible moves (left, up, right, down). The value for each cell will contain the reward to be applied. On the first iteration of the training phase, an immediate reward r_t is considered. The reinforcement factor ***M*** is equal on the first iteration to this r_t, which can have one of the following values:

$$\begin{cases} -1.0 \text{ if a collision occurs} \\ -0.5 \text{ if distance to target increases} \\ +0.5 \text{ if distance to target decreases} \\ +1.0 \text{ if target is reached} \end{cases} \tag{3}$$

where the distance to the target is given by the formula:

$$\text{distance} = \sqrt{\Delta x^2 + \Delta y^2} \tag{4}$$

A sample of possible Q-Table values is shown in the table below:

TABLE V. Q-TABLE SAMPLE DATA (PARTIAL) FOR INITIAL ITERATION

X	*Y*	Movement	r_t Reward for Reinforcement (M)
0	0	2 (X Forward)	+0.5
10	5	3 (Y Backward)	–0.5
7	3	1 (Y Forward	+1.0
3	6	0 (X Backward)	–1.0

As shown in Eq. (2), the reinforcement ***M*** is direction dependent. Supposing that there are 4 actions: *X* forward/backward, *Y* forward/backward, then the corresponding reinforcement reward is applied to the movement in the specific direction. After that, the highest action is followed. As an example, if going *X* forward gets 0.75 and going *Y* forward gets 0.50, and the others get zero, then the *X* forward motion of a full step will be applied.

After the initial iteration training phase, the Q-Table is filled with the above-mentioned values and ready to be used for the real-time testing. It is worth mentioning that if only one training iteration is done, the possible situations are only the 4 above in table V (namely M =1 or –1 or 0.5 or –0.5) because the training phase was not deep enough. In these cases, reinforcement will be one of 4 values $\frac{(1+M)}{2}$ = either 1 or 0 or 0.75 or 0.25.

In order to avoid having a dummy Q-Table with no real use except for very primitive scenarios, the learning depth (iterations) should be increased. Several iterations are run to update the table. If a cell A is adjacent to another cell B with high update function value ***Q*** (greater than 0.5) in one of its 4 direction actions, the cell A action leading to cell B will get a new update function equal to (Q–0.01).

For the further iterations, taking into consideration the Eq. (1), with alpha $\alpha = 0$, gamma $\gamma = 0.99$, the immediate reward $r_t = 0$, then the Eq. (1) will be reduced to:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_t + \gamma.max_a Q(s_{t+1},a_t) - Q(s_t,a_t)]$$
$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + 1[0 + 0.99.max_a Q(s_{t+1},a_t) - Q(s_t,a_t)]$$
$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + 0 + 0.99.max_a Q(s_{t+1},a_t) - Q(s_t,a_t)]$$
$$Q(s_t,a_t) \leftarrow 0.99.max_a Q(s_{t+1},a_t) \tag{5}$$

If the start and goal are near and the obstacles are few, a small number of iterations is required. However, if the distance becomes larger, and the obstacles are many, then a higher depth (iterations) is required. In the next section, several trials will be done to show the difference in the performance according to the training depth.

D. Fuzzy Reinforcement Model's Algorithm

In real-time scenarios, every movement is decided by a set of fuzzy rules. To apply Q-Learning reinforcement learning, the current location/direction and the decided movement for the robot is checked against the Q-Table. The ***M*** value of the reinforcement amount is used as in Eq. (2) for all possible actions for the next movement. The fuzzy rules are assessed after adding the reinforcement factor. After the defuzzification phase, the movement decision (*X* forward/backward, *Y* forward/backward) will be followed. And the same process will be repeated for next movements until target is reached. The above is described in the algorithm's flowchart in Fig. 8.

Furthermore, a more detailed Algorithm 1 is depicted next.

Algorithm 1: Fuzzy Q-Learning Reinforcement

```
1.  Choose the Begin vertex V and the Target vertex G.
2.  With present vertex i=V, Dist(i)=0, Dist is the overall moved
    distance so far from the start V.
3.  Calculate reinforcement reward Q-Table.
4.  Repeat steps from 5 to 12 to arrive to target G.
5.  Is i=G? If so, terminate loop and announce success.
6.  Calculate dist. from i to G in directions x/y:
    ΔX = x[G] – x[i]          ΔY = y[G] – y[i]
7.  Run Fuzzy Model's Fuzzification.
8.  Check Q-Table and get reinforcement M_i for all move directions.
9.  Apply reinforcement (1+M_i)/2 to all possible actions.
10. Apply Inference Mechanism.
11. Run Defuzzification.
12. Go back to step 5.
13. End.
```

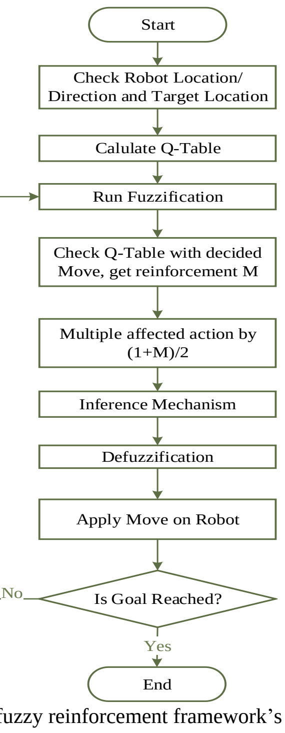

Start
Check Robot Location/ Direction and Target Location
Calulate Q-Table
Run Fuzzification
Check Q-Table with decided Move, get reinforcement M
Multiple affected action by (1+M)/2
Inference Mechanism
Defuzzification
Apply Move on Robot
Is Goal Reached?
No
Yes
End

Fig. 8. Proposed fuzzy reinforcement framework's flowchart.

In Step 3 in the above Algorithm 1, the reinforcement reward Q-Table is calculated. The calculation is done according to the specified depth. At least one iteration is done and applied according to Eqs. (3) and (4). Extra iterations are done for deeper depth according to Eq. (5). Another Algorithm 2 for calculating the Q-Table is shown next.

Algorithm 2: Q-Table calculation

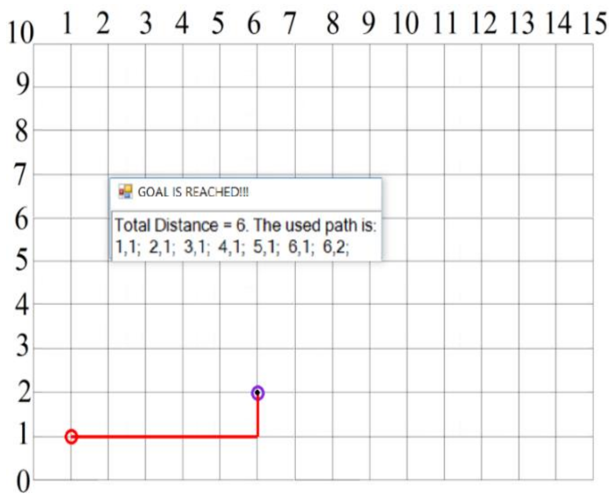
```
1.  Define Q-Table **Q(x,y,z)**: (x,y) position, z direction action (left, up,
    right, down). Initialize all Q-Table values to **zero**.
2.  Run the initial training phase, filling Q-Table according to
    equations (3) and (4).
3.  If the **depth** > **1**, repeat steps 4 to 10 for each extra iteration. Make
    **iteration = 1**
4.  If **iteration = depth**, go to step 13
5.  For each cell in **Q(x,y,z): Q(s_t,a_t)**
6.  If the neighbor cell in z direction for cell (x,y) is an obstacle, then
    make **Q(s_t,a_t) = -1**
7.  Otherwise, Get **max_a Q(s_t+1,a_t)**, the **maximum** of Q-value of next
    state, which is the neighbor cell in z direction for cell (x,y).
8.  Apply a discount rate gamma **ɣ = 0.99**.
9.  Use equation (5) to make **Q(s_t,a_t) = 0.99 max_a Q(s_t+1,a_t)**
10. Update **Q(x,y,z)** in the Q-Table.
11. **iteration = iteration + 1**
12. Repeat from step 4.
13. End.
```

V. Implementation and Testing of the Reinforcement Model

For the suggested model, a simulation was built and run to mimic the work in AS environment. It is based on the assumptions made in Section IV.A. As a limitation, the simulation is assuming no turning time for the robot, which might be not realistic in real environment. Success was checked with several scenarios, and compared to our previous work in fuzzy logic system without reinforcement [36]. The software used C# language on a standard Windows 10 OS. The performance metrics used to evaluate the new system were the time spent to achieve the goal, and whether the experiment was successful or not. The time was measured as robot movement steps assuming that each step is 1 s. If the robot was able to reach the target point, then the experiment is considered a success, otherwise it is considered as a failure.

A. *Fuzzy Experiment 1—Zero Obstacles*

A test was done without any obstacle. One simulation was performed with a start (1,1), and target (6,2). Results for the test are in Fig. 9, where the red circle indicates the source, the violet circle indicates the goal, and the red line indicates the followed path. For the simulation, the Center of Gravity defuzzification was implemented. The robot modified its movement angle at location (6,1) to reach the goal. No Reinforcement was used in this scenario. Table VI shows location vs time for the robot over six steps.



Total Distance = 6. The used path is:
1,1; 2,1; 3,1; 4,1; 5,1; 6,1; 6,2;

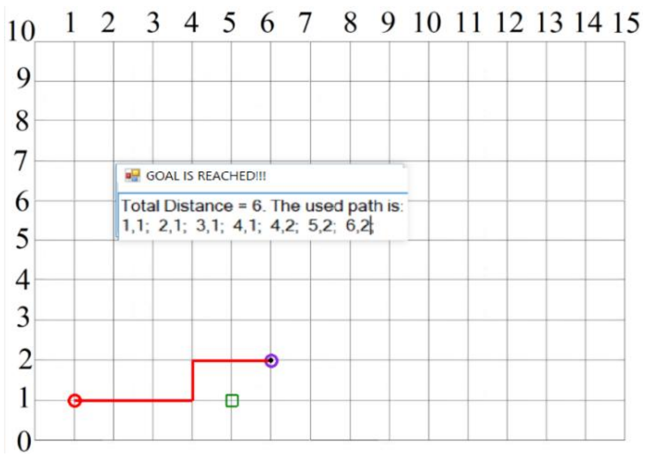
Fig. 9. Fuzzy Experiment 1—No obstacles—No reinforcement.

TABLE VI. Robot's Movements—Experiment 1

Time	Robot's Location
0	1,1
1	2,1
2	3,1
3	4,1
4	5,1
5	6,1
6	6,2

B. *Fuzzy Experiment 2—One Obstacle*

Another test was done by inserting one obstacle at location (5,1). One simulation was conducted with the starting point (1,1) and the target is (6,2). The test outputs are shown in Fig. 10. The green square indicates the obstacle. The robot modified direction at position (4,1) to avoid the obstacle, also at (4,2) to reach goal. The Fuzzy controller was active without Reinforcement. Table VII shows location vs time for the robot.



Total Distance = 6. The used path is:
1,1; 2,1; 3,1; 4,1; 4,2; 5,2; 6,2

Fig. 10. Fuzzy Experiment 2—One obstacle—No reinforcement.

TABLE VII. ROBOT'S MOVEMENTS—EXPERIMENT 2

Time	Robot's Location
0	1,1
1	2,1
2	3,1
3	4,1
4	4,2
5	5,2
6	6,2

C. *Fuzzy Experiment 3—Two Obstacles*

Another test was done by inserting two static obstacles at positions (4,2) and (5,1). Three trials were performed with the starting position (1,1) and the target is vertex (6,2). The first test was done using Fuzzy System without Reinforcement. The experiment failed because the robot got stuck in location (4,1). The second test was done using Fuzzy System with Reinforcement of depth 1. The experiment also failed because the robot was oscillating between location (4,1) and (3,1). The third test was done using Fuzzy System with Reinforcement of depth 5. The experiment was successful. The robot modified direction at position (4,1) to avoid the obstacle, then at positions (4,0) and (6,0) to reach target. The simulation results are shown in Figs. 11 and 12. Table VIII shows the location vs time for the robot, for each of the three tests.

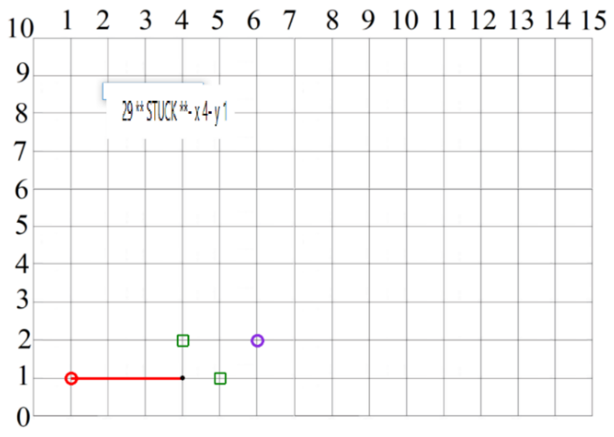
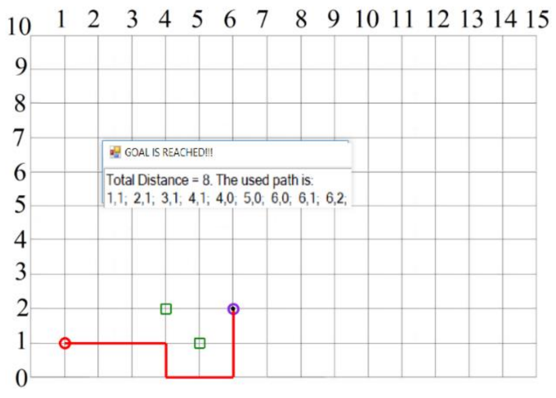


Fig. 11. Fuzzy experiment 3—Two obstacles—No reinforcement.



Total Distance = 8. The used path is:
1,1; 2,1; 3,1; 4,1; 4,0; 5,0; 6,0; 6,1; 6,2;

Fig. 12. Fuzzy Experiment 3—Two Obstacles—Reinforce: depth 5.

TABLE VIII. ROBOT'S MOVEMENTS—EXPERIMENT 3

Time	Robot Location No Reinforcement	Robot Location Reinforcement depth 1	Robot Location Reinforcement depth 5
0	1,1	1,1	1,1
1	2,1	2,1	2,1
2	3,1	3,1	3,1
3	4,1	4,1	4,1
4	4,1 (stuck)	3,1	4,0
5	–	4,1 (oscillates)	5,0
6	–	3,1	6,0
7	–	4,1	6,1
8	–	3,1	6,2

The calculation is described for location (4,1) which is the main difference between failed and sccessful attempts. With a reinforcement learning of depth 1, the movement reinforcement is just as decribed in Eqs. (3) and (4):

- Left/Down: –0.5 (distance to target increases).
- Up/Right: –1 (collision).

The system chose Left path and went back to (3,1) which has the values:

- Left/Down: –0.5 (distance to target increases).
- Up/Right: 0.5 (distance to target decreases).

where it chose the Right movement and got stuck oscillating between above 2 positions. The following table shows the reinforcement factor to each direction in the locations (3,1) and (4,1). The chosen decisions are shown in red.

TABLE IX. Q-TABLE EXPERIMENT 3—REINFORCEMENT DEPTH 1

X	Y	Move -X Reward	Move +Y Reward	Move +X Reward	Move -Y Reward
3	1	–0.5	+0.5	+0.5	–0.5
4	1	–0.5	–1.0	–1.0	–0.5

As for a reinforcement learning of depth 5 for location (4,1), the movement reinforcement becomes after repeatedly applying the change 4 times (depth 2 to 5) using Eq. (5).

$Q(s_t,a_t) \leftarrow 0.99.\max_a Q(s_{t+1},a_t)$, becomes:

- Left: 0.94. Starting from surrounding (6,2) which is +1, going back every step mutliplying by 0.99
  - ⇨ (6,1)= 0.99×1 = 0.99
  - ⇨ (6,0)= 0.99×0.99 = 0.98
  - ⇨ (5,0)= 0.99x×0.98 = 0.97
  - ⇨ (4,0)= 0.99×0.97 = 0.96
  - ⇨ (3,0)= 0.99×0.96 = 0.95
  - ⇨ (3,1)= 0.99×0.95 = 0.94
- Up/Right: –1 (collision).
- Down: 0.96. Same as above but stopping at (4,0)

It chose the Down movement.

The following table shows the reinforcement factor to each direction in the locations (4,1), (4,0), (5,0), (6,0), and (6,1). The chosen decisions are shown in red.

TABLE X. Q-TABLE EXPERIMENT 3—REINFORCEMENT DEPTH 5

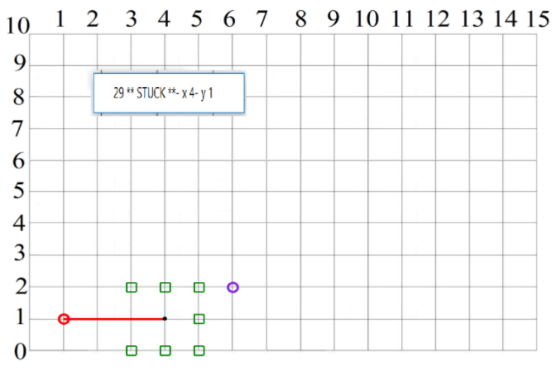
X	Y	Move -X Reward	Move +Y Reward	Move +X Reward	Move -Y Reward
4	1	0.94	–1.0	–1.0	0.96
4	0	0.95	0.95	0.97	–1.0
5	0	0.96	–1.0	0.98	–1.0
6	0	0.97	0.99	0.97	–.0
6	1	–1.0	+1.0	0.98	0.98

D. *Fuzzy Experiment 4—Cul-De-Sac*

The fourth test was performed by having seven obstacles with a shape of a Cul-De-Sac at positions (3,0), (4,0), (5,0), (5,1), (3,2), (4,2), and (5,2). Three tests were performed with the initial position (1,1) and the target is vertex (6,2). The first test was done using Fuzzy System without Reinforcement. The experiment failed because the robot got stuck in location (4,1). The second simulation was done using Fuzzy System with Reinforcement of depth 5. The experiment also failed because the robot was oscillating between location (4,1) and (3,1). The third simulation was done using Fuzzy System with Reinforcement of depth 10. The experiment was successful. The robot changed direction at position (2,1) to avoid the Cul-De-Sac, then at positions (2,3) and (6,3) to reach target. The simulation results are shown in Figs. 13 and 14. Table XI summarizes the location vs time for the robot, for each of the three tests.

TABLE XI. ROBOT'S MOVEMENTS—EXPERIMENT 4

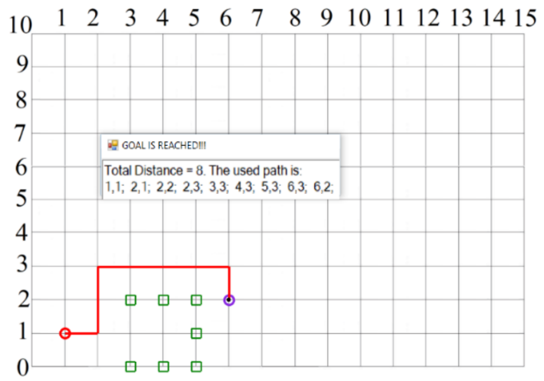
Time	Robot Location No Reinforcement	Robot Location Reinforcement depth 5	Robot Location Reinforcement depth 10
0	1,1	1,1	1,1
1	2,1	2,1	2,1
2	3,1	3,1	2,2
3	4,1	4,1	2,3
4	4,1 (stuck)	3,1	3,3
5		4,1 (oscillates)	4,3
6		3,1	5,3
7		4,1	6,3
8		3,1	6,2



29 ** STUCK **- x 4- y 1

Fig. 13. Fuzzy experiment 4—Cul-De-Sac—No reinforcement.

From the above results, a Q-Learning reinforcement of depth 5 was not enough to solve the cul-de-sac problem (Scanario 4), but it was enough to solve the previous problem with 2 obstacles (Scenario 3). As the Cul-De-Sac problem is more advanced than a simple wall of obstacles, more training iterations are needed to achieve reaching the goal. As it is seen in Fig. 14, the robot did not even enter the cul-de-sac and avoided it at the beginning, since it had prior knowledge due to extensive Q-Learning training.



GOAL IS REACHED!!!

Total Distance = 8. The used path is:
1,1; 2,1; 2,2; 2,3; 3,3; 4,3; 5,3; 6,3; 6,2;

Fig. 14. Fuzzy Experiment 4—Cul-de-sac—Reinforce: depth 10.

The calculation is described for location (4,1) for failed attempt for reinforcement with depth 5. The movement reinforcement is as decribed in Eqs. (3) and (4) for depth 1, and becomes after repeatedly applying the change 4 times (depth 2 to 5) using Eq. (5):

- Left: –0.5 (distance to target increases).
- Up/Right/Down: –1 (collision).

The system chose Left path because it has the highest value and went back to (3,1) which has the values:

- Left: –0.5 (distance to target increases).
- Right: 0.5 (distance to target decreases).
- Up/Down: –1 (collision).

where it chose the Right movement because it had the highest value and got stuck oscillating between above 2 positions. The following table shows the reinforcement factor to each direction in the locations (3,1) and (4,1). The chosen decisions are shown in red (Table XII).

TABLE XII. Q-TABLE EXPERIMENT 4 – REINFORCEMENT DEPTH 5

X	Y	Move−X Reward	Move +Y Reward	Move +X Reward	Move−Y Reward
3	1	–0.5	–1.0	+0.5	–1.0
4	1	–0.5	–1.0	–1.0	–1.0

As for a reinforcement learning of depth 10, the movement changes course at location (2,1). The reinforcement at this location becomes after the initial depth 1 and after repeatedly applying the change 9 times (depth 2 to 10) using Eq. (5).

$Q(s_t,a_t) \leftarrow 0.99.max_a Q(s_{t+1},a_t)$, becomes:

- Left/Right/Down: 0.92. Starting from surrounding (6,2) which is +1, going back every step mutliplying by 0.99
  - ⇨ (6,3)= 0.99×1 = 0.99
  - ⇨ (5,3)= 0.99×0.99 = 0.98
  - ⇨ (4,3)= 0.99×0.98 = 0.97
  - ⇨ (3,3)= 0.99×0.97 = 0.96
  - ⇨ (2,3)= 0.99×0.96 = 0.95
  - ⇨ (2,2)= 0.99×0.95 = 0.94
  - ⇨ (2,1)= 0.99×0.94 = 0.93
  - ⇨ (2,0) or (1,1) or (3,1)= 0.99×0.93 = 0.92
- Up: 0.94. Same as above but stopping at (2,2)

It chose the Up movement, because it is larger than 0.92.

The following table shows the reinforcement factor to each direction in the locations (2,1), (2,2), (2,3), (3,3), (4,3), (5,3), and (6,3). The chosen decisions are shown in red (Table XIII).

TABLE XIII. Q-TABLE EXPERIMENT 4 – REINFORCEMENT DEPTH 10

X	Y	Move−X Reward	Move +Y Reward	Move +X Reward	Move−Y Reward
2	1	0.92	0.94	0.92	0.92
2	2	0.93	0.95	−1.0	0.93
2	3	0.94	0.94	0.96	0.94
3	3	0.95	0.95	0.97	−1.0
4	3	0.96	0.96	0.98	−1.0
5	3	0.97	0.97	0.99	−1.0
6	3	0.98	0.98	0.98	+1.0

E. Comparison of the 4 Experiments

The testing done in the 4 experiments are shown and compared in Table XIV. The criteria that are used include the obstacles' count, the perfect time to arrive to the goal, the real time to arrive to the goal using the fuzzy system, and the run fuzzy system: with or without reinforcement, with the depth for the reinforcement model. Not all the experiments were successful with all fuzzy modes. In the first and second scenarios, the fuzzy model was used without reinforcement because they were both simple. The fuzzy controller was able to detect and follow the path without any issue. However, in Scenarios 3 and 4, the fuzzy model failed, and got stuck in one location. Reinforcement was applied using the Q-Learning table in both. A depth of 5 was sufficient to make the scenario 3 succeed because of its simplicity. But this depth was not enough for scenario 4 because of its complex nature, that is the cul-de-sac is not as simple as straight obstacles. Reinforcement with Q-Learning table of depth 10 achieved the navigation towards the target. In all successful cases, the actual time was optimal, equal to the ideal time. The in advance learning of the environment helped achieving an optimal time in Scenarios 3 and 4, whereby the simplicity of Scenarios 1 and 2 did not need any extra effort to have an optimal actual trial time.

F. Implementation on an Actual Robot in AS Environment

The testing done was so far in a simulation environment. The implementation of the new system on an actual robot in AS environment will include several challenges. One of them is the size of the robot is bigger than a dot like in the software. Another issue will be changing direction of the robot before moving. Furthermore, we cannot ignore the skidding effect and we need a correction measure all the way, such as beacons on every step. These things are left for a future work to be done in real implementation.

TABLE XIV. THE 4 EXPERIMENTS' COMPARISON

Scenario	Obstacles' Count	Perfect Time	Real Time	Fuzzy System
Experiment 1: No Obstacles	0	6	6	No Reinforcement
Experiment 2: 1 Obstacle	1	6	6	No Reinforcement
Experiment 3: 2 Obstacles	2	8	∞ failed	No Reinforcement
Experiment 3: 2 Obstacles	2	8	8	Reinforcement depth 5
Experiment 4: Cul-De-Sac	7	8	∞ failed	No Reinforcement
Experiment 4: Cul-De-Sac	7	8	∞ failed	Reinforcement depth 5
Experiment 4: Cul-De-Sac	7	8	8	Reinforcement depth 10

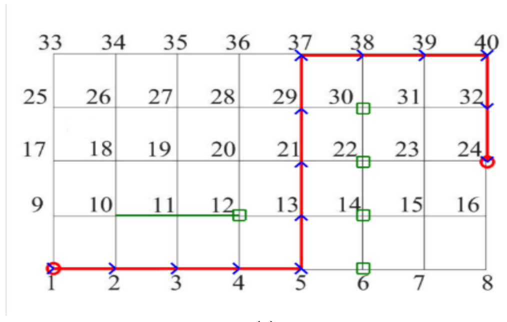
VI. CLASSICAL FUZZY VS FUZZY WITH REINFORCEMENT

The Fuzzy Model presented in our previous work [36], included 3 fuzzy sub controllers working together: Reach Goal, Avoid Obstacle, and Escape Cul-De-Sac. It proved effective on a small map, and for simple navigations with few obstacles on the way. Due to knowing only locations of the adjacent obstacles, it can be used in unknown environments. In this section, we will compare the previous system with the suggested Fuzzy Reinforcement model (learning depth = 10), using simulation on 2 scenarios applied to both systems.

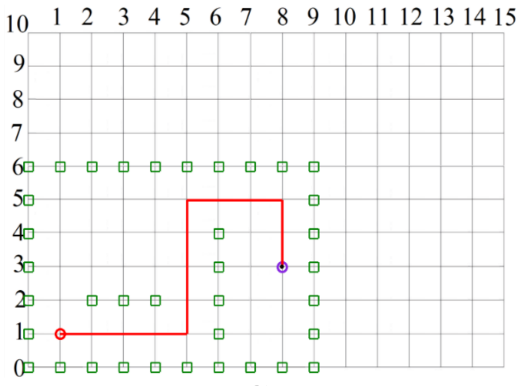
The fundamental difference between the work in [36] and the current work is the use of reinforcement learning with a training phase for the known environment. The system in [36] is able to detect the path, but not always as efficient as wanted, with the system not able in difficult scenarios to find a proper solution. The new system is always able to find the most efficient solution.

In the first scenario, the robot is moving in a small hallway surrounded with straight walls, with 90 degrees turns. Both systems managed well the situation and followed the same track. In order to have a fair comparison, the turning time/steps are ignored for the previous system. Therefore, the efficiency of both systems is the same. In Fig. 15, the followed path for both systems in scenario 1 is shown. As seen in the picture, 13 steps were needed which is the optimum path.

In the second scenario, the robot is moving in an open room with a concave obstacle (cul-de-sac) on its way. Both systems managed successfully the situation, but they followed different tracks. The efficiency of the Fuzzy Q-Learning Reinforcement system is much better because the robot avoided entering the cul-de-sac. The previous system had to enter the cul-de-sac and then leave it in search for another alternative. In Fig. 16, the followed paths for both systems in scenario 2 are shown. As seen in the picture, 8 steps were needed for the Fuzzy Q-Learning Reinforcement system which is the optimum path. The previous system needed at least 12 steps excluding the turning time/steps.



(a)



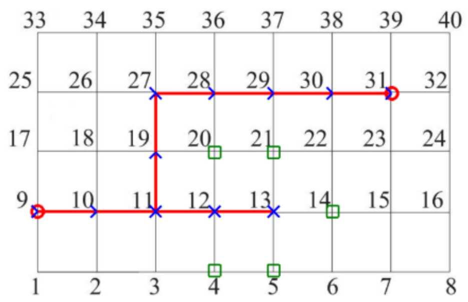
(b)

Fig. 15. Path following comparison of previous system (above) and new system (below) (a) Fuzzy alone, (b) Fuzzy with Reinforcement

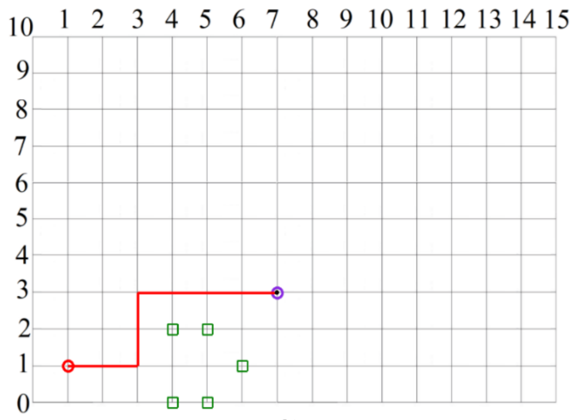
The results of applying both scenarios are compared for the two models in Table XV. As seen in the table, both systems have equal achievement with optimal time in Scenario 1. In Scenario 2, only the Fuzzy Reinforcement model achieved the optimal value, which is 33% less time than the previous fuzzy model.

TABLE XV. COMPARISON OF FUZZY MODEL WITH FUZZY REINFORCEMENT

Scenario	Obstacles' Count	Ideal Time	Actual Time	Fuzzy System
Scenario 1: Hallway with Obstacles	Many	13	13 (ignoring turns)	Old Fuzzy model
Scenario 1: Hallway with Obstacles	Many	13	13	Fuzzy with Reinforcement (Q-Learning)
Scenario 2: Cul-De-Sac	5 forming Cul-De-Sac	8	12 (ignoring turns)	Old Fuzzy model
Scenario 2: Cul-De-Sac	5 forming Cul-De-Sac	8	8	Fuzzy with Reinforcement (Q-Learning)



(a)



(b)

Fig. 16. Cul-De-Sac avoiding comparison of previous system (above) and new system (below). (a) Fuzzy alone, (b) Fuzzy with Reinforcement

VII. Our Model vs other Fuzzy with Reinforcement

The Fuzzy Model with Reinforcement Learning presented in [39], included 3 fuzzy sub controllers working together: Goal seeking, Obstacle avoidance, and Wall following. It was proven effective in several different scenarios: with or without obstacles, or with wall following. In this section, we will compare proven system with our Fuzzy Reinforcement model using simulation on a single specific scenario with obstacles applied to both systems.

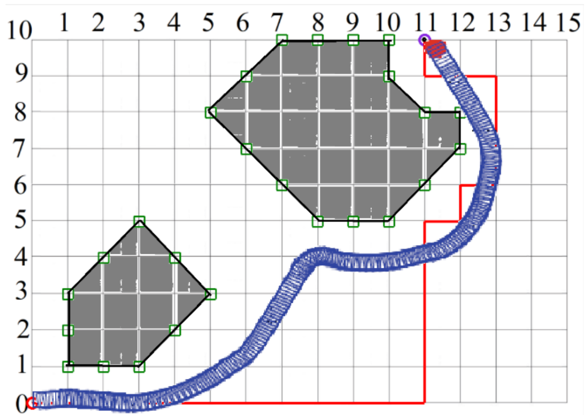


Fig. 17. Obstacle Avoidance: [39] (blue) vs proposed system (red)

In this scenario, the robot is moving in a room with two large obstacles, and the target is behind these obstacles. In Fig. 17, the followed path for both systems in the scenario is shown: the reference [39] path in blue, and our system's path in red using a training depth of 10. As seen in the picture, 25 steps were needed for our system, while the other [39] needed 20 steps if we ignore the change direction steps. System in Ref. [39] proved to be more optimum mainly because their robot was allowed to move in diagonals, while ours was not according to our set Model Assumptions/Requirements.

VIII. Conclusion and Future Work

Fuzzy logic algorithms are widely used nowadays in robotics, control, artificial intelligence and many other applications. Fuzzy logic is used as an enhancement to classical logic when ambiguities are involved. However, fuzzy logic needs a lot of expertise for good designs.

A Q-learning Reinforcement model is suggested to be applied to a fuzzy system to take care of robotic navigation in an Automated Storage (AS) archiving system. Reinforcement learning is proposed to enhance the fuzzy system sets and adjust it to respond to the requirements of the warehouse. The model is finalized and tested in a simulation environment. The results showed a success using Q-Learning Reinforcement with large training iteration depth. The new model was compared to our previous fuzzy system without reinforcement. The comparison was done on a follow path scenario and escape from cul-de-sac obstacle scenario. Those two typical scenarios were tested on both systems, with the metric used being the time in steps and second required to finish the test, and the overall success. The new system showed better performance in some scenarios especially for concave obstacles, taking 8 seconds instead of 12.33% time saving.

The new system includes a few limitations. The main one is the need to know the environment and the requirement of a training phase first. This is usually not an issue in AS, because the environment is well known, and the training can be done before the installation of a new robot. We intend in the future to build this controller on an actual robot in AS environment. This implementation will add new challenges as mentioned in Section V.F.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Chadi F. Riman and Pierre E. Abi-Char developed the theory by contributing to the design of this research. Both authors wrote the manuscript. Both authors discussed the results and commented on the manuscript. Chadi Riman performed the software simulation; Pierre AbiChar revised and refined the article. Both authors approved the final version.

References

[1] S. Permana *et al*., "Comparative analysis of pathfinding algorithms a *, Dijkstra, and BFS on maze runner game," *International Journal Of Information System and Technology*, vol. 2018.

[2] K. C. Tan, K. K. Tan, T. H. Lee, S. Zhao, and Y. J. Chen, "Autonomous robot navigation based on fuzzy sensor fusion and reinforcement learning," in *Proc. IEEE Internatinal Symposium on Intelligent Control*, 2002, pp. 182–187.

[3] C. Ye, C. Yung, N. Wang, and W. Dan, "A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance,", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, 2003.

[4] S. M. Raguraman, D. Tamilselvi, and N. Shivakumar, "Mobile robot navigation using Fuzzy logic controller," in *Proc. 2009 International Conference on Control, Automation, Communication and Energy Conservation*, 2009, pp. 1–5.

[5] N. Kumar, M. Takács and Z. Vámossy, "Robot navigation in unknown environment using fuzzy logic," in *Proc. 2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2017, pp. 000279–000284.

[6] K. Farah and M. Y. Moghrabiah, "Multilayer decision-based fuzzy logic model to navigate mobile robot in unknown dynamic environments," *Fuzzy Information and Engineering*, vol. 14, 2007.

[7] E.T. Lee, "Applying fuzzy logic to robot navigation," *Kybernetes*, vol. 24, no. 6, pp. 38–43, 1995.

[8] D. R. Parhi, "Navigation of mobile robots using a fuzzy logic controller," *J Intell Robot Syst*., vol. 42, pp. 253–273, 2005.

[9] M. Boujelben, D. Ayedi, C. Rekik, and N. Derbel, "Fuzzy logic controller for mobile robot navigation to avoid dynamic and static obstacles," in *Proc. 2017 14th International Multi-Conference on Systems, Signals and Devices (SSD)*, 2017, pp. 293–298.

[10] N. H. Singh and K. Thongam, "Mobile robot navigation using fuzzy logic in static environments," *Procedia Computer Science*, vol. 125, 2018.

[11] H. Batti, C. B. Jabeur, and H. Seddik, "Fuzzy logic controller for autonomous mobile robot navigation," in *Proc. 2019 International Conference on Control, Automation and Diagnosis (ICCAD)*, 2019, pp. 1–6.

[12] D. Babunski, J. Berisha, E. Zaev, and X. Bajrami, "Application of fuzzy logic and PID controller for mobile robot navigation," in *Proc. 2020 9th Mediterranean Conference on Embedded Computing (MECO)*, 2020, pp. 1–4.

[13] J. T. Huang and C. K. Chiu, "Adaptive fuzzy sliding mode control of omnidirectional mobile robots with prescribed performance," *Processes*, vol. 9, 2021.

[14] L. A. Dias, R. W. D. O. Silva, P. C. D. S. Emanuel, A. F. Filho, and R. T. Bento, "Application of the fuzzy logic for the development of autonomous robot with obstacles deviation," *International Journal of Control, Automation and Systems*, vol. 16, no. 2, pp. 823–833, 2018.

[15] A. Pandey and D. R. Parhi, "Optimum path planning of mobile robot in unknown static and dynamic environments using Fuzzy-wind driven optimization algorithm," *Defence Technology*, vol. 13, no. 1, 2017.

[16] A. K. Rath, D. R. Parhi, H. C. Das, M. K. Muni and P. B. Kumar, "Analysis and use of fuzzy intelligent technique for navigation of humanoid robot in obstacle prone zone," *Defence Technology*, vol. 14, no. 6, 2018.

[17] Y. Duan and X. Hexu, "Fuzzy reinforcement learning and its application in robot navigation," in *Proc. 2005 International Conference on Machine Learning and Cybernetics*, 2005, pp. 899–904.

[18] E. Ayari, S. Hadouaj, and K. Ghedira, "A fuzzy logic method for autonomous robot navigation in dynamic and uncertain environment composed with complex traps," in *Proc. 2010 Fifth International Multi-conference on Computing in the Global Information Technology*, 2010, pp. 18–23.

[19] Y. Najah, C. Rekik, M. Jallouli, and N. Derbel, "Optimized fuzzy controller for mobile robot navigation in a cluttered environment," in *Proc. 2010 7th International Multi- Conference on Systems, Signals and Devices*, vol. 2, 2010.

[20] M. Faisal *et al*., "Fuzzy logic navigation and obstacle avoidance by a mobile robot in an unknown dynamic environment," *International Journal of Advanced Robotic Systems*, vol. 2, 2013.

[21] C. Lakhmissi and M. Boumehraz, "Intelligent systems based on reinforcement learning and fuzzy logic approaches," vol. 3, *Application to Mobile Robotic*, 2012.

[22] J. Johnson and D. J. Godwin, "Indoor navigation of mobile robot using fuzzy logic controller," in *Proc. 2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, 2015, pp. 1–7.

[23] H. Tang *et al*., "Application of fuzzy logic in mobile robot navigation," *Fuzzy Logic-Controls, Concepts, Theories and Applications*, pp. 1–21, 2012.

[24] M. Nadour *et al*., "Mobile robot visual navigation based on fuzzy logic and optical flow approaches," *Int J Syst Assur Eng Manag*, vol. 10, pp. 1654–1667, 2019.

[25] F. Fathinezhad, V. Derhami, and M. Rezaeian, "Supervised fuzzy reinforcement learning for robot navigation," *Applied Soft Computing*, vol. 40, 2016, pp. 33–41.

[26] F. Abdessemed, K. Benmahammed and E. Monacelli, "A fuzzy-based reactive controller for a non-holonomic mobile robot," *Robotics and Autonomous Systems*, vol. 47, no. 1, 2004.

[27] A. Karray, M. Njah, M. Feki, and M. Jallouli, "Intelligent mobile manipulator navigation using hybrid adaptive-fuzzy controller," *Computers and Electrical Engineering*, vol. 56, 2016.

[28] S. A. L. E. Teleity, Z. B. Nossair, H. M. A. K. Mansour, and A. TagElDein, "Fuzzy logic control of an autonomous mobile robot," in *Proc. 2011 16th International Conference on Methods & Models in Automation & Robotics*, 2011, pp. 188–193.

[29] C. Lakhmissi and M. Boumehraz, "Fuzzy logic and reinforcement learning based approaches for mobile robot navigation in unknown environment," *Mediterranean Journal of Measurement and Control*, vol. 9, pp. 109–117, 2013.

[30] M. S. Masmoudi, N. Krichen, M. Masmoudi, and N. Derbel, "Fuzzy logic controllers design for omnidirectional mobile robot navigation," *Applied Soft Computing*, vol. 49, 2016, pp. 901–919.

[31] P. Nattharith and M. S. Güzel, "Machine vision and fuzzy logic-based navigation control of a goal-oriented mobile robot," *Adaptive Behavior*, vol. 24, no. 3, 2016.

[32] L. Zadeh, "Fuzzy sets," *Inf. Control*, 1965, vol. 8, pp. 338–353.

[33] F. Dernoncourt, *Introduction to Fuzzy Logic; Massachusetts Institute of Technology*, Cambridge, MA, USA, 2013

[34] A. Kwiatkowski *et al*., "A survey on reinforcement learning methods in character animation,' *Euro Graphics*, vol. 41, 2022.

[35] A. Agarwal, N. Jiang, and S. M. Kakade, "Reinforcement learning: Theory and algorithms," CS Dept., UW Seattle, 2019.

[36] C. F. Riman and P. E. A. Char, "Fuzzy logic control for mobile robot navigation in automated storage," *International Journal of Mechanical Engineering and Robotics Research*, vol. 12, no. 5, pp. 313–323, 2023.

[37] E. Avelar, O. Castillo, and J. Soria, "Fuzzy logic controller with fuzzylab python library and the robot operating system for autonomous mobile robot navigation," *Journal of Automation, Mobile Robotics and Intelligent Systems*, vol. 14, 2020.

[38] M. Sabrina *et al*., "Real-time fuzzy-PID for mobile robot control and vision-based obstacle avoidance," *International Journal of Service Science, Management, Engineering, and Technology*, vol. 13, 2020.

[39] L. Cherroun *et al*., "Mobile robot path planning based on optimized fuzzy logic controllers," *New Developments and Advances in Robot Control. Studies in Systems, Decision and Control,* vol 175, 2019.