Supervised Fuzzy Reinforcement Learning for Robot Navigation

Fatemeh Fathinezhad, Vali Derhami*, Mehdi Rezaeian

Electrical and Computer Engineering Department, Yazd University

Yazd, Iran

[E-mails: fateme.fathinezhad@stu.yazd.ac.ir][E-mails:{vderhami, mrezaeian}@yazd.ac.ir]

This paper addresses a new method for combination of supervised learning and Reinforcement Learning (RL). Applying supervised learning in robot navigation encounters serious challenges such as inconsistent and noisy data, difficulty for gathering training data, and high error in training data. RL capabilities such as training only by one evaluation scalar signal, and high degree of exploration have encouraged researchers to use RL in robot navigation problem. However, RL algorithms are time consuming as well as suffer from high failure rate in the training phase. Here, we propose Supervised Fuzzy Sarsa Learning (SFSL) as a novel idea for utilizing advantages of both supervised and reinforcement learning algorithms. A zero order Takagi-Sugeno fuzzy controller with some candidate actions for each rule is considered as the main module of robot's controller. The aim of training is to find the best action for each fuzzy rule. In the first step, a human supervisor drives an E-puck robot within the environment and the training data are gathered. In the second step as a hard tuning, the training data are used for initializing the value (worth) of each candidate action in the fuzzy rules. Afterwards, the fuzzy Sarsa learning module, as a critic-only based fuzzy reinforcement learner, fine tunes the parameters of conclusion parts of the fuzzy controller online. The proposed algorithm is used for driving E-puck robot in the environment with obstacles. The experiment results show that the proposed approach decreases the learning time and the number of failures; also it improves the quality of the robot's motion in the testing environments.

Keywords: Robot navigation, Supervised learning, Reinforcement learning, Fuzzy systems.

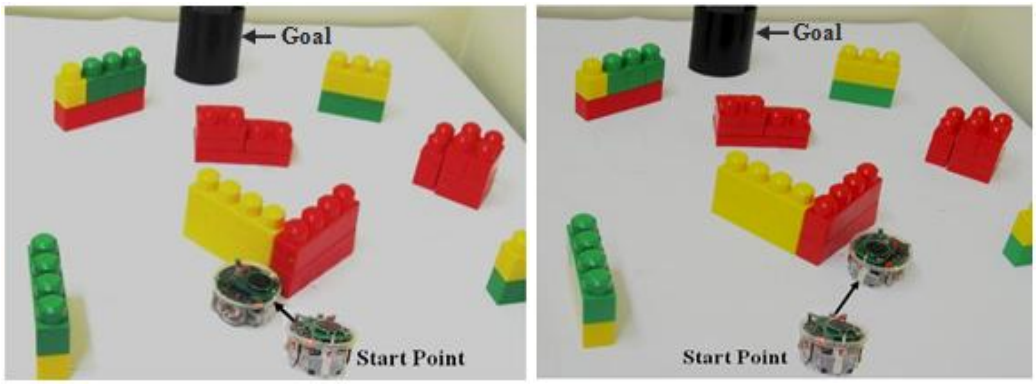
1. INTRODUCTION

The main goal in robot navigation is directing the robot to move from a starting point to a target point without hitting obstacles [1, 2]. In dynamic and uncertain environments, applying global search algorithms do not suffice to address robot navigation due to the lack of a complete model or the map of environment. Researchers have used the local search and the local path planning algorithms with help of data obtained from robot's sensors such as sonar and infrared devices [3].

Learning methods are widely used for tuning local controllers' parameters. Supervised learning is one of the first methods that have been used for training robot controllers [4,5]. In a supervised learning method, a supervisor drives robot through environment and in each time step, the supervisor command and corresponding sensors' data are gathered. Then, the collected data are used for adjusting the controller's parameters based on gradient descent methods [6-8].

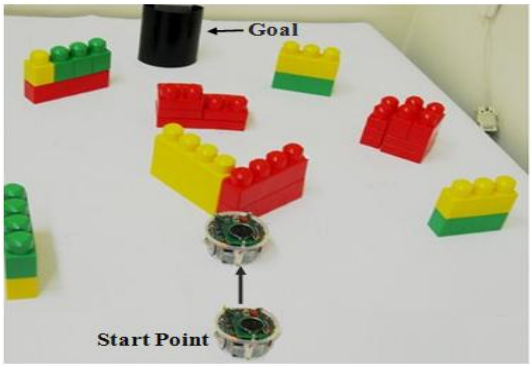
However, supervised learning in robot navigation has serious drawbacks as follows:

1- In complex situations, a supervisor cannot make right decisions, so the wrong decisions can cause significant errors in the collected (training) data.
2- Since the supervisor tries to drive the robot in the best way, many of the states may not be explored (visited).
3- The inconsistency in data causes significant errors: For example, assume that a robot is in front of an obstacle and can avoid the obstacle either from the left or the right. In such a condition, a human supervisor may decide to turn robot to the right (rotation angle $+45^o$) or turn to the left (rotation angle -45^o). Different decisions in similar situations cause big disturbance in the training procedure. Since the objective function for a training procedure is to minimize the sum of squared errors, a value close to zero (i.e. forward movement) will be obtained by gradient descent methods for this condition. This causes (or may cause) a collision with an obstacle (See Figure 1).



A: Rotate to left

B: Rotate to right



C: Moving forward and obstacle collision

FIGURE 1.Obstacle collision because of data inconsistency (The black cylinder is the goal and the colored cubes are obstacles)

Considering mentioned shortcomings in supervised learning, the interactive learning algorithms are widely used in robot navigation [2, 9, 10, 11]. Reinforcement Learning (RL) is a modern powerful interactive algorithm that can learn only by trial and error and delayed reward. RL capabilities, such as no need for desired outputs, training based on a scalar reinforcement signal, the possibility of online interactive training and high degree of exploration, encourage researchers to use RL in robot navigation problem. Due to large dimension of the discrete state-action pairs, continuous RL algorithms such as fuzzy RL (FRL) are usually employed to overcome the curse of dimensionality [2, 12,13].

Here, we focus on continuous RL algorithms based on critic-only architecture. Critic-only is a known architecture in RL employed in Fuzzy Sarsa Learning (FSL) [14] and Fuzzy Q-Learning (FQL) algorithms [15]. These algorithms present a solution for tuning the conclusion parts of rules in fuzzy inference systems.

RL algorithms are often time consuming and slow in training procedure. Since the state space in robot navigation is large, this leads to long learning time. Moreover, in the beginning of training phase, the robot does not have any knowledge, so the number of failures (punishments) would be high. These failures may cause to damage the robot in practical experiments. Our motivation in this paper is to combine “supervised learning” with “fuzzy

reinforcement learning", so that we decrease learning time and the number of failures in RL as well as prevent weaknesses of supervised learning.

Some researchers have addressed the related works in discrete space. In [16] a linear combination of supervised learning and Q-learning (one of the well known discrete RL methods) was proposed. In this approach, the selected actions by supervisor have a high chance for selection in each state during applying Q-learning. In [17], considering discrete action state space for robot navigation, a human guides the robot within the environment, when all states and actions are recorded in the defined Supervisor Table. Then, ε-greedy method is used for action selection in Q-learning, where the suggested actions from Supervisor Table have high chance to select.

In [18] supervised reinforcement learning is used for autonomous humanoid robot docking. It uses Gaussian distributed states activation so inputs can be continuous, however the action space is discrete and there are only 4 actions.

Above mentioned approaches have been proposed in the discrete (state or action) space, whereas we here focus on continuous state and action space. In [19] supervised adaptive dynamic programming was introduced for cruise control system. In there, actor- critic architecture was used for continuous RL and supervisor only guides RL process in three ways: 1) gives additional reward, 2) gives additional direct control signal to the agent; and 3) gives hints for exploration. In [2], the training data generated by the supervisor was used to determine the initial amounts of consequence parts of fuzzy rules in an actor-critic based FRL algorithm. This approach has two major weaknesses:

1- As mentioned earlier, destructive effect of inconsistency in data makes significant errors in adjustable amounts.
2- Lack of suitable exploration in actor-critic architecture [18], since the final amounts are determined around the obtained amount by training data and it causes trapping in local extrema.

In this paper, we propose a new method for combination of supervised learning and critic-only FRL algorithm. The critic-only architecture is selected since it has high potential for management of the balance between exploration and exploitation [20], which is a desired feature in robot navigation problem. The proposed idea is developed on FSL as a critic-only FRL algorithm. In our approach, instead of determining an action for each state, the value or worth of each candidate action is determined by training data. Then, for improving the performance, the final online tuning is done by FSL. The proposed combination makes the learning process faster, improves the learning quality, and reduces the number of failures (obstacle collisions). To the best of our knowledge, the proposed approach is the first work for combination of supervised learning and FRL based on critic-only architecture.

The main contributions of the paper are as follows:

1- Proposing an approach for using training (supervised) data for initialization of the value (worth) of each candidate action in critic- only based FRL architecture.
2- Designing controller using subsumption architecture [21] for robot navigation.
3- Applying the proposed method in practice for navigation of a real robot (E-Puck).

The paper is structured as follows. In Section 2, FSL algorithm is described. Design of fuzzy controller structure is presented in Section 3. In Section 4, our idea for combination of FRL and supervised learning is proposed. Section 5 presents the experimental results of applying the proposed method in a real-world application. Finally, Section 6 contains conclusion of the paper.

2. FUZZY SARSA LEARNING

FSL and FQL are two critic-only FRL algorithms. For FQL method not only there is not any theorem or lemma for convergence but also there are some divergent examples. In contrast, the existence of stationary points was established for FSL in [12]. Moreover, the experimental results in [12] signify higher learning speed and action quality for FSL compared to FQL. Therefore, we develop our idea based on FSL.

FSL is an extension of Sarsa learning (a well-known RL algorithm) [12] for continuous state and action spaces using a zero order Takagi-Sugeno (T-S) fuzzy system [6] as function approximator. In this section, we describe FSL briefly; readers can find the comprehensive information about FSL in [12].

Sarsa method estimates the value of action an in state s denoted by $Q(s,a)$ for the current policy according to the following update formula [22]:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_t [r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \tag{1}$$

where α is the learning rate, γ is the discount factor, and r_{t+1} is the immediate reward received from the environment after applying action a_t in state s_t.

Consider an n-input one-output zero order T-S fuzzy system with R rules of the following form [22]:

R_i: If x_1 is L_{i1} and … and x_n is L_{in}, then (o_{i1} with value w^{i1}) or … or (o_{im} with value w^{im})

where $s = x_1 \times \cdots \times x_n$ is the vector of n-dimensional input state, $L_i = L_{i1} \times \cdots \times L_{in}$ is the n-dimensional strictly convex and normal fuzzy set of the i-th rule with a unique center, m is the number of possible discrete actions for each rule, o_{ij} and w^{ij} are the j-th candidate action and the approximate value of the j-th action in the i-th rule, respectively. The goal of FSL is to adapt w^{ij} on-line to be used to obtain the best policy. The number of candidate actions and their amounts are fixed for the entire state space and must be determined by designer according to problem characteristics. Although the candidate actions o_{ij} in the consequence of the fuzzy rules are discrete, the final inferred action (i.e., the fuzzy system output) is a continuous action in the range of $a \in [\min_{i,j}(o_{ij}), \max_{i,j}(o_{ij})]$.

The action selection probability of the i-th candidate action in the i-th rule in state s_t is computed based on the following modified Softmax policy [22]:

$$p(a_{ij}) = \frac{\exp(\mu_i(s_t)\, w^{ij} / T)}{\sum_{k=1}^{m} \exp(\mu_i(s_t)\, w^{ik} / T)} \tag{2}$$

where $\mu_i(s_t)$ is the normalized firing strength of i-th rule for state s_t, and $T > 0$ is the temperature factor.

Notice that to calculate the overall action, first an action is selected for each rule from among the candidate actions of that rule. Denoting the selected action in i-th rule and its corresponding value by o_{ii^+} and w^{ii^+}, respectively, the system output (i.e., the overall continuous action) and its corresponding approximate Action Value Function (AVF) are computed as follows [22]:

$$a_t(s_t) = \sum_{i=1}^{R} \mu_i(s_t)\, o_{ii^+} \tag{3}$$

$$\hat{Q}_t(s_t, a_t) = \sum_{i=1}^{R} \mu_i(s_t)\, w_t^{ii^+} \tag{4}$$

Thus, the final continuous action is the weighted sum of the selected discrete actions of the rules.

Applying action a_t, the environment goes to the next state s_{t+1}, and the agent receives reinforcement signal r_{t+1}. The next final action a_{t+1} is chosen based on the present weight w_t. Then, the weight parameters of the *i*-th rule are updated by [22]:

$$\Delta w_{t+1}^{ij} = \begin{cases} \alpha_{t+1} \times \Delta \hat{Q}_t(s_t, a_t) \times \mu_i(s_t) \; if \;\; j = i^+ \\ 0 \qquad\qquad\qquad\qquad \text{otherwise} \end{cases} \tag{5}$$

where $\Delta\tilde{Q}$ is the approximate action value temporal difference error determined by:

$$\Delta \hat{Q}_t(s_t, a_t) = r_{t+1} + \gamma\, \hat{Q}_t(s_{t+1}, a_{t+1}) - \hat{Q}_t(s_t, a_t) \tag{6}$$

γ and α are the discount factor and the learning rate, respectively. As a practical matter, learning rate is gradually decreased as a function of time [7]. The algorithm procedure of FSL is summarized below:

1. Observe state s_{t+1} and receive reinforcement signal r_{t+1}.
2. Select a suitable action of each rule using modified Softmax action selection Eq. (2).
3. Compute final action a_{t+1} and the approximate AVF $\hat{Q}_t(s_{t+1}, a_{t+1})$ using Eq. (3) and Eq. (4), respectively.
4. Compute $\Delta\hat{Q}$ and update w by Eq. (6) and Eq. (5), respectively.
5. Compute new approximate AVF $\Delta\hat{Q}_{t+1}(s_{t+1}, a_{t+1})$ using Eq. (4).
6. Apply the final action.
7. $t \leftarrow t+1$ and return to step 1.

3. DESIGN OF CONTROLLER FOR ROBOT NAVIGATION

3.1 Mobile robot model

In this paper, we use an E-puck robot, a mini wheeled mobile robot, developed at the EPFL for education purposes [23-25].The E-puck robot has already been used in a wide range of applications [26,27]. E-puck is cylindrical in shape with a diameter of 70 mm and a weight of 150 g (see Figure 2). Eight infrared sensors placed around the robot measure closeness of obstacles in a 4 cm range. Readers can find the mathematical model of a wheeled mobile robot in [28], [29].

FIGURE 2. E-puck mobile robot [21]

The robot has two lateral wheels which can rotate in both directions. It is able to count the number of pulses generated by encoders which are installed on each wheel for estimating the distance traversed by each wheel. Also, depending on type of applications, we can install on auxiliary equipment such as camera, and hook [26, 27].

3.2. Controller architecture

Our objective is to design a controller to drive an E-puck from a start point to a target point without hitting obstacles. The amounts of infrared sensors indicate the approximate distance between the robot and obstacles. The outputs of infrared sensors mounted in the right, the front, and the left side are used to generate the inputs of the controller as follows:

$$S_{side}(t) = \max(sn_{side,1}, sn_{side,2}), side \in \{Left, Front, Right\} \quad (7)$$

$sn_{side,1}$ and $sn_{side,2}$ denote the outputs of the first and second sensor in the mentioned side. The output range of infrared sensors in E-puck is between 0 and 3500 (0: when the sensor does not sense any obstacle in the area, and 3500: when the sensor is almost contacted to obstacles). The fourth controller's input is the robot head angle with respect to the goal (see Figure 3), denoted by $\theta_{(t)}$, and $\theta \in [-180^o, 180^o]$. These 4 inputs are normalized and used as inputs of the controller. The controller's output is the rotation angle of the robot in the range of $[-45^o, +45^o]$. The robot's velocity is considered almost fixed in each time step.

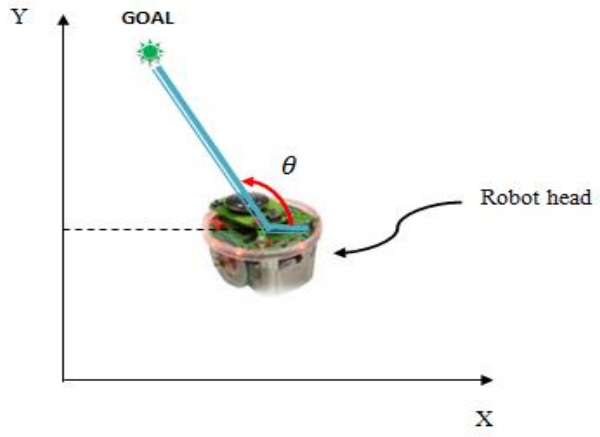


FIGURE 3.Robot angle with the goal (the star (*) shows the goal)

For decreasing complexity of controller, we utilize the subsumption architecture as a behavior–based control architecture [21,30]. The subsumption is best-known reactive robotic architecture developed by Brooks [21]. His idea was decomposing a complex task to some parallel simple tasks or behaviors. This focus on simplicity leads to a design where each individual layer operating asynchronously without any central control. In general, the different layers are not completely independent. Levels are constructed from components referred to as 'modules'; they consist of small asynchronous processors. Inputs and outputs of these modules can be inhibited or suppressed. The result is a robust and flexible robot control system.

Here two modules can be considered for the robot in environment. The first when the robot is near the obstacles, in this situation, the robot should avoid obstacles meanwhile tries to move toward the goal's position. In another module, there is not any obstacle around the robot, so it is able to rotate, whereby its head is located in the front of the goal and then the robot moves straightforward to the goal. Using subsumption architecture, we define two behaviors for the above mentioned situations: "obstacle avoidance" and "goal seeking".

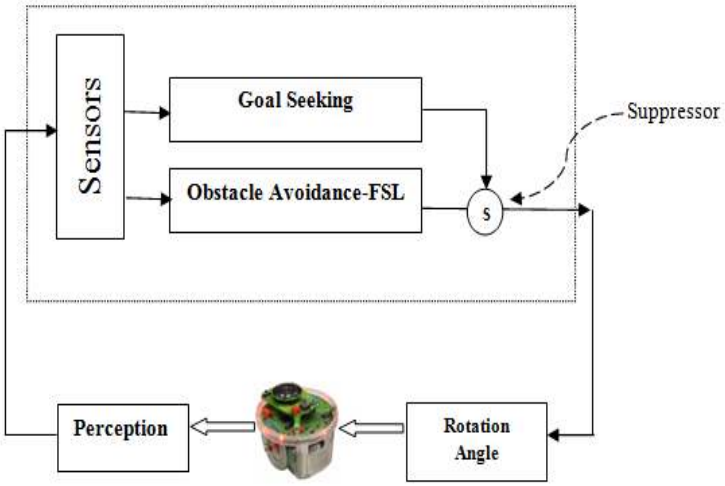


FIGURE 4. Behavior-based control schema

Figure 4 shows the designed structure of the behavior-based control system. Both modules (behaviors) receive sensor information; if the sensors do not recognize any obstacle around the robot, the module of "goal seeking" is active and its output suppresses the output of module of "obstacle avoidance"; otherwise "obstacle avoidance" module determines the angle of robot motion at each time-step for moving toward goal whereas the robot avoids obstacles. It has to be noticed that with respect to the defined tasks for modules and subsumption architecture, unlike some related works in [31-34]; in our designed structure no module is required to combine the output of modules (behaviors). Therefore, it decreases the computation cost and the system complexity.

Generating the appropriate output for "obstacle avoidance" module is a complex task; so we use a fuzzy controller for this module. For fuzzy controller, a structure compatible with FSL (introduced in Section 2) is considered. A zero order T-S fuzzy controller with four inputs and one output is suggested. The inputs include robot's distances from the obstacles in left, right and front sides (obtained by Eq. (7)) and the robot head angle with respect to the target (θ).

4. SUPERVISED FUZZY SARSA LEARNING

In this section, our proposed method that combines supervised learning and FSL is explained. In the first phase, a supervisor drives the robot in the environment and training data is collected. In contrast to the available methods

which use training data for determining an action for each state; here, a new approach to determine the worth of each candidate action in each state is presented. In fact, training data is used to initialize w^{ij} (the value of each candidate action defined in Section 2) in each rule of fuzzy controller. For example, if the supervisor selects different actions in a state in separated visits; the worth of each selected action is proportional to the number of times that action has been selected.

The output of fuzzy controller (see Eq. (3)) is the weighted sum of the selected actions of rules. So the worth of the controller's output is determined by the weighted sum of the worth of the selected actions of rules. Therefore, for determining the worth of discrete actions in rules, it is found out a set of possible candidate actions so that the computed final output would be almost close to the suggested output (by supervisor) and then, the worth of found candidate actions would be increased.

Regarding the above description, we propose the following algorithm for determining the worth (w^{ij}) of candidate actions of rules. Consider *p*-th sample of collected data as a pair of input - output (*xp, yp*) where *xp* is the input of the controller and *yp* is the suggested output by supervisor.

For the weight (w^{ij}'s) initialization, the following steps are done for each pair of data (*xp, yp*).

1- For x_p as the input of fuzzy controller, firing strengths of fuzzy rules are computed and then four dominant rules (the rules with the highest firing strength (μ)) are selected. These rules are indexed by d_l ,*l=1,2,3,4,*where $\mu_{d1} > \mu_{d2} > \mu_{d3} > \mu_{d4}$.

2- *l=1*

3- The y_p is divided by μ_{d_l} .

4- The result of division is compared with each candidate actions of the rule ($O_{d_l j}$'s). The closest action to this quotient is selected and indexed by k_l.

5- $c_{d_l k_l}$ (the amount of k_l-th action's counter in d_l-th rule) is increased one unit where C_{ij} records the number of selection of *j*-th candidate action in *i*-th rule

6- The product of the k_l-th action of the d_l-th rule and firing strength of the rule (μ_{d_l}) is computed and the result is subtracted from the action y_p : $(y'_p = y_p - \mu_{d_l} \times a_{d_l k_l})$.

7- If $l \neq 4$ then *l=l+1* and y_p is substituted by y'_p and goes to step 3.

8- Finally, when the above steps were performed for all data, the value (worth) of the *j*-th action in *i*-th rule is initialized as follows:

$$w^{ij} = \frac{c_{ij}}{\left(\sum_j c_{ij}\right)^2} \tag{8}$$

The pseudo-code of the proposed method is given in Figure 5.

Initialize counter c_{ij} for j-th candidate action in i-th rule with zero

FOR each input-output data pair (x_p, y_p)

// x_p as an input vector and y_p is the suggested output by supervisor

Compute firing strength of the fuzzy rules for x_p

Select four rules that have highest firing strength (named D_1, D_2, D_3 and D_4)

$y'_p = y_p$

WHILE (l=4)

READ rule D_l *and* μ_{dl} // μ_{dl} is firing strength of the D_l-th rule

$$a = [\frac{y'_p}{\mu_{dl}}]$$

Closest action to the division result is selected and indexed by k_l

$$C_{d_l k_l} = C_{d_l k_l} + 1$$

$$y'_p = y'_p - \mu_{dl} \times a_{d_l k_l}$$ // $a_{d_l k_l}$ is the k_l-th candidate action of the D_l-th rule

END WHILE

END FOR

FOR each rule i

FOR each candidate action j

$$w^{ij} = \frac{c_{ij}}{\left(\sum_j c_{ij}\right)^2}$$ // w^{ij} is worth of the j-th candidate action in i-th rule

END FOR

FIGURE 5.Pseudo-code of the proposed method for initialization of worth of candidate actions

In the second phase, FSL algorithm is used for online fine-tuning of conclusion parts of the fuzzy controller where its w^{ij}'s have been initialized by the proposed method. We call our proposed combination method: Supervised Fuzzy Sarsa Learning (SFSL). Figure 6 shows the block diagram of SFSL.

The total procedure of SFSL can be explained in the following steps:

1. Moving robot in the environment by supervisor, and gathering training data.
2. Initialization of the value (worth) of the candidate actions in each rule by proposed method (See Figure5).
3. Final tuning of the conclusion parts of the rules using FSL.

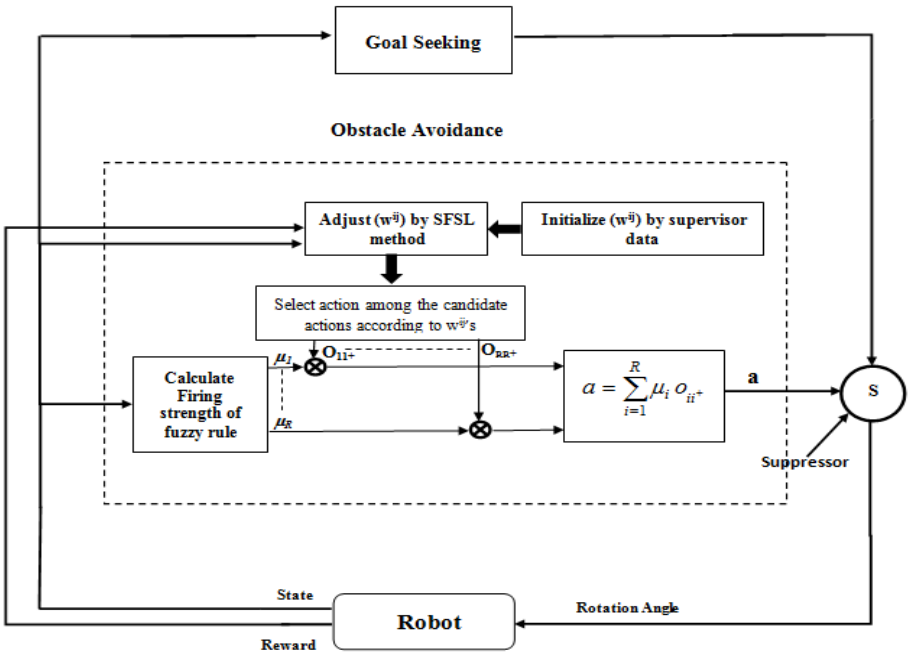


FIGURE 6. Block diagram of SFSL method (The dashed zone is the obstacle avoidance module implemted by SFSL)

5. EXPERIMENTS

The practical experiments are obtained using an E-puck robot. First, a training environment (120×80 cm^2) was prepared (see Figure 7). The color rectangles are the obstacles and the black cylinder is the goal. The subsumption architecture is used for controlling the robot. As it is stated in Section 3, the controller has 4 inputs. The main task is adjusting the parameters of the fuzzy controller used as "obstacle avoidance' module in our architecture. Figure 8 shows the defined membership functions for each input. Based on grid partitioning [7] the controller has 24 fuzzy rules.

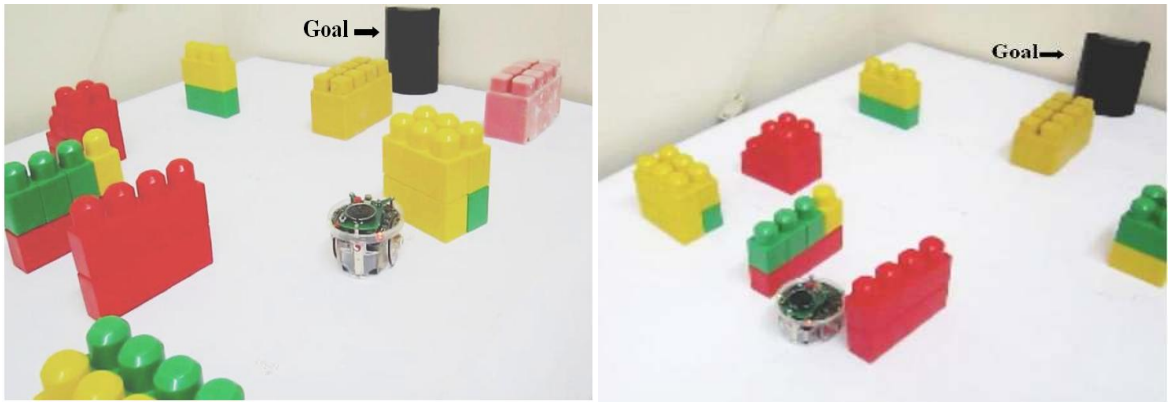


FIGURE 7.Examples of training environment for E-puck robot (The black cylinder is the goal and the colored cubes are obstacles)

The controller's output determines the rotation angle of the robot. The consequence of each rule is a constant value that must be selected from the candidate actions set, defined as:

$$A = \{-45, -30, -20, -15, -10, -5, 0, 5, 10, 15, 20, 30, 45\} \quad (9)$$

Appropriate action for each rule is determined using the proposed learning method (SFSL).

As the first step, the supervisor drove the robot in the training environment using a joystick, and 1200 data pairs were gathered. In the second step, the gathered data were used to initialize the value (worth) of candidate actions (w^{ij}) of the fuzzy controller based on the proposed method in Section 4.

Also, we used the conventional supervised learning (least squared error method) to determine the actions of each fuzzy rule similar the first phase in [2]. After this phase (as hard tuning), the obtained fuzzy controller has been applied to derive the robot. However, the performance was poor and the robot collide the obstacles much, and it could not reach the goal. It is due to inconsistency in data which was explained in introduction.

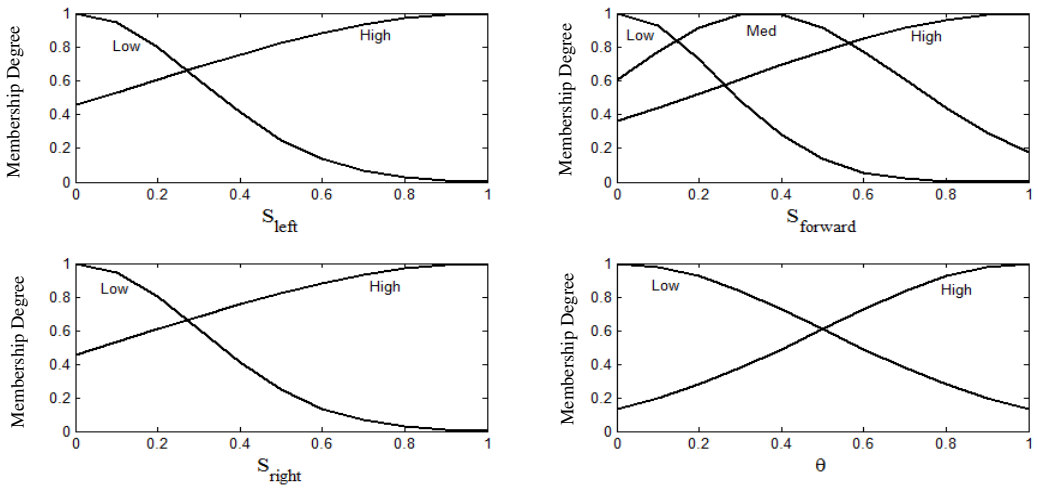


FIGURE 8.Input membership functions of the fuzzy controller

In the third step as the final tuning, FSL was used for adjusting the conclusion parts of the rules when w^{ij} 's have been initialized by the proposed method. The upper bound of learning is 500 episodes. An episode begins from a starting point and finishes when the robot reaches the goal or the number of steps in episode exceeds 300. In each episode, the starting point of the robot and the goal were determined randomly. The learning procedure is finished whenever the robot could reach goal in 10 successive non-failure episodes or the number of episodes exceeds 500. The number of episodes at the end of learning phase is called Learning Duration Index (LDI).

The reinforcement signal is defined as follows:

$$r(t)=\begin{cases}-1 & N \geq 2500 \\ -0.5 & 2500 > N \geq 2000 \\ -0.05+\Delta/100 & \Delta \leq 0 \,\&\, N < 2000 \\ \Delta/100 & \Delta > 0 \,\&\, N < 2000 \\ 1 & goal \end{cases} \quad (10)$$

$$, \quad \Delta = |\theta(t-1)| - |\theta(t)|$$

where θ is the robot head angle with respect to the goal, and N indicates that how much the robot is close to obstacles and is defined as follows:

$$N = \max(4*S_{left}, 4*S_{right}, S_{forward}) \;\; Side \in \{Left, Front, Right\} \quad (11)$$

If N is greater than 2500, a failure is considered and reinforcement signal is -1 as a punishment. If $2500 > N > 2000$, it indicates the robot is close to obtalces but it has not hitted the obstacle yet. Thus, reinforcement signal is -0.5. This punishment causes the robot learns that it moves in environemnt with an appropriate distance from obtacles. In the case of $\Delta \leq 0 \;\; \& N < 2000$, despite the robot has a suitable distance from obstacles, but it has not rotated toward the goal direction $(\Delta \leq 0)$ in previous state, so it receives a punishment proportional to Δ. In the case of $\Delta > 0 \;\; \& N < 2000$, the robot has a suitable distance from the obstacles and it has decreased its angle with respect to the goal, so it receives a reward proportional to Δ. If the distance between the robot and the goal is less than 5cm, it means that the robot reaches the goal and it receives the reward.

During the experiments, when the robot collides obstacles (failures positions); it receives the reinforcement (punishment) signal and the weights of the algorithm are updated based on Eq. (5). For getting away from collision region, the robot moves directly back about 5 cm, then it rotates randomly in the range of [-45°, + 45°]. Afterwards, the episode continues and the algorithm determines the rotation angle and the speed of the robot in the path, again.

After training, the test phase is done for assessing the quality of learning. We compare the performance of our pr oposed method (SFSL) versus FSL. The major criteria in evaluation RL algorithms are learning speed, the number of failures, and action quality in the test [2,14,17,32,33]. Here, we evaluate the algorithms by 3 measures: LDI as learning speed, "Failure Rate" as the number of failures (hitting obstacles) in the training and testing phases, and "traversed distnace" by the robot as a criterion for action quality in the testing phase. The tests are done in 2 new env ironments (see Figure. 9). To calculate the traversed distance, the total distance that the robot travels from start point to goal is considered. In practice, we count all of the encoder pulses of robot's wheels during episode and then multiply it by 0.013 where it is the related coefficient for converting encoder pulse to travelled distance of each wheel in centimeter scale.

The evaluation of the experiments is done by the average of evaluation criteria over 5 independent runs. Every run includes a "learning phase" and a "testing phase". In the learning phase for FSL, W^{ij} 's initialized to zero.

Table 1 shows the experimental results. The average LDI is computed by taking the average of 5 independent runs of each method. The "Failure Rate 1" and "Failure Rate 2" are the average number of failures during the training and testing phases, respectively. The "Avg.Distance" is the average of traversed distance by the robot within the test environments.

TABLE 1.Experiment results for robot navigation

Methods	Avg.LDI	Failure Rate 1	Failure Rate 2	Avg.Distance
SFSL	25.4	8.8	1	107.09
FSL	37.6	87.3	2	124.001

As seen in Table1, SFSL outperforms FSL, significantly. The learning speed of SFSL is 27% faster than FSL. The number of failures in training (Failure Rate1) for SFSL has been decreased 89% compared to FSL. The number of failures in the test (Failure Rate2) has been decreased, too. The robot could reach the goal in both methods in the testing phase; the traversed distance (Avg. Distances) for SFSL is 13% less than FSL. Figure 9 shows the traversed path by the E-puck robot in two testing environments for SFSL. As seen in this figure, the robot has reached to the goal, moving among the obstacles.

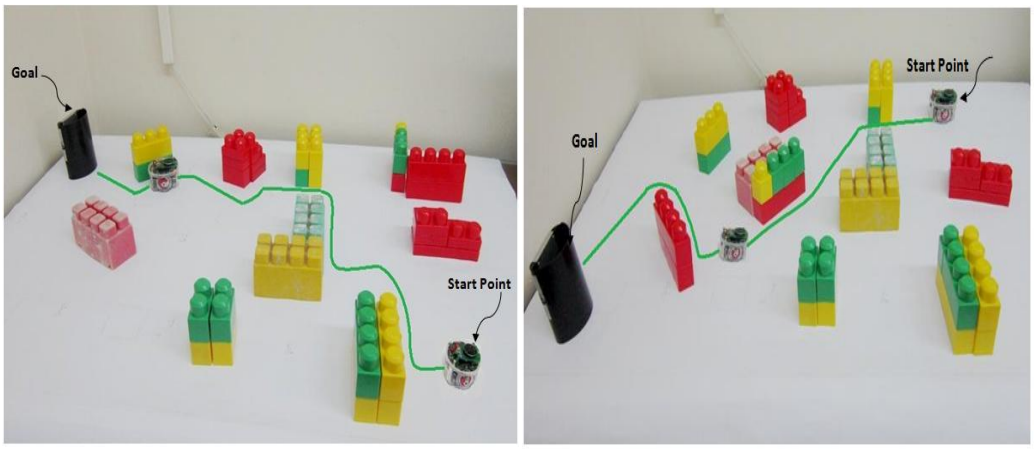
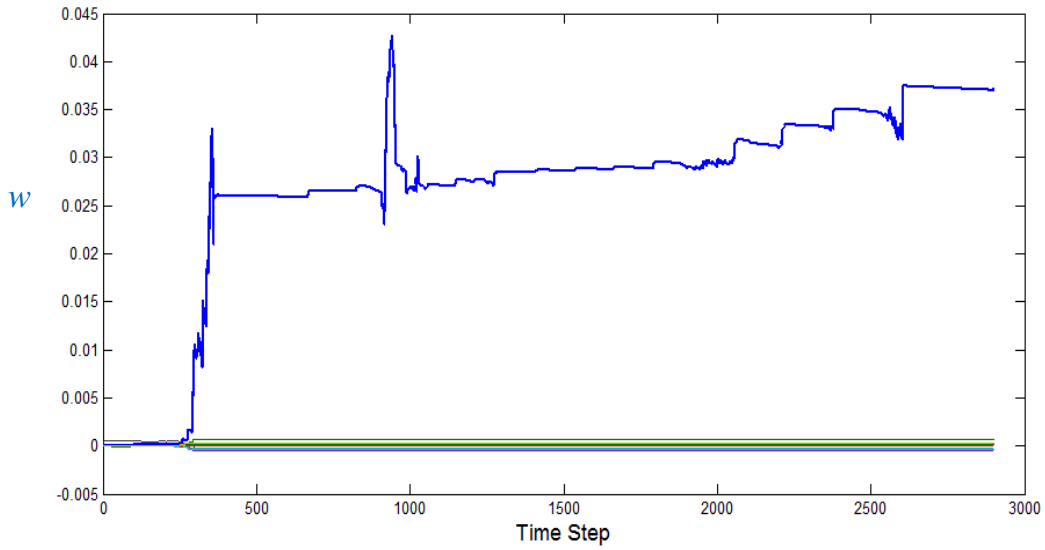
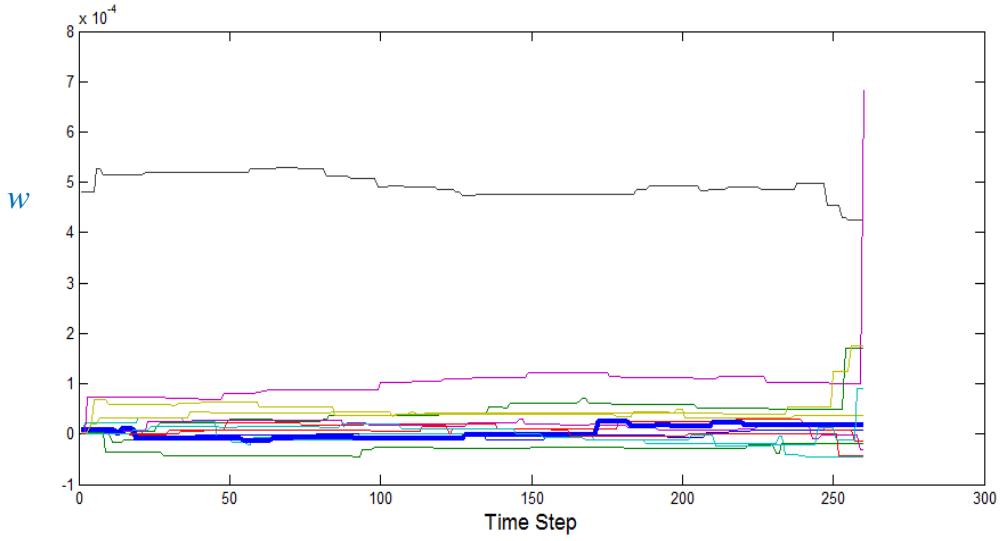


FIGURE 9.The traversed path by E-puck robot in two testing environments (The black cylinder is the goal, the colored cubes are obstacles and the green line shows the traversed path by the robot)

The changes of the value of candidate actions (w^{ij}) can be examined during learning by Figure 10. It shows the changes of values $(w^{23,j}, j = 1,2,..13)$ in 23-rd fuzzy rule. As seen in Figure 10 (a), the value of the first candidate action (o_1= -45°) has the highest amount. Figure 10 (b) shows the changes in the beginning of learning; as seen, a short time after learning starts (about 250 time-steps) the value of the action o_1= -45° has surpassed from others. Moreover, the order of values of the candidate actions does not change after 250-th time-step.



(a) The changes of the value of candidate actions in 3000 time-steps



(b) The zooming curves in the begining

FIGURE 10.Changes of values $(w^{23,j}, j = 1,2,..,13)$ in 23-rd fuzzy rule.

6. CONCLUSION

In this paper, a new approach for combination of supervised learning and fuzzy reinforcement learning was proposed. The proposed method was used to drive an E-puck robot in various environments with obstacles. The command control was generated based on subsumption architecture. A zero order T-S fuzzy controller was employed as obstacle avoidance module in this architecture. The aim of learning was to find the best conclusion part for each rule of the controller. The proposed method uses the training (supervisory) data to obtain an approximated value (worth) of each candidate action of the fuzzy rules as initial value. Afterwards, FSL was used as a continuous RL algorithm for online final tuning of conclusion parts of the fuzzy controller. The results of experiments on the real robot revealed a significant improvement in decreasing the learning time and the number of failures. These

advantages were achieved because of the training data were used for initializing the value (worth) of candidate actions for each state, in contrast to the conventional methods that use training data for determining controller's output value for each state. Therefore, not only destructive effects of inconsistent data were prevented but also the hidden knowledge in this data was extracted. As another advantage, the proposed structure does not need a module for combining behaviors, in each time only the output of one of modules is applied to the robot. Therefore, the computational cost and designing complexity have been reduced.

ACKNOWLEDGMENT

This research was supported by Iran National Science Foundation (INSF). The authors would like to thank INSF for its support.

REFERENCES

[1] T.Belker, M.Beetz and A.Cremers, Learning action models for the improved execution of navigation plans, *Robot. Auton.Syst*. 38 (2002), 137-148.

[2] C.Ye, N.H.C.Yung and D.Wang, A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle voidance, *IEEE Trans.Syst. Man Cybern. Part B* 33(2003), 17-27.

[3] T.Fong, I.Nourbakhsh and K.Dautenhahn, A survey of socially interactive robots, *Robot. Auton. Syst.* 42(2003), 143-166.

[4] Y.LeCun, U.Muller, J.Ben, E.Cosatto and B.Flepp, Off-Road obstacle avoidance through end-to-end learning, *Advances in Neural Information Processing Systems (NIPS),* 18, *MIT Press* 2005.

[5] S.Wang, W.Chaovalitwongse and R.Babuska, Machine learning algorithms in bipedal robot control, *IEEE Trans. Syst. Man Cybern. Part C* 42(2012), 728-743.

[6] P.Hartono and T.Trappenberg, Internal representation of sensory information for training autonomous robot, *International Conference on Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS),* 2012, pp. 341 – 345.

[7] J.S.R.Jang, CT.Sun and E.Mizutani , Neuro-Fuzzy and soft computing. Prentice-Hall, Taiwan 1997.

[8] PK.Mohanty and D.R.Parhi, Navigation of an Autonomous mobile robot using intelligent hybrid technique, *IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)* 2012, pp. 136 – 140.

[9] F.Bahreynian and M.Palhang, Depth control of under-water robot with coefficients of PID tune by reinforcement,*learning, International Conference on Robotics and Mechatronics (ICROM),* 2013.

[10] P.Melin, L.Astudillo, O.Castillo, F.Valdez and M.Garcia, Optimal design of type-2 and type-1 fuzzy tracking controllers for autonomous mobile robots under perturbed torques using a new chemical optimization paradigm, *Expert Syst.* 40 (2013), 3185-3195

[11] A.Melendez and O.Castillo, Evolutionary optimization of the fuzzy integrator in a navigation system for a mobile robot, *Hybrid Intelligent Systems* (2013), 21-31.

[12] M.J.Er and C.Deng, Online tuning of fuzzy inference systems using dynamic fuzzy Q-learning, *IEEE Trans. Syst. Man Cybern. Part B* 34 (2004), 1478 – 1489.

[13] X.Li and K.Hirasawa, Continuous probabilistic model building genetic network programming using reinforcement learning, *Applied Soft Computing,* 27 (2015), 457–467.

[14] V.Derhami, V.Majd and M.Nili Ahamadabaadi, Fuzzy Sarsa learning and the proof of its stationary points, *Asian. J. Cont.* 10 (2008), 535-549.

[15] L.Jouffe, Fuzzy inference system learning by reinforcement methods, *IEEE Trans.Syst. Man Cybern. Part C* 28,

(1998), 338-355.

[16] J.Henderson, O.Lemon and K.Georgila, Hybrid reinforcement/supervised learning for dialogue policies from communicator data, *Computational Linguistics*, 34 (2008), 487-511.

[17] L.Lin, H.Xie, D.Zhang and L.Shen, Supervised neural Q_learning based motion control for bionic underwater robots*, J. Bionic. Eng*. 7 (2010), 177-184.

[18] N.Navarro-Guerrero, C.Weber, P.Schroeter and S.Wermter, Real-world reinforcement learning for autonomous humanoid robot docking, *Robot. Auton.Syst.* 60(11), (2012), 1400-1407.

[19] D.Zhao, Z.Hu, Z.Xia, C.Alippi, Y.Zhu and D.Wang, Full-range adaptive cruise control based on supervised adaptive dynamic programming, *Neurocomputing,* 125 (2014), 57-67.

[20] V.Derhami, V.Majd and M.Nili Ahamadabaadi, Exploration and exploitation balance management in fuzzy reinforcement learning,. *J.Fuzzy set. Syst.* 161 (2010) 578-595.

[21] R.A.Brooks, A robust layered control system for a mobile robot, *IEEE J. Robot.Autom,* 2 (1986), 14-23.

[22] RS.Sutton and A.G.Barto, Reinforcement learning: An introduction, *Cambridge, MIT Press*,1998.

[23] F.Mondada, M.Bonani, X.Raemy, J.Pugh, C.Cianci, A.Klaptocz, D.Floreano and A.Martinoli ,The E-puck, a robot designed for education in engineering, *Proceeding of 9th Conference on Autonomous Robot Systems and Competitions*, 2009, pp.59–65.

[24] E-puck web site, EPFL education robot, *http://www.E-puck.org, Accessed* 20 May 2013.

[25] W.Liu and AF.Winfield, Open-hardware E-puck linux extension board for experimental swarm robotics research, *Microprocessors and Microsystems*, (2011), 60–67.

[26] A.Gutierrez, AM.Campo, J.Dorigo, F.Donate and L.Monasterio-Huelin, Open E-puck range and bearing miniaturized board for local communication in swarm robotics. *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3111–3116.

[27] M.Mirian, B.N.Araabi, M.Nili Ahmadabadi and R.Siegwart, METAL: a framework for mixture-of-experts task and attention learning, *J. Intell. Fuzzy.Syst.* 23 (2012), 111-128.

[28] W.Junfeng, Z.Wanying and W.Shengda, A two-wheeled self-balancing robot with the fuzzy PD control method, *Mathematical Problems in Engineering,* (2012),1-13.

[29] P.Melin, L.Astudillo, O.Castillo, F.Valdez and M.Garcia, Optimal design of type-2 and type-1 fuzzy tracking controllers for autonomous mobile robots under perturbed torques using a new chemical optimization paradigm, *Expert Systems with Applications* 40 (2013), 3185–3195.

[30] C.J.Harper and A.F.T.Winfield, A methodology for provably stable behaviour-based intelligent control, *Robot. Autom. Syst,* 54 (2006), 52-73.

[31] K.Anam, Prihastono, H.Wicaksono, R.Effendi , SI.Adji, S.Kuswadi, A.Jazidie and M.Sampei , Hybridization of fuzzy Q-learning and behavior-based control for autonomous mobile robot navigation in cluttered environment, *ICROS-SICE International Joint Conference*, 2009, pp. 1023 – 1028.

[32] H.Maaref, C.Barret ,Sensor-based navigation of a mobile robot in an indoor environment, *Robot. Auton. Syst.* 38 (2002), 1–18.

[33] Prihastono, H.Wicaksono, K.Anam, R.Effendi , SI.Adji, S.Kuswadi and A.Jazidie, Autonomous five legs robot navigation in cluttered environment using fuzzy Q-learning and hybrid coordination Node, *ICROS-SICE International Joint Conference*, 2009, pp. 2871 – 2874.

[34] P.M.Yanik, J.Manganelli, J.MeriNo and A.L.Threatt, Kinect depth data and growing neural gas for gesture based robot control, *6th International Conference on Pervasive Computing Technologies for Healthcare*, 2012, pp. 283 – 290.

*Graphical abstract (for review)