Mobile Robot Navigation Using Reinforcement Learning in Unknown Environments

M. U. KHAN

Abstract— **In mobile robotics, navigation is considered as one of the most primary tasks, which becomes more challenging during local navigation when the environment is unknown. Therefore, the robot has to explore utilizing the sensory information. Reinforcement learning (RL), a biologically-inspired learning paradigm, has caught the attention of many as it has the capability to learn autonomously in an unknown environment. However, the randomized behavior of exploration, common in RL, increases computation time and cost, hence making it less appealing for real-world scenarios. This paper proposes an informed-biased softmax regression (iBSR) learning process that introduce a heuristic-based cost function to ensure faster convergence. Here, the action-selection is not considered as a random process, rather, is based on the maximum probability function calculated using softmax regression. Through experimental simulation scenarios for navigation, the strength of the proposed approach is tested and, for comparison and analysis purposes, the iBSR learning process is evaluated against two benchmark algorithms.**

Index Terms— **Reinforcement learning, mobile robots, navigation, autonomous, unknown environment**

I. Introduction

In recent years, the impact of robots in our daily lives and in industry has increased by manifolds. According to the World Robotics Report 2018, the demand for robots increased by 31% in one year as compared to 2016. Whether it is an industrial robot, mobile robot or any other, if it has to interact with the environment, it should be able to navigate. The task of navigation can be further broken down into localization and path planning. In localization, the robot's pose, i.e. its orientation and translation needs to be determined with respect to the surroundings. Path-planning is a process in which a robot should be able to find out its collision-free, feasible path in an on-line or off-line fashion, from the start to the goal point.

The task of path-planning can be more challenging in the presence of obstacles and in unknown environments. The basic trait of the autonomous mobile robot is that it should be able to traverse through the obstacles in a safe manner and attain the goal position. Navigation requires the robot to continuously update its information about the surroundings and plan its next action, accordingly. In general, such information is acquired by means of various sources, such as GPS [1], ultrasonic [2], and laser range finder [3]. This information about the robot's pose and obstacles' location is, then, utilized to control the movements of the robot through its actuators.

Over the last decade, many researchers have been continuously striving to solve the problem of path-planning in an effective manner. The most notable algorithms in this respect are A* [4], Dijkstra [5], PRM [6], and RRT [7]. More recently, researchers have shown their keen interest towards biologically-inspired algorithms, such as ANN [8], GA [9], PSO [10], and RL [11]. Among these, RL algorithms have received special attention due to their efficient problem solving in various fields such as control engineering, game theory, and even robotics.

The most renowned one of all RL algorithms is Q-learning [12], proposed in 1989, based upon the learning from delayed rewards and punishments. Since then, many researchers have quite effectively utilized Q-learning for mobile robot navigation and obstacle avoidance [13, 14]. The Q-learning has also been tested for solving mobile robots' path-planning problem in 3D environments [15]. Many authors have also proposed hybrid approaches by combining the un-supervised RL with Fuzzy Logic [16, 17], or Artificial Neural Network [18, 19].

The learning process, true online SARSA Q-biased softmax regression (TOSL-QBIASSR) [20], evolved from classical Q-learning, has attempted to tackle a wide variety of robotic tasks with minimal tuning required. A complimentary low-reward-loop evasion algorithm has been utilized to avoid local minima sequences. An open-source software framework is also developed with a large collection of various learning processes.

Recent research has attempted to integrate deep learning with RL as deep reinforcement learning (DRL) to deal with a wide variety of control problems. Although, deep learning enables an effective learning process for high-dimensional tasks, it also requires high computational costs both in terms of the number of used cores and computational time. This is a drawback which prevents such algorithms to be considered for real-time applications.

From the literature review, the conclusion can be drawn that there is a remarkable tendency among researchers to solve navigation problems using RL since, in most cases the

MUHAMMAD UMER, KHAN, is with Department of Mechatronics Engineering, Atilim University, Ankara, Turkey, (e-mail: umer.khan@atilim.edu.tr).

https://orcid.org/0000-0002-9195-3477

Manuscript received February 26, 2019; accepted June 10, 2019.

DOI: 10.17694/bajece.532746

environment is unknown and involves a great degree of uncertainty. The bench marking Q-learning algorithm has been tested for many scenarios and is found to be effective as it can deal with unknown non-deterministic Markovian systems. The more recent TOSL-QBIASSR learning tested for multiple tasks is claimed to be effective. Therefore, for comparison and analysis, Q-learning with softmax regression (Q-SR) and TOSL-QBIASSR has been chosen as the test bench.

In this paper, using simulation, it is shown that the proposed approach outperforms the Q-SR and TOSL-QBIASSR. In detail, the main focus of this research is to perform 2D navigation in an unknown environment by combining the recent TOSL-QBIASSR learning process with a more informed action-selection technique. The contributions of this paper are three-fold:

- The true online SARSA informed-Biased Softmax Regression (TOSL-iBSR) is proposed which introduce a heuristic-based cost function to TOSL-QBIASSR to ensure faster convergence;
- An optimum action-selection is proposed based upon the maximum probability function value of the state instead of randomly picking the action; and
- A learning process based upon the Boltzmann distribution does not use a constant thermodynamic temperature; rather, an annealing schedule is introduced to ascertain the global maximum by moving towards narrower and narrower regions from wider ones in the start.

The paper is structured as follows: In section II, a brief summary of RL is presented together with a problem statement. The proposed approach based upon the existing QBIASSR algorithm is detailed in Section III. Section IV addresses the environment setup and implementation issues. Section V includes simulation results, analysis and discussion. Finally, section VI summarizes the conclusion and future work.

II. Background

In this section, a brief summary of the related RL algorithms is provided, followed by the problem statement.

Assumption:
The differential drive robot considered is from the broad class of Wheeled Mobile Robots (WMRs). It is assumed that the moving frame is associated to the robot using which location of the robot (x,y) in 2D plane and heading angle θ can be updated and available at all times.

2.1 Markov Decision Process

The process of learning and improvement has always been considered as a built-in feature among humans. Based upon these characteristics, a well-known mathematician, Alan Turing, invented the Turing machine in 1936 [21]. This machine mechanically operated on discrete states, where the state register stored the state of the machine. It was also featured with decision-making so as to select suitable actions. Markov generalized this idea for the situations, where the outcomes are partly random and partly under the control of a decision-maker [22]. The process, formally named after Markov as Markov Decision Process (MDP), is extensively applied in many disciplines, such as robotics and automatic control to name a few. An MDP is defined as a 5-tuple $\left(\mathcal{S},\mathcal{A},\mathcal{P}_a\left(s'|s,a\right),\mathcal{R}\left(r|s\right),\vartheta\right)$, where

- $\mathcal{S}$ is a finite set of states,
- $\mathcal{A}$ is a finite set of actions,
- $\mathcal{P}_a\left(s'|s,a\right)$ is the transition probability to define the chances that the next state s' will be picked as a consequence of action a,
- $\mathcal{R}\left(r|s\right)$ is the expected reward received due to action a while in state s', r is a signed value used for the reward or punishment,
- ϑ are specific parameters for some RL algorithms' settings. This will be the discount factor $\gamma\in[0,1]$ and learning rate $\alpha\in[0,1]$ in our case.

The output of the process is dependent upon the selection of the policy $\pi:\mathcal{S}\rightarrow\mathcal{A}$, that specifies the action $\pi(s)$, chosen by the decision-maker while in state s. The goal of the Markov record process is to determine the state and reward sequence given that the policy $\pi(s)$ ensures a maximum cumulative reward. RL has proved itself to be an effective tool for closed-loop problems that satisfy the Markov property as to maximize numerical reward in an unknown environment [11], [23], [24].

2.2 Reinforcement Learning

Almost in all RL algorithms, the main focus of RL algorithms is upon maximizing policy value $\mathcal{V}$, which is a direct indicator of the long-term desirability of states considering the rewards available in those states.

Widely accepted, Q-learning is probably the most practical and effective algorithm that belongs to the Temporal Difference (TD) learning, a model-independent and fully-incremental algorithm. The state-value function V^{π} gives information about the desirability of that state for an agent under a policy π, and is defined as:

$$\mathcal{V}_k^{\pi}\left(s\right)=max_a\, Q_k\left(s,a\right) \qquad (1)$$

The successful outcome of the Q-learning (Algorithm 1), as well as of most RL algorithms is heavily affected by the exploration and exploitation phenomena. The optimal choice between these two needs to be made because none of them can be pursued exclusively without failing at the task. The agent's behavior must be evaluated by repeatedly trying different combinations of parameters α and γ in order to define the balance between the two.

Algorithm 1: Q-Learning

```
Input:
    States $\mathcal{X} = \{1,\dots,n_x\}$
    Actions $\mathcal{A} = \{1,\dots,n_a\}$
    Reward function $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
    Learning rate $\alpha \in [0,1]$, typically $\alpha$ is set to be 0.1
    Discounting factor $\gamma \in [0,1]$

Procedure:
    Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
    repeat
        Pick state $s \in \mathcal{X}$
        repeat
            select new action $a$ (based upon the exploration strategy)
            perform action $a$
            observe new state $s'$ and attain reward $\mathcal{R}$
            update
            $Q(s,a) \leftarrow Q(s,a) + \alpha(\mathcal{R} + \gamma V(s') - Q(s,a))$
            $V(s) = max_a Q(s,a)$
            update state $s \to s'$
        until $s$ is not a terminal state
    until $Q$ is not converged
Output:
    Best action selection $a'$
```

Algorithm 2: TOSL Q-biased softmax regression (TOSL-QBIASSR)

```
Input:
    States $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$
    Actions $\mathcal{A} = \{a_1, a_2, \dots, a_p\}$
    Input variables $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$
    $Q(s) = [Q(s,a_1), Q(s,a_2), \dots, Q(s,a_p)]$
    $\mathcal{SQ}(s \in \mathcal{S}, x_i \in \mathcal{X})$ = subset of states $ss \in \mathcal{S}$ with
    $x_j(ss) = x_j(s) \forall j \in [1,2,\dots,n] \neq i$
    Agent in state $s$ must select action $a'$ given $Q$

Procedure:
    for all the $x_i \in x$ do
        $bias(s,x_i) \leftarrow avg(\mathcal{SQ}(s,x_i)) \quad \forall ss \in \mathcal{SS}(s,x_i)$,
        pick state $s \in \mathcal{X}$
    end
    $bias(s) \to \frac{1}{n}\sum_{i=0}^{n} bias(s,x_i)$
    $Q(s)_{biased} \leftarrow Q(s) + bias(s)$
    $a' \leftarrow$ softmax_selection$(Q_{biased}, Temperature)$

Output:
    Best action selection $a'$
```

An improved and practical modification to the Q-learning, State-Action-Reward-State-Action (SARSA) [25], performs the learning based upon the action performed by the current policy instead of the greedy policy.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(\mathcal{R} + \gamma Q(s',a') - Q(s,a)) \tag{2}$$

Hence, a', the action to be performed in the next step must also be evaluated before updating $Q(s,a)$. Another variant of SARSA, the true online SARSA(λ) algorithm (TOSL), proved to be a more efficient learning process [26].
The recently proposed mechanism, TOSL-QBIASSR, combines the advantages of both TOSL and softmax regression. The basic idea is to improve efficiency for the cases where agent experiences new states with strong resemblance to already explored ones. In this approach, softmax regression is performed over a scaled Q-value, defined as $Q(s)_{biased}$:

$$Q(s)_{biased} \leftarrow Q(s) + bias(s) \tag{3}$$

where, $bias(s)$ is a vector generated based upon the information gathered from other states of Q similar to s. At each step the bias is updated using averaged information from sets of states that share some structure with the current state s. The working of TOSL-QBIASSR is detailed in Algorithm 2 [20]:

2.3 Problem Statement

Consider an agent, in our case a mobile robot, operating in a virtual environment in the presence of obstacles. Given the initial state s_0 and ending goal state g, the robot has to determine its path using RL. The path is a sequence of adjacent traversable cells, $(s, next(s), next(s), \dots, g)$; and $next(x)$ stands for the successor of cell x and, for this paper, it will be the neighboring (adjacent) cell.

III. LEARNING PHASE OF THE AGENT

In mobile robotics, navigation and wandering are considered as the foremost scenarios for operating with the former regarded as more complex. In robotics navigation, the task is to enroute through the obstacles safely and approach the target. In most RL algorithms, the mobile robot observes the environment and updates its reward, which it intends to maximize. Usually, the agent drives itself based upon the acquisition of the highest award. Therefore, it is desired to induce information about the goal to accelerate the achievement of the highest reward, thus also ensuring that the agent approaches the solution in a finite time.
In TOSL-iBSR (Algorithm 3), an acceptance probability function $P(e,e',\mathcal{T})$, depending upon energies $e = E(s)$ and $e' = E(s')$ of the two states and a global time varying temperature $\mathcal{T}$, is introduced in order to define the probability

Algorithm 3: TOSL informed-biased softmax regression (TOSL-iBSR)

```
Input:
    States $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$
    Actions $\mathcal{A} = \{a_1, a_2, \ldots, a_p\}$
    Input variables $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$
    $Q(s) = [Q(s,a_1), Q(s,a_2), \ldots, Q(s,a_p)]$
    Temperature $\mathcal{T} \leftarrow \mathcal{T}_{max}$
    $\mathcal{S}Q(s \in \mathcal{S}, x_i \in \mathcal{X}) =$ subset of
    states $ss \in \mathcal{S}$ with $x_j(ss) = x_j(s) \forall j \in [1,2,\ldots,n] \neq i$
    Agent in state $s$ must select action $a'$ given $Q$

Procedure:
    for all $x_i \in x$
        $bias(s, x_i) \leftarrow avg(\mathcal{S}Q(s, x_i)) \quad \forall ss \in \mathcal{SS}(s, x_i)$ ,
        pick state $s \in \mathcal{X}$
    end
    generate biased state $bias(s) \rightarrow \frac{1}{n}\sum_{i=0}^{n} bias(s, x_i)$
    generate virtual vector $Q(s)_{biased} \leftarrow Q(s) + bias(s)$
    temperature update $\mathcal{T} \leftarrow g(\mathcal{T})$
    input feature for softmax
    regression $z^{(i)} \leftarrow \frac{Q(s)_{biased} + \mathcal{J}(s')}{\mathcal{T}}$
    acceptance probability
    function $P(e, e', \mathcal{T}) \leftarrow \phi_{softmax}(z^{(i)})$
    select action $a' \leftarrow \text{argmax}_a \; P(e, e', \mathcal{T})$
Output:
    Best action selection $a'$
```

of making the transition from the current state s to a candidate new state s'. The probability function P is bound to be non-negative in order to avoid local minimum.

In the proposed algorithm, when computing the input feature for softmax regression, the energy-based cost function is introduced by means of the cost function $\mathcal{J}(s')$, which is calculated at each possible next state during the learning process. Many approaches can be used to define a suitable $\mathcal{J}(s')$ as either: heuristic-based function such as the distance covered from the start, the remaining distance, or a combination of both. In this work, cost is introduced as the Euclidean norm of the remaining distance:

$$\mathcal{J}(s') = \|s' - g\| \tag{4}$$

The evolution of the state s of the system is also affected by temperature $\mathcal{T}$. The evolution process is sensitive to coarser energy variations for a large $\mathcal{T}$; whereas, for a small $\mathcal{T}$, it is sensitive to finer such variations. Considering this fact, the algorithm initializes with quite high temperature, $\mathcal{T}_{max}$, and gradually approaches towards zero, $\mathcal{T} \approx 0$, following some annealing schedule $g(\mathcal{T})$.

$$\mathcal{T} \leftarrow g(\mathcal{T}) = \mathcal{T}(1-\varepsilon) \tag{5}$$

where, ε is considered as a small number. Combining this with the built-in feature of RL as to search for the states with highest reward, the agent drifts towards low-energy regions that become narrower and narrower and, finally, moves downhill.

To assess the goodness of true online SARSA with informed-biased softmax regression (TOSL-iBSR), a comparison with two well-known approaches is made, namely, Q-learning with softmax regression (Q-SR), and true online SARSA with Q-biased softmax regression (TOSL-QBIASSR). In the context of this paper, 2D navigation scenario in Virtual Robot Experimentation Platform (V-REP) is considered. The setup and results are described in Section 4 and 5. This comparative study proves that TOSL-iBSR outperforms other learning processes, both in terms of the mean-average reward and the convergence rate.

IV. IMPLEMENTATION PRE-REQUISITES

In the first step, the virtual environments are developed in V-REP. The routines for the learning process and the robot movement's control were written in Python. The versatility of the scheme that makes it different from others is the use of a mobile robot with physical characteristics in a virtual environment. The whole learning process is executed in real-time where Remote API is used to control the robot's movement from within Python.

4.1 Physical characteristics of the mobile robot

The Pioneer 3dx mobile robot, shown in Fig. 1, is a differential drive robot developed by Adept. The 3D model of this robot is readily available in the V-REP library. The robot is equipped with 16 ultrasonic sensors, with related information as to the obstacles available in Python. It features 2 powered rear wheels and 1 castor wheel.

Fig. 1. Pioneer 3dx

4.2 Modeling of the work environment

The real-world environments are commonly filled with obstacles of different sizes. Figs. 2 and 3 represents the virtual scenes developed in V-REP. The elements that exist in the environments are:

Virtual Scenario 1

Workspace: The environment is a $4\times 4m$ grid resembling a maze, which is fully confined from the outer end.

Obstacles: Three obstacles with size attribute of $(0.1,1.0,0.4)m$, and another bigger one as $(0.1,2.0,0.4)m$.

Virtual Scenario 2

Workspace: The environment is a $6\times 6m$ grid fully confined from the outer end.

Obstacles: Five obstacles are placed in a disordered manner with size attributes of $(0.1,0.5,0.4)m$, and $(0.1,1.0,0.4)m$.

Target: The target is a bounding box.

Robot and sensors: The Pioneer robot viewable in the scene is equipped with 16 ultrasonic sensors covering all sides. The ultrasonic beam angle is defined as 30° and conically shaped with the range measurement defined as $1.0m$.

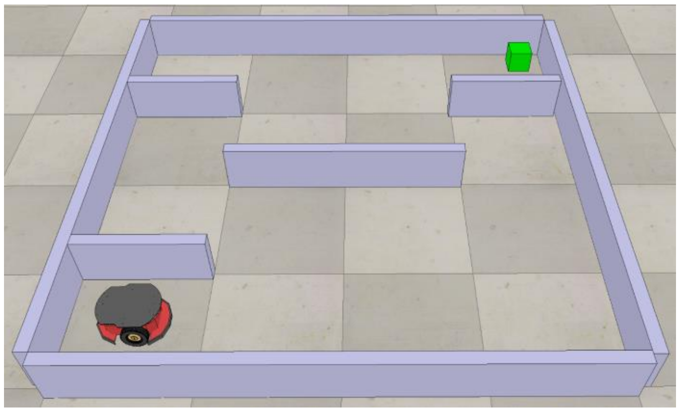
Fig. 2. Navigation scenario, $4\times 4m$ with obstacles

4.3 Reward Function

The reward value is determined based upon the observation information of each state. Mostly, the reward value is updated based upon the distance from the obstacles and/or the target location. To define a simple yet effective reward function, the desired end-position of the task is defined as g; each time the robot goes within the safe distance d, defined between obstacle O and the robot, a collision n_c is detected. The reward rules are defined as follows:

$$r=\begin{cases} R & ,s=g \\ -0.2R & ,s\in O \text{ and } n_c=1 \\ -R & ,s\in O \text{ and } n_c>1 \end{cases} \tag{6}$$

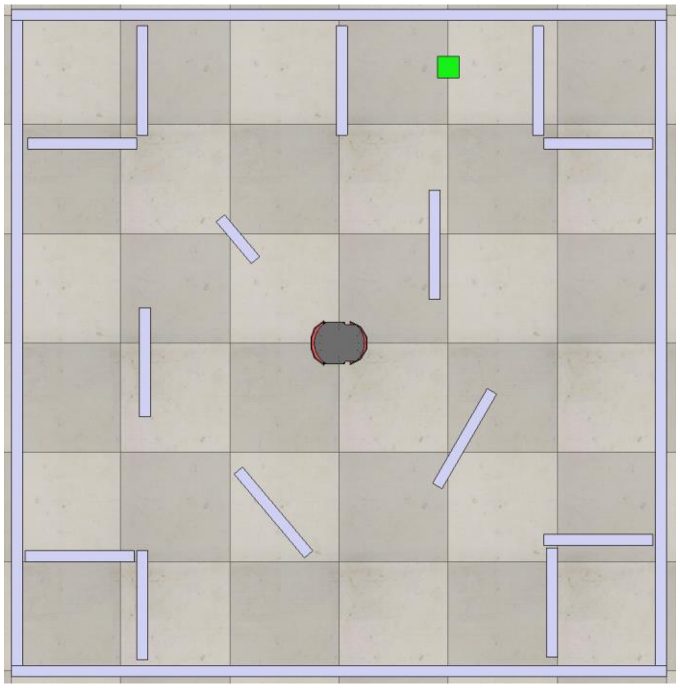
Fig. 3. Navigation scenario, $6\times 6m$ with obstacles

4.4 Interfacing Python with V-REP

Fig. 4 illustrates the application architecture adapted for the implementation of the entire process. The architecture is system-independent; though, depending upon the operating system, the workspace configuration files need to be selected from within the V-REP installation folder. These configuration files consist of Remote API and Dynamic Link Libraries. The link to the V-REP server can be made through socket communication by specifying the IP address as well as a connection port. On the V-REP, Pioneer will perform the action chosen by Python. The sensory information is, then, forwarded to Python and is used for calculating the reward points and the next sequence state.

The task is considered completed once the robot reaches target g. The first time it comes in close contact with an obstacle, a small penalty is imposed.

V. SIMULATED EXPERIEMNTS AND DISCUSSIONS

For the experimental evaluation of 2D navigation, the V-REP scene is kept the same as in [20] for true comparison. The Pioneer 3dx is a differential drive robot with 2 DC motors and encoders. The kinematics for the differential drive robot under no-slippage is

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}=\begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \tag{7}$$

where, v is the robot translational velocity and ω is the robot rotational velocity defined with respect to the body frame. The linear and angular velocities of the robot are associated to the individual wheel's angular velocities as:

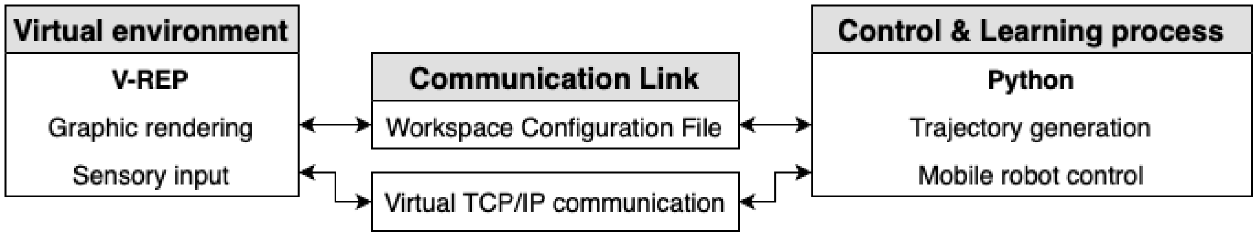

Fig. 4. Application architecture

$$v = \frac{(\omega_r + \omega_l)}{2}$$
$$\omega = \frac{(\omega_r - \omega_l)}{W} \qquad (8)$$

where, ω_r and ω_l are the angular velocities of the right and the left wheels, and W is the distance between the two's centers. The control input is directly given to the wheels in terms of ω_r and ω_l.

For the defined task and scenarios, the agent has the freedom to choose between any of its immediate neighboring states $\in \{E, NE, N, NW, W, SW, S, SE, \phi\}$. The last state, i.e. the null state, refers to the case where the next state is the same as the current one; therefore, in total there will be 9 possible actions. Through preliminary tests, the tuning parameters α and γ are selected as 0.1 and 0.9. Three algorithms: Q-SR, TOSL-QBIASSR and TOSL-iBSR are evaluated in terms of the performance.

A. Scene 1: Maze grid $4\times4m$

The results of the learning process for the three algorithms are averaged for 6 episodes and plotted against 3600 times steps. From Fig. 5, it can be observed that the TOSL-iBSR outperforms both TOSL-QBIASSR and Q-SR. For episodic tasks, the mean-average reward is a true indicator as it gives us a clearer picture. At the start of the task, the mean-average reward is evaluated as negative. It can be witnessed that just after 500 steps with the learning process in place, the mean-average reward began to pull up. The QBIASSR shows steady behavior towards the end of the task as it creeps towards the iBSR curve; therefore, it will obviously require additional steps in order to reach it.

The learning curves for the three algorithms are also shown in Figs. 6-8. The average reward obtained using the Q-SR is acceptable, but still far from the maximum reward and, hence, it demands additional convergence time. For the QBIASSR, the difference between the learning curves is high compared to iBSR, thereby generating a low mean-average reward as witnessed in Fig. 5. On the other hand, the Q-SR is observed to yield much flatter learning process.

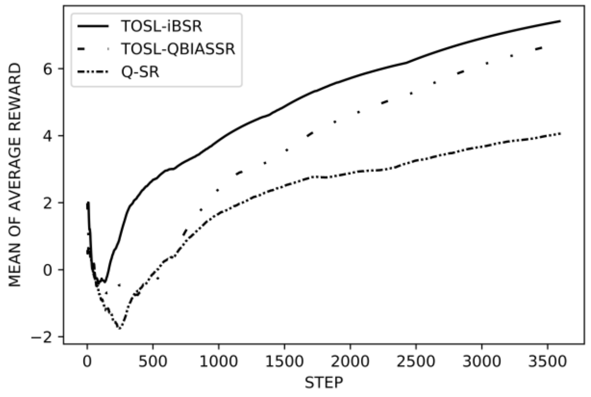

Fig. 5. Learning results for 6 episodes

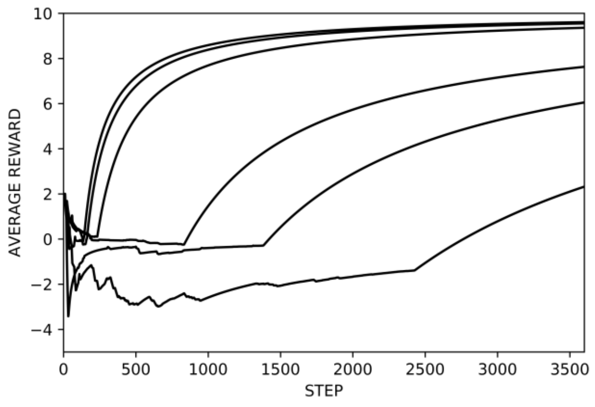

Fig. 6. Average reward for TOSL-iBSR for 6 episodes

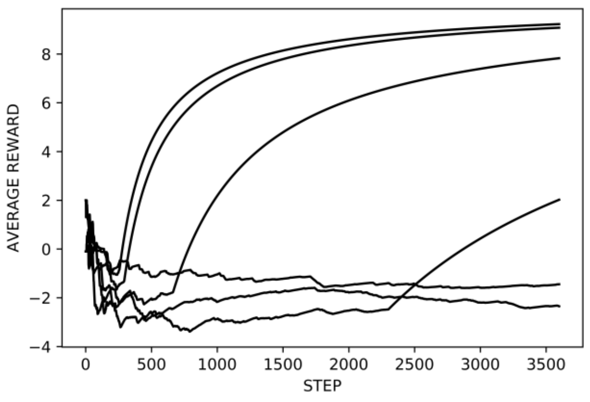

Fig. 7. Average reward for Q-SR for 6 episodes

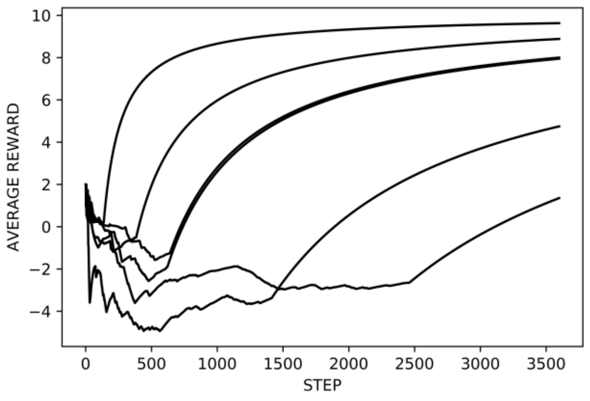

Fig. 8. Average reward for TOSL-QBIASSR for 6 episodes

Fig. 9 illustrates the mean-average reward for the three algorithms over 3600 steps and 6 episodes. This gives us a clear picture as to the performance of the three algorithms. As it can be seen, the Q-SR curve shows poor learning since, after 3 episodes, it is still unable to repeat its behavior which is indicative of a lack of learning, repeatability and consistency. It shall be stated that the task was not fully completed after episode 4. The learning curve of iBSR shows that the robot has learned to perform the task in an effective manner with continuous improvement; whereas, QBIASSR curve, albeit starting with a higher reward, remained unable to maintain its superiority. For implementation, Intel Core 2 Duo processor under Windows 64 bits has been utilized. To evaluate the computational cost, CPU time per step (s) is defined as a measure. The results in Table 1 shows that Q-SR is the second most effective learning process after TOSL-iBSR; whereas, TOSL-QBIASSR demanded more learning steps to complete the task. The same can be observed through Figs. 7 and 8.

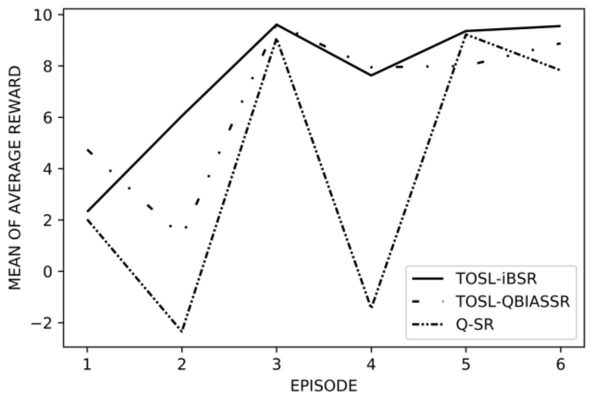

Fig. 9. Mean-average reward per 3600 steps

B. Scene 1: Maze grid $6\times 6m$

A more complex scenario (Fig. 3) for navigation is considered where obstacles are placed randomly. The learning process is repeated for 6 episodes; 3600 time steps per episode. The results shown in Fig. 10 highlight the performances of each learning process. It is worth mentioning that Q-SR has not been able to complete the task even for once within specified number of steps. TOSL-QBIASSR has been moderate in performance and is dominated by TOSL-iBSR for attaining highest reward. The QBIASSR learning process is very slow and sluggish as compare to the iBSR whose learning curve demonstrates notable continuous improvement.

The results of the learning experiments are individually shown in Figs. 11-13. The average reward obtained using Q-SR is always negative; therefore, completely fails in this scenario. For the QBIASSR, in only half of the episodes, the learning curve has been able to attain steady state value, thereby generating a low mean-average reward as compare with iBSR whose performance has been very much consistent throughout.

The mean-average reward for the three learning processes over 3600 steps and 6 episodes are shown in Fig. 14. The learning curve of Q-SR presents continuous attainment of negative reward; hence, it has not been able to accomplish the task even for once. The mean average reward obtained using iBSR exhibits that it started with higher positive value and is able to continuously improve as compared to QBIASSR that experiences many ups and down before attaining the steady state value after 4 episodes. The computational cost for scene 2, shown in Table 1, has been increased due to its complexity. The computational cost of TOSL with QBIASSR is higher in magnitude as compared to other two. The TOSL-iBSR found to be computationally efficient due to the reason that it requires much lesser number of steps to accomplish the task.

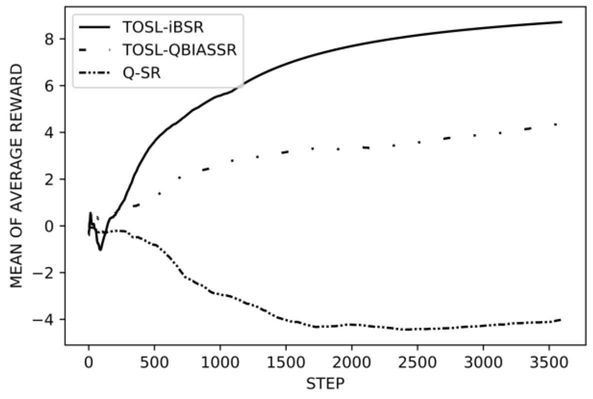

Fig. 10. Learning results for 6 episodes

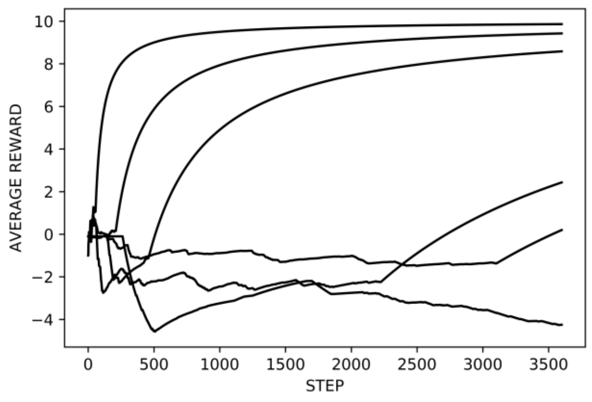

Fig. 13. Average reward for TOSL-QBIASSR for 6 episodes

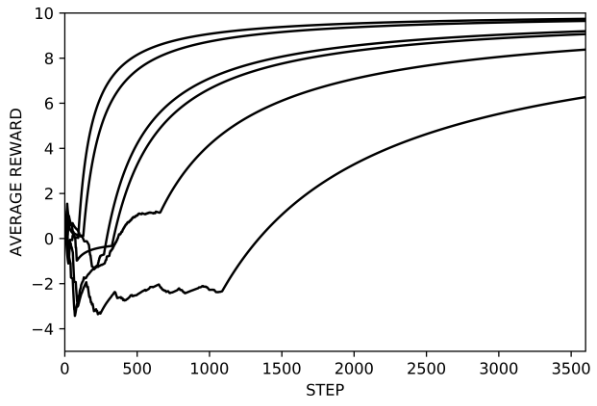

Fig. 11. Average reward for TOSL-iBSR for 6 episodes

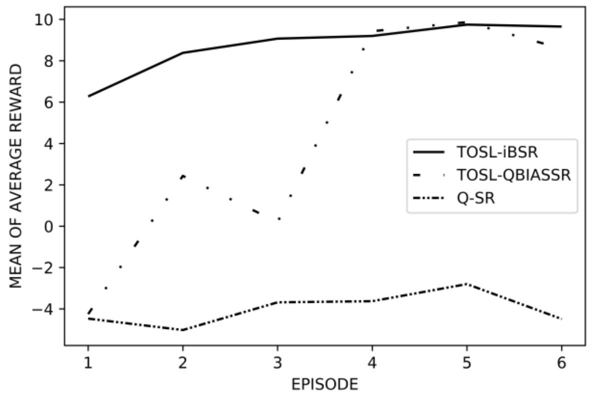

Fig. 14. Mean-average reward per 3600 steps

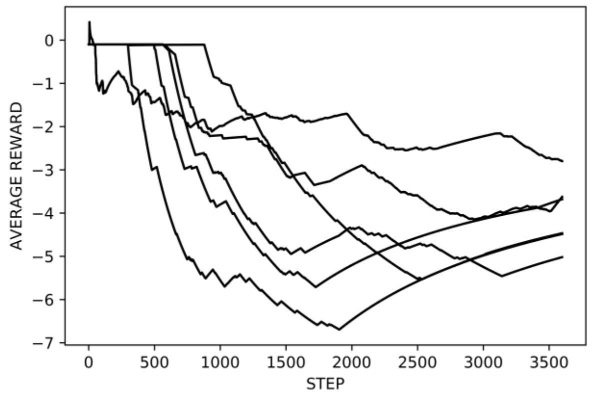

Fig. 12. Average reward for Q-SR for 6 episodes

Table I
COMPUTATIONAL COST

	CPU time / step (s)	
RL algorithm	**Scene 1**	**Scene 2**
Q-SR	0.203	0.350
TOSL-QBIASSR	0.352	0.481
TOSL-iBSR	0.040	0.068

VI. Conclusion

In this research, a new and improved learning process, named as, TOSL informed-biased softmax regression (TOSL-iBSR) is presented for mobile robot navigation. The novel exploration technique is equipped with the ability to pick the most suitable action in order to maximize the reward point and minimize the convergence rate. One of the notable contributions of the present study is to establish the frame-work between Python and the V-REP and to use an actual robot with all physical parameters, instead of merely using a point robot [27, 28, 29] as in most RL navigation-based research literature. A virtual scenario for 2D navigation is generated to test the efficiency of the proposed approach. The performance of the robot during navigation using the TOSL-iBSR is, then, compared with the Q-SR and the TOSL-QBIASSR. Using the approach presented in this paper, the robot is found to complete the navigation task while attaining a higher positive reward and less computational cost.

Despite this accomplishment, one of the minor drawbacks that reduces the generalization may be that the true pose of the robot is available at all time sequences and that the sensory information is noise-free, - which is certainly not the case in real workspaces. In such scenario, an effective approach can be the introduction of a robust estimation technique. As to the future work, it involves the extension and testing of the iBSR for multiple real tasks in actual real-world settings. In addition, the proposed algorithm can be extended for more practical high-dimensional tasks. Incorporating DRL into the proposed algorithm can also generate a promising learning process. Finally, we intend to test the proposed algorithm in a more complex environment, namely scattered, denser, and dynamic.

REFERENCES

[1] S.-H. Kim, C.-W. Roh, S.-C. Kang and M.-Y. Park, "Outdoor navigation of a mobile robot using differential GPS and curb detection," in *Proceedings of IEEE international conference on Robotics and Automation*, 2007.

[2] L. Moreno, J. M. Armingol, S. Garrido, A. D. L. Escalera and M. A. Salichs, "A genetic algorithm for mobile robot localization using ultrasonic sensors," *Journal of Intelligent and Robotic Systems,* vol. 34, no. 2, pp. 135-154, 2002.

[3] A. Sinha and P. Papadakis, "Mind the gap: Detection and traversability analysis of terrain gaps using LIDAR for safe robot navigation," *Robotica,* vol. 31, no. 7, pp. 1085-1101, 2013.

[4] S. J. Russell and P. Norvig, Artificial intelligence: a modern approach, Pearson Education Limited, 2016.

[5] R. E. Korf, Artificial intelligence search algorithms, Chapman & Hall/CRC, 2010.

[6] L. E. Kavraki, M. N. Kolountzakis and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," in *Proceedings international conference on robotics and automation*, 1996.

[7] N. A. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *Proceedings of IEEE international conference on robotics and automation*, 2007.

[8] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 34, no. 1, pp. 718-724, 2004.

[9] M. Z. Malik, A. Eizad and M. U. Khan, Path planning algorithms for mobile robots: a comprehensive comparative analysis, LAP LAMBERT Academic Publishing, 2014.

[10] M. S. Alam, M. U. Rafique and M. U. Khan, "Mobile robot path planning in static environments using particle swarm optimization," *International journal of computer science and electronics engineering*, vol. 3, no. 3, 2015.

[11] R. S. Sutton and A. G. Barto, Reinforcement Learning: An introduction, MIT press, 2018.

[12] C. J. C. H. Watkins, "Learning from delayed rewards," King's College, Cambridge, Ph.D. thesis, 1989.

[13] K. L. Smart W, "Practical reinforcement learning in continuous spaces," in *Proceedings of the seventeenth international conference on machine learning*, 2000.

[14] K. L. Smart W, "Effective reinforcement learning for mobile robots," in *Proceedings of the international conference on robotics and automation*, 2002.

[15] D. Aranibar and P. Alsina, "Reinforcement learning-based-path planning for autonomous robots," ENRI: Encontrol Nacional de Robotica Inteligente, 2004.

[16] H. Boem and H. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE Transaction on System, Man, and Cybernetics*, vol. 25, pp. 464-477, 1995.

[17] N. Yung and C. Ye, "Self-learning fuzzy navigation of mobile vehicle," in *Proceedings of the international conference on signal processing*, 1996.

[18] G. Yang, E. Chen and C. An, "Mobile robot navigation using neural Q-learning," in *IEEE proceedings of the third international conference on machine learning and cybernetics*, 2004.

[19] K. Macek, I. Petrovic and N. Peric, "A reinforcement learning approach to obstacle avoidance of mobile robots," in *7th international workshop on advanced motion control*, 2002.

[20] A. Martínez-Tenor, J. A. Fernández-Madrigal, A. Cruz-Martín and J. González-Jiménez, "Towards a common implementation of reinforcement learning for multiple robotic tasks," *Expert Systems with Applications,* vol. 100, pp. 246-259, 2018.

[21] H. Andrew, Alan Turing: The Enigma, Princeton University Press, 2012.

[22] R. Bellman, "A Markovian decision process," *Journal of Mathematics and Mechanics,* pp. 679-684, 1957.

[23] A. G. Barto, R. G. Sutton and C. W. Anderson, "Neuronlike elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, pp. 835-846, 1983.

[24] A. G. Barto, R. S. Sutton and P. S. Brouwer, "Associative search network: A reinforcement learning associative memory," *Biological Cybernetics,* pp. 201-211, 1981.

[25] G. A. Rummery and M. Niranjan, On-line Q-learning using connectionist system, vol. 37, Cambridge, England: University of Cambridge, Department of Engineering, 1994.

[26] H. Van Seijen, M. A. R, P. M. Pilarski and M. M. C. a. S. R. S, "Ture online temporal-difference learning," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 5057-5096, 2016.

[27] R. Abiyev, D. Ibrahim and B. Erin, "Navigation of mobile robots in the presence of obstacles," *Advances in Engineering Software,* vol. 41, pp. 1179-1186, 2010.

[28] A. I. Panov, K. S. Yakovlev and R. Suvorov, "Grid path planning with deep reinforcement learning: preliminary results," *Procedia Computer*

Science, vol. 123, pp. 347-353, 2018.

[29] J. d. R. Millan, "Reinforcement learning of goal-directed obstacle-avoiding reaction strategies in an autonomous mobile robot," *Robotics and Autonomous Systems*, vol. 15, pp. 275-299, 1995.

[30] R. Bellman, "The theory of dynamic programming," RAND Corp Sanata Monica CA, 1954.

BIOGRAPHIES

Muhammad Umer KHAN received the B.S. in computer science from the International Islamic University, Islamabad, in 2004, and the Ph.D. in electrical engineering from the Pakistan Institute of Engineering and Applied Sciences, Islamabad, in 2011.

From 2011 to 2017, he was serving as an Assistant Professor with the Department of Mechatronics Engineering, Air University, He was a post-doctoral fellow with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, from 2014 to 2015. Since 2018, he has been with the Mechatronics Engineering Department, Atilim University, Turkey. His current research interests include robust control, linear matrix inequalities, reinforcement learning, vision-based control, and navigation.