A Comparison of Two Reinforcement Learning Algorithms for Robotic Pick and Place with Non-Visual Sensing

Muhammad Babar Imtiaz, Yuansong Qiao, and Brian Lee
Software Research Institute, Athlone Institute of Technology, Athlone, Ireland
Email: b.imtiaz@research.ait.ie, {yuangsongqiao, blee}@ait.ie

***Abstract*—In this study, we perform a comparative analysis of two approaches we developed for learning to carryout pick and place operations on various objects moving on a conveyor belt in a non-visual environment, using proximity sensors. The problem under consideration is formulated as a Markov Decision Process. and solved by using Reinforcement Learning algorithms. Learning robotic manipulations using simple reward signals is still considered to be an unresolved problem. Our reinforcement learning algorithms are based on model-free off-policy training using Q-Learning and on-policy training using SARSA. Training and testing of both algorithms along with detailed a comparison analysis are performed in a simulation-based testbed. Our results prove our approaches to be successful in pick and place operations in non-visual industrial setups.**

***Index Terms*—robotic manipulation, non-visual, Markov decision problem, reinforcement learning, Q-learning, SARSA**

I. Introduction

The term 'Industry 4.0' appeared for the first time in 2011 and saw the introduction of many cutting-edge technologies such as Cyber-Physical Systems (CPS), Internet of Things (IoT), and Digital Twins (DT). World-leading scholar Warren G. Bennis was quoted in 2016 saying "The factory of the future will have two employees: a human and a dog. The task of the human will be to feed the dog. The dog will have the task to dissuade the human to touch the automated systems." This quote clearly sums all the expectations and beliefs regarding the implementation of Industry 4.0. Materializing the concept of Industry 4.0 has created a number of global efforts such as Europe's Industry 4.0 [1], America's Advanced Manufacturing [2], China's Made in China 2025 [3], Japan's Super smart society [4], etc. *Smart factories* realize manufacturing processes with the aid of artificial intelligence (AI), the latest novel sensors, and use of robotics.

Improving robotic manipulations with the help of advanced learning techniques has been a focus of the research community for some time now. Reinforcement learning (RL) addresses this task by performing sequential decision-making through a policy learned during the process of maximizing an expected reward. Various complex tasks such as learning and playing board games [5] and video games [6] have been independently mastered by RL agents due to the recent use of deep neural networks as function approximators for reinforcement learning trained agents.

Reinforcement learning has been widely used to address various robotic manipulation problems. Manipulations involving multiple stages like stacking items and hand manipulations [7],[8] have multi-dimensional state and action spaces and complex dynamics and are extremely challenging to solve. Promising results have, however, been recorded in various manipulations such as grasping [9],[10], hand manipulations [7],[8], and stacking [11]. Other manipulations like pick and place [12], pushing objects [12],[13], dual block-stacking [13],[14] and target finding [15],[16] have also shown positive results. Pick and place manipulation has been addressed by various studies, some of which have used deep neural networks [17],[18] while others have used neural networks as function approximators for the RL agents [19]–[23]. But all these studies discuss vision-based approaches to pick and place manipulation.

Machine vision systems evolved over the years to provide us with capabilities such as 3D graphical data and full-color megapixel resolution. But still, often there are various requirements such as high cost which can restrict its use. For instance, the space available for installing an industrial vision system may be only a few centimeters. Other factors that can affect the use of vision system in some particular settings can be vibrations, dust, or even wash-up from water jets. Therefore, in this paper, we present a novel approach that allows learning and performing of pick and place operation in a non-visual industrial environment.

The approaches presented in this paper address the problem of pick and place in a smart production line, where a number of variable-shaped objects are moving on a conveyor belt at different positions and orientation, and where the belt may assume different speeds. The conveyor belt is equipped with ray-type infrared proximity sensors which detect the object and signal the robotic arm to operate. Once the robotic arm receives a

Manuscript received November 27, 2020; revised March 21, 2021.

doi: 10.18178/ijmerr.10.10.526-535

signal from the proximity sensor, it attempts to grasp and pick the moving object, using a pliers-like gripper, from the conveyor belt and place it at a designated location. Our RL agents were trained to learn to successfully perform the pick and place operation on various objects at different positions, orientations, and speeds while receiving inputs only from proximity sensors. After extensive training and testing in our simulated testbed designed using the V-REP simulator, we found our approach achieved a success rate in excess of 90% to pick up objects from the conveyor belt and place them at designated place positions.

The rest of the paper is organized as follows. In section ll, we look at some of the other approaches taken. In section III, we discuss the background of the approach we deployed. In section IV, we develop the methodology of our approach in detail. In section V, we discuss the results and findings of the experiments carried out to test our approach. Finally, section VI concludes the paper with some suggestions regarding future work.

II. Related Work

We find a number of reinforcement learning-based approaches to the problem of pick and place problem in a visual environment but none for the non-visual environment. Generally, these approaches can be divided into two main categories, those that use a geometrical model and the those that don't.

Most early approaches for this pick and place manipulation were totally dependent on the information regarding shape. These shape-based approaches can be seen from the perspective of grasping in [24]. This approach has been also used by others for different applications such as segmentation and manipulations [25], [26], [27]. These approaches lack performance when they are applied to objects that are difficult to be segmented or not completely cuboid and cylindrical in shape.

To deal with new and unknown shapes of objects, a different approach has been used by some researchers whereby the shape of the object in question is estimated from the data stored recently by the sensor feedback. In [28], an approach is presented where the shape of the object is modeled using a Gaussian process. To achieve successful grasping the same idea has been implemented in [29] using tactile feedback. These approaches fail to perform when inadequate information is available to predict the shape of the model with a high degree of certainty. Accurately estimating shape from available data is an active area of research [30], [31], [32].

Recently, there has been good progress to solve the problem of grasping novel objects [33]. A number of these approaches involve agents that have been trained using supervised learning to predict whether successful grasping has been achieved or not without using geometrical information of the object. For instance, an approach to place new objects at new places without having or estimating geometrical information of the object is presented in [34]. Similar approaches to handle the grasping problem can be also found in [35].

Reinforcement learning has been studied the perspective of its applications in robot control [36]. Over time, approaches based on deep reinforcement learning have become well known and popular for handling manipulations in robotics [37], [9], [38].

This study focuses on the RL-based pick and place solution in a non-visual environment, which has not been much addressed before.

III. Background

A. Reinforcement Learning

Reinforcement learning resolves a task by performing sequential decision-making through a policy learned during the process of maximizing the expected reward. The RL agent observes its' environment state and takes an action in order to earn a reward and transition to the next state. What the agent needs to learn is to take optimal action at every stage so that the total reward i.e., *the return*, should be maximized. As the agent continues to interact with the environment, the resulting random or stochastic process is described by a Markov Decision Process (MDP). This MDP is defined by a tuple ($\mathcal{S}$, A, δ, $\mathbb{R}$, λ) [39]. Here the state space is represented by $\mathcal{S}$, action space is denoted by A, δ is commonly known as the transition function of the environment, $\mathbb{R}$ denotes the reward function deciding the rewards for the agent's selections of actions at any certain state and λ represents the discount factor ranging from 0 to 1 to discount the rewards earned by the agent after the selections of action at any certain state. At any given timestep, the RL agent being at state, $s \in \mathcal{S}$, selects an action from the action space, $a \in A$, according to the policy $\mu(s|\theta_\mu):\mathcal{S}\rightarrow A$ where policy parameters are represented by θ_μ. The environment moves to the next state, s' after the reward is received by the agent, $r = \mathbb{R}(s, a) : \mathcal{S}\times A\rightarrow\mathcal{S}$, with the help of the transition function, $s' = \delta(s, a) : \mathcal{S}\times A\rightarrow\mathcal{S}$. The final objective is to maximize the expected return at each time step t, $\mathbb{E}_{st,at\sim\mu}\sum_t \lambda^{t-1}\mathbb{R}(s_t,a_t)$.

B. Temporal-Difference Learning

Temporal-difference (TD) learning is one of the key ideas of reinforcement learning. Temporal-difference learning can be seen as a blend of ideas from dynamic programming (DP) and Monte Carlo methods. Temporal-difference learning is termed model-free learning because in this type of learning, the agent learns through actual experimentation instead of learning from a model such as a transition table. This factor enables TD to define and work with a large number of state-action pairs. The agent is completely unaware of the outcomes for its actions and learns accordingly after experiences. In temporal-difference learning, the agent learns from each and every action, as the updating is performed at every timestep rather than on completion of each episode. It can be generalized as

$$Estimate \leftarrow Estimate' + StepSize[Target - Estimate'] \quad (1)$$

In the equation above "Target – Estimate'" is also known as the *target error*. Through this equation, we get the "target" because the updating is performed at every timestep. The higher the target value is, the better the state agent into which the agent transitions. So, in nutshell, the agent is left to play in a new world, where it has zero awareness of any states, rewards, and transition tables. Interacting with this world makes it learn by continuously updating its existing knowledge after every interaction. For temporal-difference control methods, we can see both off-policy and on-policy approaches which guide. For this scope of this study, we will review the Q-Learning, an off-policy method., and SARSA, an on-policy method. Off-policy Q-learning method was introduced earlier and subsequently an on-policy variant of this algorithm was introduced which formed the basis of the on-policy SARSA algorithm.

C. Q-Learning

Watkins in 1989 made a great advance in the field of reinforcement learning by developing a temporal difference off-policy model-free control algorithm called Q-learning [39]. In this algorithm, learning is achieved through actions chosen and carried out according to another policy. It can be described through the Bellman equation as follows

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \cdots | s, a] \quad (2)$$

$$Q^{\pi}(s,a) = \mathbb{E}_{s'}[r + \lambda Q^{\pi}(s', a') | s, a] \quad (3)$$

$\mathbb{E}$ represents the expected and discount is denoted by the λ. The main goal of Q-learning is to maximize the Q-value, through policy iteration and value iteration. Policy iteration means a continuous loop of policy evaluation and improvement. In policy evaluation, we use the greedy policy achieved from the last policy improvement to estimate the value function V, whereas the policy is updated with actions that will increase the V to the maximum level for each state in the policy improvement part of the loop. Updating is done through the Bellman equation and the loop continues until the convergence point is reached. With the help of the optimal Bellman equation value function V is updated in value iteration, which can be stated as follows

$$v_*(s) = \max_a \mathbb{E}\left[\mathbb{R}_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a\right] (4)$$

$$v_*(s) = \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')] \quad (5)$$

A general off-policy temporal-difference algorithm flow can be seen in Fig. 1.

D. SARSA

Rummery and Niranjan in 1994 proposed an on-policy version of Q-learning, "Modified Connectionist Q-Learning" (MCQ-L) [40]. Later it was named SARSA, being an acronym for state-action-reward-state-action. SARSA is an on-policy temporal-difference control method, *Q(s, a, r′ ,s′ , a′)* where agent at state *s* takes an action *a* at time t, collects reward *r′*, with next state-action pair *s′, a′* at time *t+1*. Update rule followed in SARSA is as follows

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \lambda[R_{t+1} + Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (6)$$

A general off-policy temporal-difference algorithm flow can be seen in Fig. 2.

IV. Methodology

In a a pick and play systems, the robotic arm has to pick various moving objects on the conveyor belt and place them at a designated location e.g., in a bin. Objects move at different positions and orientations on conveyor belt, i.e., left-aligned, center-aligned, or right-aligned. The conveyor belt may also at different speeds e.g., slow speed, medium speed, fast speed. When we say objects, we mean shapes such as cuboids, spheres, or cylinders. Fig. 3 describes the scene.

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$;
  until s is terminal

Figure 1. An off-policy TD control algorithm

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

Figure 2. An on-policy TD control algorithm

A. MDP Formulation

Our problem can be formulated as an MDP. For our pick and place problem the elements of this MDP can be described as follows:

- s ∈ $\mathcal{S}$: (*C, N, G, speed, position, path*) where *C* is any of the XYZ coordinates depending upon the nature of the action; *N* represents the nature of the action i.e., pre-pick, pick, place; *G* is the set of potential grasping poses detected if any; *speed* represents the speed of the moving object; *position* is the position alignment of the moving object with reference to the conveyor belt,; the *path* being the calculated points of the path, from initial position to pre-pick position or from pre-pick position to pick position, to reach these particular *C* coordinates last time if any.

- a ∈ *A*: (C, *N*, *G*, *path*) where *C* is any of the chosen XYZ coordinates depending upon the nature of the action chosen; *N* represents the action i.e., pre-pick, pick, place; *G* is the set of potential grasping poses detected, the *path*, from initial position to pre-pick position or from pre-pick position to pick position, being the calculated points of the path to reach these particular *C* coordinates.
- $r \in \mathbb{R}$: *r* is the reward and is 1 if the pick and place task is accomplished, 0.5 if only the grasping part is done successfully and 0 otherwise. Some bonus rewards and negative reinforcement learning schemes have also been tried.
- δ: δ is the transition function and is an unknown stochastic function of the RL agent and the environment - as this problem is being dealt with as model-free, a transition function will emerge from trial-and-error sampling [24].
- $\lambda : \lambda$ is the discount factor and is between 0 and 1. Discounting future rewards enables the RL agent to learn faster.

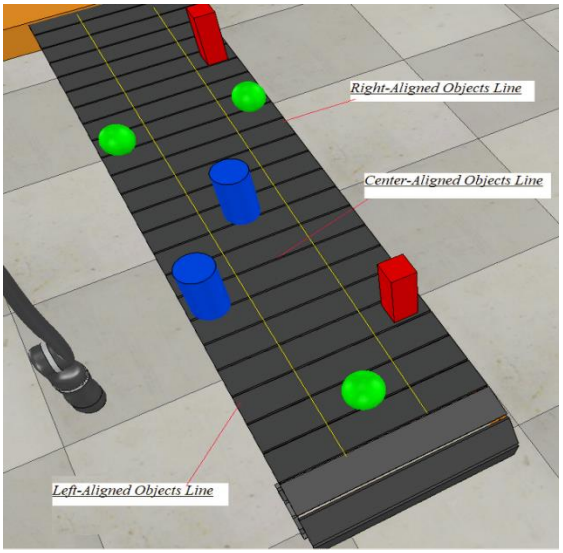

Figure 3. Conveyor belt scene

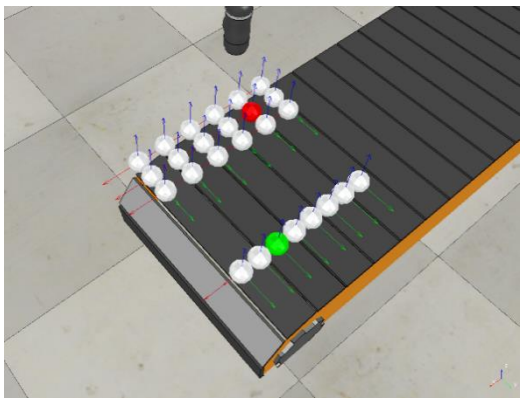

Figure 4. Pre-Pick and Pick XYZ Coordinates

In order to improve learning opportunity, we divided the task of picking an object into two parts, i) firstly moving the arm to a coordinate near the object and ii) then moving via a linear path from the first position to pick the object. We have divided the overall approach into four phases. The first phase is called the 'Initial Phase', in which the robotic arm is at the rest position, waiting for the proximity sensors installed at the conveyor belt to detect the coming object and signal the presence, position, and speed of the object. Once the signal is received, now the robotic arm enters the phase called 'Pre-Pick Phase', by selecting one of the potential pre-pick XYZ coordinates and calculating the path using motion planning to make the end-effector reach the selected pre-pick coordinates. After reaching the chosen pre-pick coordinates, now the third phase called 'Grasp & Pick Phase' begins. In this phase, one of the potential pick XYZ coordinates is chosen, the linear path calculation is done, and grasping is performed after choosing the potential grasp pose. Fig. 4 below helps illustrates the movement from 'pre-pick' to 'pick' coordinates more easily. Each ball-like object represents a potential XYZ coordinate. XYZ Coordinates on the left-hand side (three adjacent rows) are potential pre-pick coordinates and the coordinates on the right-hand side (single row) are pick coordinates. This figure is used to describe the concept- the number and positions of coordinates in the actual implementation are different. So according to Fig. 4, our agent selected the red pre-pick XYZ coordinate and the green pick XYZ coordinate. Subsequently the robotic arm gripper will calculate the path to reach the red coordinate and then once it reaches this point successfully it will calculate the path to reach the green coordinate and advance for the grasping.

Once the grasping is done successfully, the RL agent enters the last phase of the iteration, called "Place Phase", wherein it calculates the path using motion planning to make the end-effector reach the placing position XYZ coordinates and release the gripper to place the object. This whole loop of the four phases continues as shown in Fig. 5. Rewards (0 and 1) are calculated on the basis of the end result i.e., successful pick and place or not. The agent has been trained and tested with negative reinforcement learning also. A reward bonus scheme was also tested. This scheme gave an additional reward given for choosing the best hand-marked coordinates for picking the object from the conveyor belt. A comparison of these reward schemes is shown in the results section.

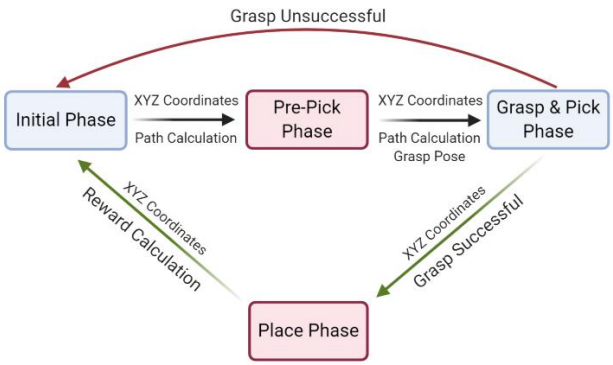

Figure 5. MDP overview

The basic idea behind the formulation of the MDP for this episodic task of pick and place is to compute an

optimal policy by trial-and-error in order to maximize the later discounted reward.

$$E_t = r_t + \lambda r_{t+1} + \lambda^2 r_{t+2} \dots \lambda^{T-1} r_T \quad (7)$$

This methodology and reward schemes are implemented using both off-policy Q-learning and on-policy SARSA algorithms. How they differ technically is discussed in the next sub-section.

B. Q-Learning vs SARSA

As this study compares off-policy and on-policy temporal-difference control-based approaches, it is important to understand the key differences between them. In off-policy algorithms, agents tend to converge to an optimal policy by updating a policy different from the behavior policy. In this way, without actually following any greedy policy, the off-policy algorithm estimates an expected reward and allocates a new value to the expected new state [39]. On other hand, in on-policy algorithms, the agent sticks to the same optimal policy and follows it throughout. The policy that is being used for updating is the same policy that is being followed to choose actions in contrast to off-policy algorithms [39].

When we examine the pseudocode of the Q-learning and SARSA algorithms, we notice similarities but also certain differences [41]. The main difference between Q-Learning and SARSA is in the way they update their Q-values in the Q-table. In Q-learning, the updating of the Q-table is done by selecting the best available action in the next state agent has transitioned to. In SARSA, on the other hand, a new reward value is generated by selecting an action by following the same policy. In SARSA the future action selection will not always be perfect, and in case of the presence of some unwanted states, the agent could end up in any such undesired state. Therefore, SARSA follows a safer pattern to minimize the chances of any such transitions that can land the agent in an undesired state. Q-learning, however, doesn't take this probability of the agent ending up in an undesired state into account and assumes to do action selection solely based on the Q-values of the state in the Q-table. This difference of behavior between Q-learning and SARSA can be clearly seen in [39] when they are applied to the famous cliff-walking problem.

C. Motion Planning

Many motion planning and control solutions are available for robotics. We considered different options to perform our motion planning task such as OpenRave and Trajpot [42], [43]. Open Motion Planning Library (OMPL) [44] proved to be the best approach as it provided us with a high degree of customization. OMPL contains a number of geometric and control-based planners. Some of the many sampling-based planners available are Single-query Bi-direction Lazzy (SBL), Expansive Space Trees (EST), Rapidly-exploring Random Trees (RRT), Probabilistic Roadmap Method (PRM) along with their many variants. The planner we used for motion planning and path calculations is a single-query planner, a bidirectional variant of Rapidly-exploring Random Trees (RRT), known as the RRT-Connect [45]. The key idea of RRT-Connect is to develop two RRT, one at the start point and the other at the end, then connecting them. For this reason, RRT-Connect planner tends to outperform the RRT planner.

D. V-REP

The Virtual Robot Experimentation Platform (V-REP) is a 3D robotic simulator with an integrated development and coding support [46]. It also uses physics engines Bullet and ODE for real-time emulation of the objects involved in the simulation. The API and threaded/non-threaded Lua scripting functionalities make it a good choice for combining multiple platforms such as python, java, C++ etc. for experimentation. Using this API, we were able to make our python-based RL agent communicate with the Lua-scripted simulated environment.

V-REP provides various calculation modules including the forward and inverse kinematics module. Forward kinematics means using kinematic equations, taking joints parameters as input, to calculate the position of end-effector, whereas inverse kinematics is the reverse process, calculating joint parameters for a given position of end-effector [47]. The collision detection module is another important module in V-REP to highlight collisions if any. We used the JACO robotic arm [48] for our experiments. This is a six (6) degrees of freedom robotic arm, and we used an RG2 gripper for grasping purposes. Fig. 6 shows various grasp poses for various objects from our approach.

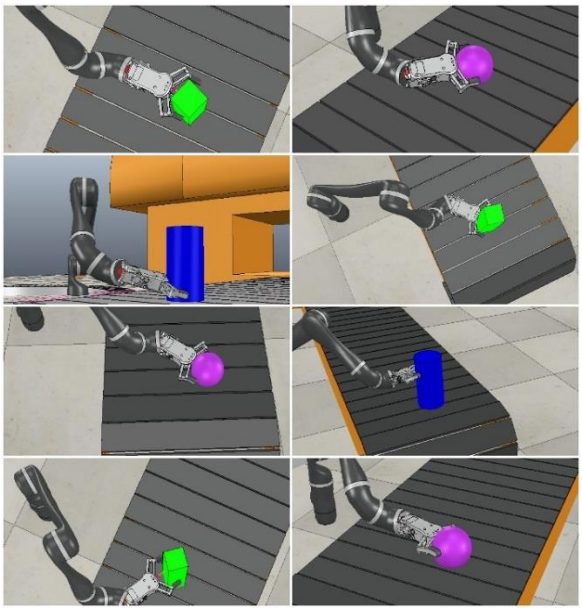

Figure 6. Various grasping poses

V. Experimental Results & Discussion

The Q-learning and SARSA RL agents described in the previous section were extensively trained and evaluated in our experimentation phase.

A. Training & Testing on Individual Shapes

Firstly, we trained and evaluated the performance of the Q-learning and SARSA agents individually for each shape. Later we also evaluated the performance of off-policy and on-policy agent on random objects at random position alignments moving at a random speed in order to get a better view of the performance.

While dealing with each object's shape individually, we also trained and evaluated both off-policy and on-policy agents for each of the speeds (slow, medium, fast) and position alignment (left, center, right) separately. The success rates of the Q-learning agent picking an object from the conveyor belt and placing it at a designated place position, for each object type, at each position alignment, while moving at three different speeds (slow, medium, fast) are shown in Tables I, II, and III respectively.

We can clearly see from Table I, II, and III that the Q-learning RL agent has learned throughout the training phase elements such as suitable XYZ coordinates for picking and placing, different position alignments, speeds, and shapes of the objects. We can draw a number of inferences from these results such as that the agent seems to struggle with pick and place of spherical objects as compared to others. This holds true for all speeds.

TABLE I. Q-LEARNING AGENT'S INDIVIDUAL SUCCESS RATE (%) AT SLOW SPEED

Objects	Training			Testing		
	Left	*Center*	*Right*	*Left*	*Center*	*Right*
Cuboid	89%	95%	91%	95%	99%	96%
Cylinder	92%	91%	88%	94%	93%	93%
Sphere	83%	88%	85%	92%	95%	91%

TABLE II. Q-LEARNING AGENT'S INDIVIDUAL SUCCESS RATE (%) AT MEDIUM SPEED

Objects	Training			Testing		
	Left	*Center*	*Right*	*Left*	*Center*	*Right*
Cuboid	91%	93%	93%	96%	98%	96%
Cylinder	89%	94%	90%	96%	99%	97%
Sphere	84%	90%	87%	93%	94%	94%

TABLE III. Q-LEARNING AGENT'S INDIVIDUAL SUCCESS RATE (%) AT FAST SPEED

Objects	Training			Testing		
	Left	*Center*	*Right*	*Left*	*Center*	*Right*
Cuboid	91%	95%	96%	97%	99%	95%
Cylinder	93%	91%	90%	95%	95%	96%
Sphere	78%	83%	85%	90%	91%	93%

In the same way, the Q-learning agent has shown better performance with all objects that were center-aligned at medium speed i.e., the most suitable coordinates and grasp poses were found and learned by the agent for the center-aligned objects moving at medium speed. Testing results clearly depict the level of learning achieved during the training phases.

The results of the SARSA agent picking an object from the conveyor belt and placing it at a designated place position, for each object type, at each position alignment, while moving at three different speeds (slow, medium, fast) are shown in Tables IV, V, and VI respectively.

Results shown in Table IV, V and VI clearly indicate that, as with the Q-learning agent, our on-policy SARSA agent has also managed to learn suitable XYZ coordinates for picking and placing, different position alignments, speeds, and shapes of the objects during the training phase. The overall results shows that our SARSA agent has not performed as well as our Q-learning agent. We can also see the deterioration in the performance of the SARSA agent as the speed of objects increased. Similar to the Q-learning agent, the SARSA agent also has difficulty to deal with spherical objects during training and testing phases. The results also highlight that the SARSA agent has performed well in the case of left and center-aligned objects at all speeds as compared to the objects which were right-aligned.

TABLE IV. SARSA AGENT'S INDIVIDUAL SUCCESS RATE (%) AT SLOW SPEED

Objects	Training			Testing		
	Left	*Center*	*Right*	*Left*	*Center*	*Right*
Cuboid	83%	81%	80%	82%	82%	80%
Cylinder	82%	80%	78%	80%	81%	78%
Sphere	80%	79%	79%	78%	79%	77%

TABLE V. SARSA AGENT'S INDIVIDUAL SUCCESS RATE (%) AT MEDIUM SPEED

Objects	Training			Testing		
	Left	*Center*	*Right*	*Left*	*Center*	*Right*
Cuboid	81%	78%	75%	79%	77%	73%
Cylinder	80%	78%	76%	78%	80%	77%
Sphere	77%	77%	73%	78%	77%	72%

TABLE VI. SARSA AGENT'S INDIVIDUAL SUCCESS RATE (%) AT FAST SPEED

Objects	Training			Testing		
	Left	*Center*	*Right*	*Left*	*Center*	*Right*
Cuboid	79%	78%	72%	78%	78%	71%
Cylinder	77%	78%	73%	78%	77%	72%
Sphere	75%	72%	71%	76%	71%	69%

B. Training & Testing on Random Objects

After training and testing specific objects at individual position alignments and speeds, we proceeded to train and evaluate the performance of the both Q-learning and SARSA RL agents on episodes of random objects at random position alignments moving at a random speed. In order to evaluate the performance of the RL agents in this manner, we designed 5 test cases, each having 50 iterations/objects. In test case 1, each alignment position was assigned one-third of the random objects moving at a random speed. In test case 2, each speed option was assigned to one-third of the random objects at random position alignments. Test cases 3, 4, and 5 had at least 40% cuboid, sphere, and cylinder objects respectively moving at a random speed at random alignment positions. The testing process can be seen in Fig. 7. The success rate for these test cases for both off-policy and on-policy agents is shown in Table VII. The average success rate for Q-learning results shown in Table VII is around 93%. Our Q-learning RL agent has performed well in all test cases but comparatively lowly in test case 4. The reason is that test case 4 includes a greater proportion oof spherical objects more than 40%. As the RL agent performance was seen to be poorer in individual testing in the case of spherical objects, it also affected our random testing in test case 4. On the other hand, the average success rate for SARSA agent according to Table VII is around 80%. As might be expected the SARSA agent also performed least well in test case 4. In overall performance, it can be clearly witnessed that our Q-learning agent outperformed the SARSA agent. The fact that the SARSA agent failed to perform at the higher belt speed and also on right-aligned objects are the major factors in its poorer showing.

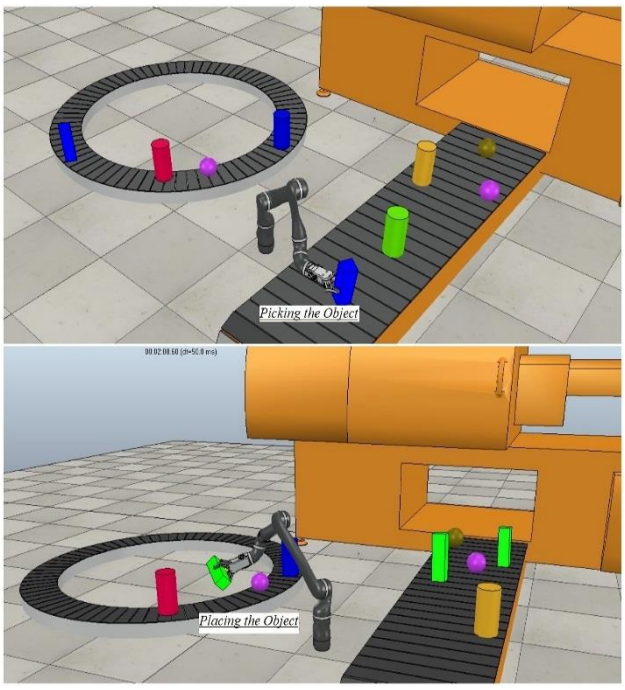

Figure 7. Training process

TABLE VII. RANDOM TEST-CASES SUCCESS RATE (%)

Test Case	Q-Learning Agent's Success Rate	SARSA Agent's Success Rate
Test Case 1	93%	82%
Test Case 2	95%	81%
Test Case 3	99%	80%
Test Case 4	83%	77%
Test Case 5	97%	81%

A comparison of Q-learning and SARSA agents' performance in random testing of up to 3000 timesteps is given in Fig. 8, where the success rate (%) has been plotted against the number of steps. The figure clearly indicates that the Q-learning agent outperforms the SARSA agent. In the initial time steps, it seemed SARSA agent is having an edge over Q-learning agent, maybe due to practicing more exploitation over exploration, but gradually the scenario is reversed.

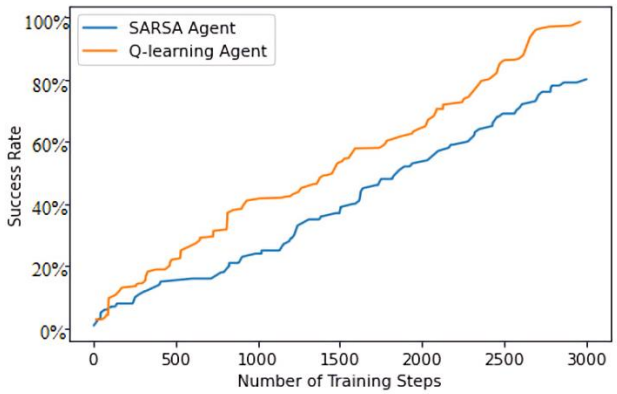

Figure 8. Performance comparison of Q-learning and SARSA agents

C. Reward Scheme Variations

As we discussed earlier in the methodology section, we trained both off-policy and on-policy agents with a reward system of 1 if the pick and place task is accomplished, 0.5 if only the grasping part is done successfully but failed to place it at the designated location, otherwise 0.

To experiment with the reward scheme, we also trained and tested a variant of our both Q-learning and SARSA agents with negative reinforcement learning where -1 was awarded on failing the task completely and a bonus reward of 1 for some best suitable coordinate's selection. Fig. 9 shows the performance comparison between the Q-learning agent with normal reward scheme and its variant trained with negative reinforcement learning and in the same manner, Fig. 10 shows the same for the SARSA agent. The blue line in both diagrams represents the agent trained with the normal reward scheme while the orange line represents the variant trained with negative rewarding as well as bonus rewarding scheme.

In the case of the Q-learning agent, Fig. 9 reveals better performance of the variant trained with negative reinforcement learning. However, for the case of the SARSA agent, Fig. 10 shows the ups and downs of both normal reward scheme agent and negative reward scheme agent. This comparison clearly shows that the negative reward scheme in Q-Learning performs better over the same number of training steps.

D. Discussion

We trained and tested our both Q-learning agent and SARSA agent on the same testbed. Agents were trained and tested for both individual shapes task and random shapes task. In all cases the Q-learning agent outperformed the SARSA agent in our testing.

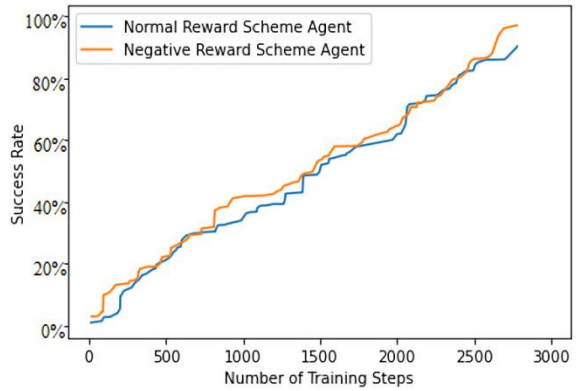

Figure 9. Performance comparison of Q-learning normal reward scheme agent and the negative reward scheme agent

The SARSA agent has faced difficulties in maintaining success as the speed of the conveyor belt increases. Its lack of performance can be also be witnessed in the case of all right-aligned objects for both individual and random shapes. Spherical objects remained problematic for both our agents but the Q-learning agent still managed to deal with them better than the SARSA agent.

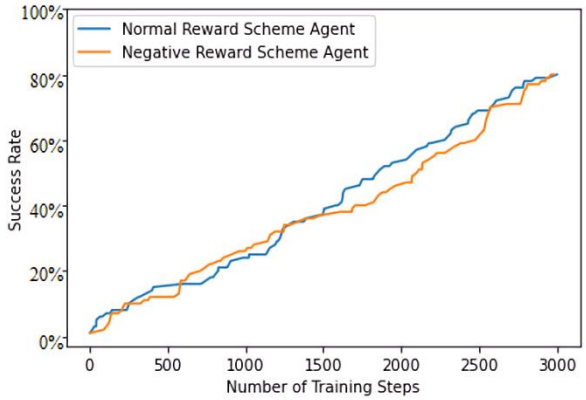

Figure 10. Performance comparison of SARSA normal reward scheme agent and the negative reward scheme agent

The reason behind the lack of performance in our experimentation by SARSA in comparison to Q-learning can be seen as the exploration-exploitation dilemma. SARSA follows a safer pattern to minimize the chances of any such transitions that can land the agent in an undesired state while earning itself a large negative reward. But by doing so, it becomes conservative and loses chances of exploration and finding optimal options to move forward. Meanwhile, the Q-learning agent doesn't take this probability of the agent ending up in an undesired state into account much and assumes to do action selection solely based on the Q-values of the state in the Q-table. This difference of behavior led to the optimal convergence and better performance of Q-learning agent over the SARSA agent.

VI. Conclusion

In this paper, we have presented an approach to address the problem of industrial robotic pick and place in a non-visual environment. We formulated the problem as an MDP for which Reinforcement Learning provides a very extensive framework for dealing with such tasks. We deployed both model-free off-policy temporal difference RL algorithm (Q-Learning) and on-policy temporal difference RL algorithm (SARSA). We trained and tested Q-learning and SARSA agents on different shaped objects at different position alignments moving at different speeds.

We designed test-cases to evaluating the performance of our Q-learning and SARSA agents on different objects, which resulted in an average success rate of 93 percent and 80 percent respectively. We also improved the performance by retraining the Q-learning and SARSA agents with negative and bonus rewards.

In this paper, we present a novel approach that allows learning and performing of pick and place operation in a non-visual industrial environment where deployment of vision-based system is difficult due to various requirements such as high cost which can restrict its use. For instance, the space available for installing an industrial vision system may be only a few centimeters. Other factors that can affect the use of vision system in some particular settings can be vibrations, dust, or even wash-up from water jets. Known limitations of our approach are the excessive time periods required for training for various objects with different parameters (about 4 to 5 hours for each object class on Intel Core i7-3770 @ 3.40 GHz with 12 GB RAM), run-time path computation overhead, difficulties to understand the geometric features of certain objects like spheres due to the absence of vision sensors in the setup.

So, for future work, we plan to address these limitations by using a camera, and deploying deep reinforcement learning to handle the state and action complexity. In order to increase the efficiency, a hybrid approach combining on-policy and off-policy algorithms such as backward Q-learning presented in [49] and [50] is also being worked on. We also plan to explore the possibility of deploying a multi-query planner for motion planning instead of a single-query planner in order to have multiple options computed on the run, hence increasing the chances of higher efficiency.

Conflict of Interest

The authors declare no conflict of interest.

ACKNOWLEDGMENT

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number SFI 16/RC/3918, co-funded by the European Regional Development Fund.

REFERENCES

[1] B. Melzer, "Reference architectural model industrie 4.0 (RAMI 4.0)," p. 15.
[2] "Pcast-advanced-manufacturing-june2011.pdf." .
[3] C. C. Kuo, J. Z. Shyu, and K. Ding, "Industrial revitalization via industry 4.0–A comparative policy analysis among China, Germany and the USA," *Glob. Transit.*, vol. 1, pp. 3–14, 2019.
[4] Y. Harayama, "Society 5.0: Aiming for a new human-centered society," p. 6.
[5] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.
[6] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[7] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *Proc. 2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, May 2019, pp. 3651–3657.
[8] OpenAI *et al.*, "Learning dexterous in-hand manipulation," *ArXiv180800177 Cs Stat*, Jan. 2019, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1808.00177.
[9] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *ArXiv150400702 Cs*, Apr. 2016, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1504.00702.
[10] I. Popov *et al.*, "Data-efficient deep reinforcement learning for dexterous manipulation," p. 12.
[11] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," *ArXiv170910089 Cs*, Feb. 2018, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1709.10089.
[12] M. Andrychowicz *et al.*, "Hindsight experience replay," *ArXiv170701495* Cs, Feb. 2018, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1707.01495.
[13] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. J. Johnson, and S. Levine, "SOLAR: Deep structured representations for model-based reinforcement learning," *ArXiv180809105 Cs Stat*, Jun. 2019, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1808.09105.
[14] M. Riedmiller *et al.*, "Learning by playing - solving sparse reward tasks from scratch," *ArXiv180210567 Cs Stat*, Feb. 2018, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1802.10567.
[15] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," *ArXiv170505363 Cs Stat*, May 2017, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1705.05363.
[16] N. Savinov *et al.*, "Episodic curiosity through reachability," *ArXiv181002274 Cs Stat*, Aug. 2019, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1810.02274.
[17] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," p. 10.
[18] D. Morrison, J. Leitner, and P. Corke, "Closing the loop for robotic grasping: a real-time, generative grasp synthesis approach," presented at the Robotics: Science and Systems 2018, Jun. 2018, doi: 10.15607/RSS.2018.XIV.021.
[19] A. Zeng *et al.*, "Multi-view self-supervised deep learning for 6D pose estimation in the Amazon picking challenge," in *Proc. 2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, Singapore, May 2017, pp. 1386–1383.
[20] A. Zeng *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," *ArXiv171001330 Cs*, May 2020, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1710.01330.
[21] Q. Shao *et al.*, "Suction grasp region prediction using self-supervised learning for object picking in dense clutter," in *Proc. 2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*, Singapore, May 2019, pp. 7–12.
[22] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *Proc. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Oct. 2018, pp. 4238–4245.
[23] I. Sarantopoulos, M. Kiatos, Z. Doulgeri, and S. Malassiotis, "Split deep q-learning for robust object singulation," in *Proc. 2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, May 2020, pp. 6225–6231.
[24] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Proc. 2003 IEEE* International *Conference on Robotics and Automation (Cat. No.03CH37422)*, Taipei, Taiwan, 2003, pp. 1824–1829.
[25] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in human environments," p. 7.
[26] K. Harada, K. Nagata, T. Tsuji, N. Yamanobe, A. Nakamura, and Y. Kawai, "Probabilistic approach for object bin picking approximated by cylinders," p. 6.
[27] T. Morwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze, "BLORT - The blocks world robotic vision toolbox," p. 8.
[28] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *Proc. 2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 2845–2850.
[29] J. Mahler *et al.*, "GP-GPIS-OPT: Grasp planning with shape uncertainty using Gaussian process implicit surfaces and Sequential Convex Programming," in *Proc. 2015 IEEE* International *Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA, May 2015, pp. 4919–4926.
[30] S. Hinterstoisser *et al.*, "Gradient response maps for real-time detection of textureless objects," *IEEE Trans Pattern Anal Mach Intell*, vol. 34, no. 5, pp. 876–888, May 2012.
[31] K. Pauwels and D. Kragic, "SimTrack: A simulation-based framework for scalable real-time object pose detection and tracking," in *Proc. 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, Sep. 2015, pp. 1300–1307.
[32] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3D pose estimation," in *Proc.2015 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR*, Jun. 2015.
[33] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis-a survey," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 289–309, Apr. 2014.
[34] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, *Learning to Place New Objects in a Scene*. 2012.
[35] M. Gualtieri, A. Ten Pas, and R. Platt, *Pick and Place without Geometric Object Models*, 2018.
[36] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement Learning in Robotics: A Survey," p. 38.
[37] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, *Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection*. 2016.
[38] U. Viereck, A. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using easily simulated depth images," 2017.
[39] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," p. 352.
[40] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," p. 22.
[41] M. Tokic and G. Palm, "Value-difference based exploration: adaptive control between epsilon-greedy and softmax," in *Proc. KI 2011: Advances in Artificial Intelligence*, vol. 7006, J. Bach and S. Edelkamp, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 335–346.
[42] R. Diankov, "Automated construction of robotic manipulation programs," p. 263.
[43] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," presented at the Robotics: Science and Systems 2013, June 2013.

[44] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.
[45] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, San Francisco, CA, USA, 2000, vol. 2, pp. 995–1001.
[46] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira, "Virtual robot experimentation platform V-REP: A versatile 3D robot simulator," in *Simulation, Modeling, and Programming for Autonomous Robots*, vol. 6472, N. Ando, S. Balakirsky, T. Hemker, M. Reggiani, and O. von Stryk, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 51–62.
[47] Z. Zou, J. Han, and M. Zhou, "Research on the inverse kinematics solution of robot arm for watermelon picking," in *Proc. 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chengdu, Dec. 2017, pp. 1399–1402, doi: 10.1109/ITNEC.2017.8285026.
[48] K. Iturralde, T. Kinoshita, and T. Bock, "Grasped element position recognition and robot pose adjustment during assembly," presented at the 36th International Symposium on Automation and Robotics in Construction, Banff, AB, Canada, May 2019.
[49] Y. H. Wang, T. H. S. Li, and C. J. Lin, "Backward Q-learning: The combination of Sarsa algorithm and Q-learning," *Eng. Appl. Artif. Intell.*, vol. 26, no. 9, pp. 2184–2193, 2013.
[50] X. Li, Z. Lv, L. Wu, Y. Zhao, and X. Xu, "Hybrid online and offline reinforcement learning for Tibetan Jiu chess," *Complexity*, vol. 2020, p. 4708075, May 2020.

Muhammad Babar Imtiaz is currently a doctorate scholar funded by Science Foundation Ireland (SFI) CONFIRM Smart Manufacturing Center, at Software Research Institute (SRI) at Athlone Institute of Technology (AIT). Previously, his Master's degree in Computer Sciences MS (CS) is with distinction from The Islamia University Bahawalpur, Pakistan in 2017. His Bachelor degree in Computer Sciences BS(CS) is from The Islamia University Bahawalpur, Pakistan in 2015. His area of research includes robotics, smart manufacturing and artificial intelligence.

Yuansong Qiao is a Senior Research Fellow in the Software Research Institute (SRI) at Athlone Institute of Technology (AIT) Ireland. He is a Science Foundation Ireland (SFI) Funded Investigator in the SFI CONFIRM Smart Manufacturing Centre. He received his Ph.D. in Computer Applied Technology from the Institute of Software, Chinese Academy of Sciences (ISCAS), Beijing, China, in 2008. He completed a B.Sc. and an M.Sc. in Solid Mechanics from Beihang University, Beijing, China in 1996 and 1999 respectively. He is a member of IEEE (Communications, Computer and Robotics and Automation societies and Blockchain Community) and ACM (SIGCOMM and SIGMM). His research interests include Future Internet Architecture, Smart Manufacturing, Blockchain Systems, IoT Systems, and Edge Intelligence and Computing.

Brian Lee is currently the Director of the Software Research Institute (SRI), Athlone Institute of Technology (AIT), Athlone, Ireland. He is a Science Foundation Ireland (SFI) Funded Investigator in the SFI CONFIRM Smart Manufacturing Centre. He received the Ph.D. degree in application of programmable networking for network management from the Trinity College Dublin, Dublin, Ireland. He has more than 25 years of research and development experience in telecommunications network monitoring, their systems and software design, and the development for large telecommunications products with very high impact research publications. He was the Director of Research for LM Ericsson, Ireland, with responsibility for overseeing all research activities, including external collaborations and relationship management. He was an Engineering Manager of Duolog Ltd., where he was responsible for strategic and operational management of all research and development activities.