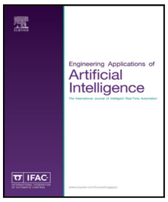
Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Advising reinforcement learning toward scaling agents in continuous control environments with sparse rewards[☆]

Hailin Ren, Pinhas Ben-Tzvi[*]

Robotics and Mechatronics Lab, Mechanical Engineering Department, Virginia Tech, Blacksburg, VA 24061, USA

ARTICLE INFO

Keywords:
Reinforcement learning
Advising framework
Continuous control
Sparse reward
Multi-agent

ABSTRACT

This paper adapts the success of the teacher–student framework for reinforcement learning to a continuous control environment with sparse rewards. Furthermore, the proposed advising framework is designed for the scaling agents problem, wherein the student policy is trained to control multiple agents while the teacher policy is well trained for a single agent. Existing research on teacher–student frameworks have been focused on discrete control domain. Moreover, they rely on similar target and source environments and as such they do not allow for scaling the agents. On the other hand, in this work the agents face a scaling agents problem where the value functions of the source and target task converge at different rates. Existing concepts from the teacher–student framework are adapted to meet new challenges including early advising, importance of advising, and mistake correction, but a modified heuristic was used to decide on when to teach. The performance of the proposed algorithm was evaluated using the case study of pushing, and picking and placing objects with a dual arm manipulation system. The teacher policy was trained using a simulated scenario consisting of a single arm. The student policy was trained to handle the dual arm manipulation system in simulation under the advice of the teacher agent. The trained student policy was then validated using two Quanser Mico arms for experimental demonstration. The effects of varying parameters on the student performance in the advising framework was also analyzed and discussed. The results showed that the proposed advising framework expedited the training process and achieved the desired scaling within a limited advising budget.

1. Introduction

After obtaining great success in performing basic tasks such as solving classic control problems, designing stable legged locomotion gaits, and playing classical Atari games (Brockman et al., 2016), reinforcement learning (RL) associated with deep neural networks has been gaining success in solving highly nonlinear problems without using explicit mathematical modeling. These problems are not suited for hand-engineered/heuristic solutions and are hard to be expressed through explicit mathematical models, making them difficult to be solved using traditional approaches. For instance, a physical Shadow Dexterous Hand could perform vision-based object reorientation solely based on simulated training without any demonstration (Andrychowicz et al., 2018). This approach surpassed all existing non-learning-based approaches. Another work (Riedmiller et al., 2018) explored the use of active scheduling and execution of auxiliary policies to perform a sequence of correct actions to complete tasks with sparse rewards using a Kinova. Apart from performing complicated single agent tasks, reinforcement learning was also applied in multi-agent tasks, where multiple robots cooperate with each other to complete one specific task with action optimization Munemasa et al. (2018). Despite these great successes, solving complicated problems and achieving shorter convergence time remain major challenges in the domain of reinforcement learning research.

This work focuses on addressing the challenges associated with training multiple agents to collaboratively solve a given problem. This problem hereon referred to as "scaling agents problem" in this paper, comes up mainly when trying to transfer knowledge from a single agent to multiple agents. Existing research has focused on teaching a single agent to complete tasks with a high level of complexity. However, complicated tasks in general require multiple agents to cooperate, which requires learning toward scaling agents. The different existing techniques to control multi-agent systems can be broadly classified into three categories: (1) **training the agents from scratch**, (2) **building a hierarchical structure** and reusing a well-trained single agent neural network and (3) **transfer learning approach** which involves transferring the knowledge from a well-trained agent in an old environment to a new training agent in a new environment so that it is provided with

[☆] No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2020.103515.

[*] Corresponding author.
E-mail addresses: hailin@vt.edu (H. Ren), bentzvi@vt.edu (P. Ben-Tzvi).

https://doi.org/10.1016/j.engappai.2020.103515
Received 25 January 2019; Received in revised form 28 November 2019; Accepted 23 January 2020
Available online xxxx

a basic understanding to quickly start the learning process. There are many methods that address the above problem, but a comprehensive discussion summarizing all of the existing work in this domain is beyond the scope of this work. Interested readers should refer to Silva and Costa (2019) for more details.

Training from scratch without any guidance will be computationally expensive, especially as more agents become involved in the target tasks. Based on existing literature, learning from scratch can gain faster convergence either through a careful design of the reward function (Pong et al., 2018; Popov et al., 2017) or through demonstration data that could be expensive to be collected (Ghalamzan and Ragaglia, 2018). Using model guidance (Levine et al., 2015), or inverse RL (Vasquez et al., 2014) could also benefit learning to perform complicated tasks. **Building a hierarchical structure** is also a traditional approach, for example, a global and local planner/controller will be built and connected together. The global planner/controller (high-level scheduler) generates a sequence of tasks according to an overall schedule, while the local one solves the planned sub-tasks in a small time period (Riedmiller et al., 2018). **The transfer learning approach** aims to reduce the need for samples from the target tasks by using prior knowledge obtained from the source tasks. Existing work succeeded in generalizing or imitating well-trained behaviors from prior experience and gaining faster learning in the target tasks (Parisotto et al., 2016; Ho and Ermon, 2016; Gupta et al., 2017); however, existing work focuses majorly on learning among single agent tasks.

Each of the above-mentioned approaches have their own specific strengths and weaknesses. As such, the specific application requirements needs to be taken into consideration before choosing any of the above methods. The motivating application behind this work is semi-autonomous victim extraction from disaster scenarios. Casualties in natural and man-made disaster scenarios are often in need of immediate evacuation and medical attention. Autonomous and/or semi-autonomous rescue robotic systems such as the Semi-Autonomous Victim Extraction Robot (SAVER) (Williams et al., 0000) is an ideal solution in such scenarios. Use of rescue robotic systems helps to minimize the risk to the lives of human rescue personnel. The overall SAVER concept is shown in Fig. 1, where a robotic mobile stretcher drives up to a casualty and then performs casualty pose manipulation and extraction based on high level instructions given by a remote operator. In order to realize safe casualty interaction, this work will focus on developing a framework to teach a dual-arm manipulation system using a well-trained single arm manipulation system. As an initial step in this direction, the proposed work aims to enable dual-arm pick and place tasks as well as push tasks without internal collisions between the arms, using a trained single arm architecture.

Training a dual-arm manipulation system from scratch may fail or become computationally expensive when using dedicated reward functions as well as the collection of demonstration data. Applying a hierarchical structure needs large memory for both global and local planner/controller, which in turn limits real-life applications. Based on the specific requirements of the application at hand, this work explores the application of transfer learning approach to achieve faster convergence on dual-arm learning.

The proposed transfer learning framework is based on teacher–student frameworks, which help a knowledgeable agent to teach a new agent to perform specific tasks. The teacher–student framework in reinforcement learning was first introduced by Torrey et al. along with a set of heuristic teaching algorithms which only require an agreement on the action set between teachers and students, and allows different state representations (Torrey and Taylor, 2013). The convergence of these teaching algorithms is guaranteed even if using sub-optimal teacher policy (Zhan and Taylor, 2015). To avoid the manual parameter tuning inside the heuristic teaching algorithms, learning-based teaching algorithms were studied to determine when to give advice to the student agent (Zimmer et al., 2014; Fachantidis et al., 2017). Fachantidis et al. further explored the impact of the reward factor on the students' learning performance using learning-based teaching algorithms (Fachantidis et al., 2017). Even though learning-based teaching algorithms do not require manual parameter tuning, the computational expense associated with these methods is still high. Compared to the slight improvement obtained from using learning-based teaching algorithms, this work explores heuristic-based approaches. Besides applying the teacher heuristics in which the teacher decides when to provide advice, various student heuristics were also explored as interactive training strategies (Amir et al., 2016). However, most of the student heuristics performed worse than the teaching heuristics. Multiple agents learning from each other was also studied in Silva and Costa (2019), Leno Da Silva et al. (2017) and Omidshafiei et al. (2019). Knowledge of individual agents to perform the same task in a shared environment can be transferred to each other and thereby achieve a faster learning and cooperative behavior. This is similar to the idea of distributed learning using multiple threads. All the frameworks mentioned above are designed for learning among single agents. This work is proposed to improve the learning performance of scaling agents. The main contributions of this paper are as follows:

- We extend the success of the teacher–student framework in transferring knowledge between single agents toward multi-agents problem. Different approaches are proposed and compared in two case studies using an experimental setup with a dual-arm manipulation system.
- Compared to existing work that focus on performing discrete-control tasks, this work proposes procedures in the teacher–student framework to solve continuous-control tasks.
- Experimental evaluation of the proposed framework is also provided. Based on the results, the proposed framework has a faster learning rate in the training process as compared to training from scratch. This makes it more suitable for real-life robotic applications.

The rest of paper is organized as follows. Section 2 introduces the background of RL along with the RL algorithms and techniques used in this paper. The related work on teacher–student framework with a budget is also presented. Section 3 presents the proposed teacher–student framework toward scaling reinforcement learning agents. Following that, implementations and results of the simulation and experiments are presented and discussed in Section 4. Finally, Section 5 concludes the work with directions for future research.

2. Preliminaries

In this section, we introduce the reinforcement learning background and related algorithms that we developed in this paper. To perform continuous control and stable learning, we chose an off-policy and policy-gradient actor–critic algorithm as the main method used in this work. Hindsight Experience Replay is presented to improve positive sampling in the sparse reward environment. One teacher–student framework with a budget, which our work is based on, is also described in this section.

2.1. Reinforcement learning

Considering a standard reinforcement learning setup, an agent interacts with an environment E in discrete time-steps, which can be formalized as a Markov Decision Process (MDP) and defined by a tuple $(S, A, T, R, \rho, \gamma)$, where S is a set of states that describes the environment, E, and is assumed equal to the observation, O. In this work, A is a set of actions via which the agent interacts with the environment. $T : S{\times}A{\times}S \to [0, 1]$ presents the distribution of transition to the next state, $s' \in S$, after taking an action $a \in A$ from the state $s \in S$. $R : S{\times}A \to \mathbb{R}$ is the reward that the agent receives when starting from the state s and taking the action a, $\rho : S \to [0, 1]$ is the distribution of the initial state s_0, and $\gamma \in [0, 1]$ is the discounted factor for the reward which determines the importance of the short-term reward over

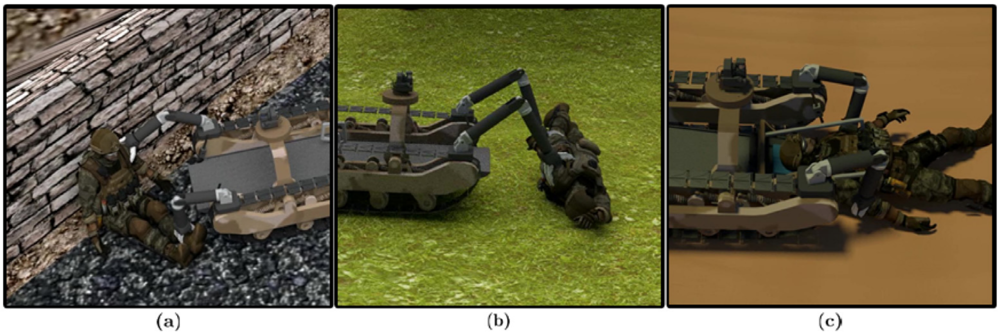


Fig. 1. Scenarios of Human–Robot Interaction using dual-arm manipulation for human extraction purposes: (a) dual arm for human pose manipulation, (b) dual arm for human body rolling, (c) dual arm for human limbs manipulation during extraction.

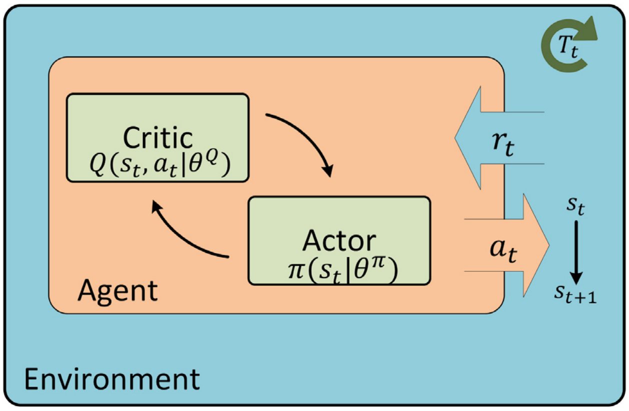


Fig. 2. Diagram of actor–critic method in reinforcement learning.

the long-term ones. The accumulated discounted reward, R_t, is defined as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where $r_{t'} \in R$ is the reward received at time t' and T is the time step at which the learning episode terminates. π is the policy that indicates how an agent acts in a certain state. The aim of reinforcement learning is to determine the optimal policy π^* that maximizes the expected accumulated discounted reward using

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s_0 \sim \rho, a_i \sim \pi, r_i, s_{i+1} \sim E}[R_0] \tag{1}$$

$Q_\pi(s_t, a_t)$, as shown in (2), is the state–action value function that describes the expected return value conditioned on taking an action, a_t, at the initial state, s_t, and thus taking actions according to the policy, π.

$$Q^\pi(s_t, a_t) = \mathbb{E}_{a_i \sim \pi, r_i, s_{i+1} \sim E}[R_t | s_t, a_t] \tag{2}$$

2.2. Deep deterministic policy gradients

Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015) is a model-free reinforcement learning approach for the continuous control which combines the actor and critic frames as shown in Fig. 2, and thus inherits the policy gradient from the Deterministic Policy Gradient (DPG) (Silver et al., 2014) and the value function gradient from the Deep Q Network (DQN) (Mnih et al., 2013). The actor approximates the target deterministic policy, $\pi : S \to A$, which maps states to a specific action with parameters θ^π while the critic approximates the state–action value function, $Q : S \times A \to \mathbb{R}$, which presents the value of the state–action pair parameterized by θ^Q. The critic is meant to drive the value function to the optimal state–action value function Q^*. The actor is trained to obtain the policy by maximizing the expected return, J, from the start of the distribution,

$$J = \mathbb{E}_{s_0 \sim \rho, a_i \sim \pi, r_i, s_{i+1} \sim E}[R_0] \tag{3}$$

A replay buffer and a separate target network are used to avoid the unstable training (Mnih et al., 2013; Foerster et al., 2017), making the learning approach off-policy. In practice, the actor is optimized using the mini-batch gradient descent on the loss by rewriting (3) as

$$L(\theta^\pi) = -\mathbb{E}_s Q(s, \pi(s|\theta^\pi)) \tag{4}$$

where s are the samples from the replay buffer.

The critic is trained in a similar way as the Q-function in DQN to minimize the approximation loss,

$$L(\theta^Q) = E_{a_i \sim \pi, r_i, s_{i+1} \sim E}[(Q(s_t, a_t|\theta^Q) - y_t)^2] \tag{5}$$

where the target, $y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}|\theta^\pi)|\theta^Q)$, is computed using actions outputted by the actor.

2.3. Hindsight experience replay

Similar to the human ability to learn from unwanted outcomes along with desired ones, hindsight replay experience (Andrychowicz et al., 2017) becomes a goal conditioned policy learning. It extends the concept of the universal value function approximator (Schaul et al., 2015) to include goals, $g \in G$, into the MDPs. The training policy and the value functions are modified to take g as additional inputs. Thus, the policy and value function become goal conditioned, $\pi : S \times G \to A$ and $Q : S \times A \times G \to \mathbb{R}$. Changes are made to the replay buffer such that a subset of other goals, achieved goal g_a in the episode, will be stored in the transitions along with the original goal, $(s_t|g, a_t, r_t, s_{t+1}|g, g_a)$, which makes the transitions more informative. During the training, a subset of samples are modified in which the original g is replaced by the achieved goal, $g_a \in G$, to improve the positive sampling and encourage the learning process. It should also be noted that the reward in each sample needs to be recalculated based on the new goal, e.g. $r_t(s_{t+1}, g) = -(\|f_g(s_{t+1}) - g\| > \epsilon)$, where f_g presents the achieved goal at a state, s, and ϵ is a precision threshold.

2.4. Teacher–student framework with a budget

The teacher–student framework for discrete control reinforcement learning under a limited amount of advice is introduced with a set of heuristic strategies to decide when to give advice such as early advising, importance of advising, mistake advising, and predictive advising (Torrey and Taylor, 2013). The advice used in this work is the action recommended from the teacher agent. The teacher agent is first well-trained before the teaching process in the environment to gain a decent policy. It is then used to guide one student agent to behave in the same environment, which has the same state and action space as in the teacher training environment. In this teacher initiated teaching (Torrey and Taylor, 2013), state importance is used to decide the importance of giving advice at the state, $s \in S$, by computing the difference between the maximum and minimum Q-function of a well-trained teacher as in,

$$I(s) = \max_a Q_{tr}(s, a|\theta^{tr}) - \min_a Q_{tr}(s, a|\theta^{tr}) \tag{6}$$

The condition of the state importance value, $I(s) \geq t$, is then used along with other criteria to determine whether advice is given or not, where t represents a fixed value threshold. Mistake correcting methods take one more condition: whether the student-announced action is the same as the output of the teacher policy, $\pi_{st}(s|\theta_{st}) \neq \pi_{tr}(s|\theta_{tr})$. Predictive advising uses the action from another prediction model of the student instead of using the output from the student policy, $\tilde{\pi}_{st}(s|\theta_{st}) \neq \pi_{tr}(s|\theta_{tr})$. It is worth noting that the state-importance based teacher–student framework is limited to applications with finite discrete control due to the computational requirements of the state importance at each advising process in (6). As the discrete action space increases, the computation of deciding when to teach becomes more expensive and infeasible, thus cannot be applied directly in continuous action.

3. Algorithm

In this section, we introduce the teacher–student framework which is adapted to an environment with a continuous action space. The algorithm is also further modified for scaling the number of agents in the tasks, in which the teacher policy is trained for a single agent while the student policy is designed for multiple agents in performing same type of a task. Similar to the idea as in inter-task mapping (Taylor et al., 2007), the proposed algorithm is based on the assumption that the state feature, $\hat{s}_i$, of the ith agent can be reconstructed from the state in the target environment, s, to be in the form of the state in the source environment,

$$\hat{s}_i = f_i^s(s) : \hat{s}_i \in S_{tr}, s \in S_{st} \tag{7}$$

where f_i^s is a state feature mapping function from the target environment to the source environment for each agent, i. The action in the target environment can be constructed from the action in the source agent policy via the mapping function,

$$a = f^a(a_1, a_2, \dots, a_n) : a \in A_{st}, a_i \in A_{tr} \tag{8}$$

and vice versa,

$$a_i = h_i^a(a) : a \in A_{st}, a_i \in A_{tr} \tag{9}$$

where $h_i^a(a)$ is the inverse mapping function for agent i.

3.1. Motivating example

The focus of this work is to perform dual-arm pick and place tasks as well as push tasks without internal collisions. These are the initial steps toward safe casualty pose manipulation and extraction in a semi-autonomous manner based on high level inputs from a remote operator and real-time human pose estimation results (Ren et al., 2018). Following the development of push, as well as pick and place tasks, a learning architecture to enable dual-arm object manipulation in the presence of soft and hard constraints will be developed. But this will be performed as part of future work as mentioned in Section 5. Even though the proposed research aims to enable safe casualty interaction as the overall goal, the proposed approach has a wide range of applications, including situations that need fast training of a target network to control multiple agents based on the knowledge of a single trained agent. For instance, robot arms are widely used for manufacturing, assembling, and packaging in industry. They can also be used as an assistive tool for the disabled in their activities of daily living (ADL). Enabling an autonomous robotic system to perform these high level complicated tasks require the development of fundamental functions including picking and placing, as well as pushing objects to a target position.

Techniques to control a single robotic arm using explicit kinematic and dynamic models (glass box, also referred to as white box) have been developed previously to fulfill these requirements. Artificial intelligence (black box) (Andrychowicz et al., 2017) can also be used to enable autonomous execution of the above tasks. However, as the task becomes more complicated, the need to control multiple robotic arms cooperatively arises. In traditional control, a hierarchical system could be built with a global controller/planner for overall path planning and making each arm be aware of other arms, while local controllers enforce local path following. The design of a local controller using a glass box approach can take advantage from the experience of designing a controller for a single arm. In contrast, designing a hierarchical system using a black box approach may introduce redundant layers and the computational need for pre-processing, which can include feature extraction and encoding as well as design and training of a global planner. The increased computation may present a heavy burden to the control system in real-time applications and thus decrease the performance of the deployed algorithm. However, training from scratch faces sampling inefficiency as the state and action spaces increase dramatically. This situation leads to the need for training of a scaling multi-agents problem using the prior knowledge of a well-trained single agent.

3.2. Teacher–student framework for continuous control

In this section, we will discuss the proposed procedures, including early advising, mistake correcting with Q-value and mistake correcting with an action filter. The proposed procedures, mistake correcting with Q-value and mistake correcting with an action filter, require student feedback in action selection for the next step.

Early advising

A student agent benefits from a guided action data set generated by an expert policy, which is similar to using the demonstration data directly in the same target and source environment (Hester et al., 2017). In this way, a student agent could gain a basic understanding of tasks and the environment, thus obtaining an early start to the learning process. Since the scaling problem addresses different target and source environments with the assumption that the state in the target environment can be mapped to the source state, the proposed procedure is designed as follows:

Procedure 1 EARLY ADVISING $(\pi_{tr}, \{f_i^{sg}\}, f^a, n_b)$

```
for each target environment state, s do
    if ($n_b > 0$) then
        $n_b \leftarrow n_b - 1$
        for each agent, i do
            Compute state that feeds into teacher policy $\hat{s}_i = f_i^{sg}(s)$
            Compute teacher policy output action $a_i = \pi_{tr}(\hat{s}_i)$
        end for
        $a = f^a(a_1, a_2, ..a_n)$
    end if
end for
```

where f_i^{sg} is a goal conditioned state mapping function in (7), n_b is the advice budgets.

Mistake correcting with a Q-value filter

Importance Advising and Mistake Correcting are proposed separately for tasks with a discrete action space (Torrey and Taylor, 2013). In Importance Advising, a teacher agent provides advice whenever the state importance value exceed the threshold, while Mistake Correcting only considers giving advice when the action proposed by the student agent is different from the teachers. Importance Advising saves time by not acquiring the proposed action from a student agent. This is different from processing in the discrete action space. In discrete action space, the maximum and the minimum value of the Q-function can be easily found. On the other hand, estimating the state importance value in a continuous space is expensive and infeasible. Instead, the proposed procedure utilizes the teacher policy to estimate the proposed action from the student agent and compared it with the optimal action calculated from the teacher policy. If the difference reaches some threshold, the teacher proposed action will be applied to the agent. This approach was named as Mistake Correcting since it involves both the calculation from teacher and student neural network while the Importance Advising only involves the calculation from the teacher neural network. To distinguish whether the heuristic is based on the q value or the action, Mistake Correcting Q-value and Mistake Correcting Action were named separately. In both heuristics, the modified state–action importance was denoted as,

$$I_i(s) = Q_{tr}(\hat{s}_i, \pi_{tr}(\hat{s}_i)|\theta^{tr}) - Q_{tr}(\hat{s}_i, h_i^a(\pi_{st}(s))|\theta^{tr}) \tag{10}$$

The proposed procedure pertaining to the Mistake Correcting with a Q-value Filter is described as follows,

Procedure 2 MISTAKE CORRECTING Q-VALUE $(\pi_{tr}, \{f_i^{sg}\}, f^a, \{h_i^a\}, n_b)$

```
for each target environment state, s do
    if ($n_b > 0$) then
        Advise = False
        for each agent, i do
            Compute proposed actions from teacher and student policy
            $a_i^{tr} = \pi_{tr}(f_i^{sg}(s)), a_i^{st} = h_i^a(\pi_{st}(s))$
            $a_i = a_i^{st}$
            Compute state–action importance, $I_i(s) = Q_{tr}(\hat{s}_i, \pi_{tr}(\hat{s}_i)|\theta^{tr}) - Q_{tr}(\hat{s}_i, h_i^a(\pi_{st}(s))|\theta^{tr})$
            if $I_i(s) > t_I$ then
                $a_i = a_i^{tr}$, Advise = True
            end if
        end for
        $a = f^a(a_1, a_2, ..a_n)$
        if Advise then
            $n_b \leftarrow n_b - 1$
        end if
    end if
end for
```

where t_I is the threshold for the Q-value Filter to indicate whether a proposed action will lead the agent to a valued state.

Mistake correcting with an action filter

One difference between the discrete control space and the continuous control environment is that the actions selected for the discrete control problem are always distinguishable to each other. This makes the original Mistake Correcting approach judge whether the proposed actions from the teacher and the student policy are different. However, continuous control does not judge a difference under a certain range, which leads to the idea of Mistake Correcting with an Action Filter. In this approach, the teacher agent only provides advice when the proposed actions from the student agent are different enough from the teacher's policy in a certain range. This can be understood as following a proposed trajectory with a certain precision. A distance function is designed to measure the distance an action is from a proposed action. In our work, the Euclidean distance was used where $D(a_1, a_2) = \|a_1 - a_2\|_2$.

The proposed procedure pertaining to the Mistake Correcting with an Action Filter is described as follows,

Procedure 3 MISTAKE CORRECTING ACTION $(\pi_{tr}, \{f_i^{sg}\}, f^a, \{h_i^a\}, n_b)$

```
for each target environment state, s do
    if ($n_b > 0$) then
        Advise = False
        for each agent, i do
            Compute proposed actions from teacher and student policies
            $a_i^{tr} = \pi_{tr}(f_i^{sg}(s)), a_i^{st} = h_i^a(\pi_{st}(s))$
            if $D(a_i^{tr}, a_i^{st}) = \|a_i^{tr} - a_i^{st}\|_2 > t_a$ then
                $a_i = a_i^{tr}$, Advise = True
            end if
        end for
        $a = f^a(a_1, a_2, ..a_n)$
        if Advise then
            $n_b \leftarrow n_b - 1$
        end if
    end if
end for
```

where t_a is the threshold for the Action Filter to specify the maximum distance that one action can be from the action proposed by the teacher agent. According to whether t_a relies on the value of the teacher action output, these methods are divided to relative (MC_ACT) and absolute versions (MC_ACT_ABS).

$$t_a = \begin{cases} \alpha, & \text{MC\_ACT.} \\ \alpha * \pi_{tr}(f_i^{sg}(s)), & \text{MC\_ACT\_ABS.} \end{cases} \tag{11}$$

where α is a positive constant chosen as the absolute threshold for the action filter.

4. Simulations and experiments

This section is organized as follows. Section 4.1 describes the RL environments for training the teacher policy and the student policy with guidance from the teacher policy. Section 4.2 presents the simulation results from different teaching strategies along with the impact of different thresholds on the learning process and analysis. Section 4.3 describes the physical experiments with the dual-arm manipulation system using the trained policy.

4.1. Environments

To train the agent policy and to test the proposed methods, the simulation environment was setup using the 6 Degree of Freedom (DOF) Kinova Jaco arm, which has a three-finger gripper. The gripper has multiple DOFs but is activated through a single input corresponding with the opening and closing of all the fingers together. The robot is simulated using the MuJoCo physics engine (Todorov et al., 2012). Fig. 3 (a,b) presents the training environment for the teacher policy. The neural network is trained to control only one Jaco arm for the push as well as pick and place tasks. Fig. 3 (c,d) shows the training environment for student policy, where a different neural network is trained to control two Jaco arms to complete the scaled tasks. In the push task, the arm needs to push the objects to a desired location on the table while the gripper of the arm is not used. In the pick and place task, the aim is to pick up the objects placed on the table and then place them at the target positions. The target positions can be either in air or on the table and the three fingers are actuated symmetrically by one single input state. Soft contact between the objects and the fingers are simulated in MuJoCo to oppose the slip in the tangential plane and rotation around the contact normal direction. In both tasks, the initial position of the gripper and the object, and the target position of the object are randomly sampled within the workspace of the Jaco arm. Inspired by Plappert et al. (2018), the gripper orientation is fixed toward the desk at all times. A Timestep of 0.002 s is set in the simulator to perform fast and accurate simulation of the dynamic model and soft contact.

Observations

The states of the system are obtained from the MuJoCo engine and consist of the robots gripper states and object states including the orientation, position, and velocity. The teacher training environment consists of the states of one robot and one object while the student training environment consists of a pair of robots and objects with states in the same order as in the teacher training environment.

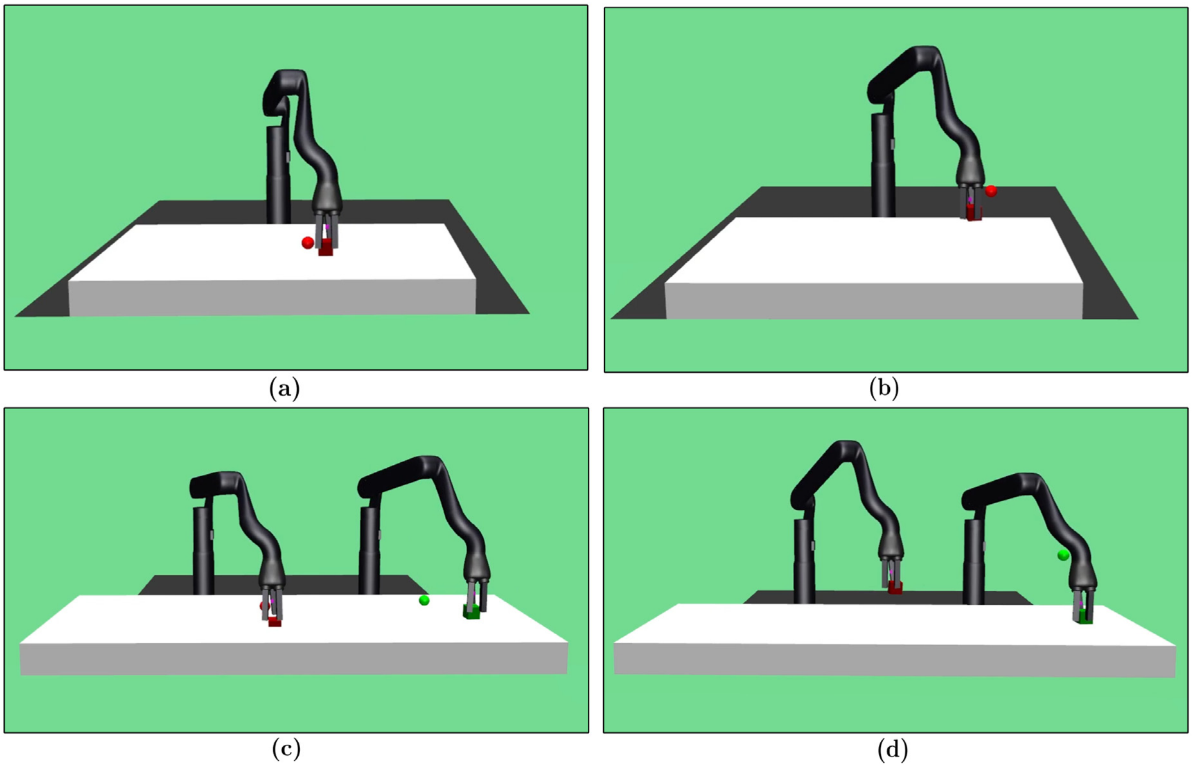


Fig. 3. Training environment in simulation: (a) pushing an object to a target position using a single arm, (b) picking and placing an object at a desired location using a single arm, (c) pushing objects to target places using the dual arm system, and (d) picking and placing objects at desired locations using the dual arm system.

Table 1
Environment and neural network summary.

Model Structure	Source environment 3 × [256] MLP		Target environment 4 × [256] MLP			
Task	Push	Pick and Place	Push		Pick and Place	
Agent	–	–	*Student*	*Teacher*	*Student*	*Teacher*
States	$\mathbb{R}^{28}$	$\mathbb{R}^{28}$	$\mathbb{R}^{56}$	$\mathbb{R}^{28}$	$\mathbb{R}^{56}$	$\mathbb{R}^{28}$
Goal	$\mathbb{R}^{6}$	$\mathbb{R}^{6}$	$\mathbb{R}^{6}$	$\mathbb{R}^{3}$	$\mathbb{R}^{6}$	$\mathbb{R}^{3}$
Actions	$\mathbb{R}^{3}$	$\mathbb{R}^{4}$	$\mathbb{R}^{6}$	$\mathbb{R}^{3}$	$\mathbb{R}^{8}$	$\mathbb{R}^{4}$

Actions

Instead of controlling the joint angles of the robot directly, position of the end effectors are used as the control input for the robot. This allows for transferability among different robotic arms. In real world applications, the desired joint angles can be calculated using an inverse kinematic model. The orientation of the end effector is fixed toward the ground during the task. The actions consist of position of the end effector along with one state for controlling the gripper to open/close, $A_{tr} = \{a_i : a_i \in \mathbb{R}^4\}$ and $A_{st} = \{a_i : a_i \in \mathbb{R}^8\}$.

Goals

The goals are defined as the set of target positions of objects: $G_{tr} = \{g_i : g_i \in \mathbb{R}^3\}$ is for the teacher environment and $G_{st} = \{g_i : g_i \in \mathbb{R}^6\}$ is for the student environment. In this work, the observation, goal, and reward in the student environment are constructed by appending the corresponding data from the individual agents, $d_{st} = d_1 \parallel d_2$, where d_i represents the data of agent i.

Rewards

In both teacher and student training environments, sparse rewards are used as, $r_t(s_{t+1}, g) = -(\|f_g(s_{t+1}) - g\| > \epsilon)$, where f_g maps the state, s, to an achieved goal, g, and ϵ determines the control precision in the task. The agent only receives a reward of 1 when getting the object within a threshold of the target position; otherwise, a reward of 0 is received. The deterministic policies are represented as Multi-Layer Perceptrons (MLPs) with Rectified Linear Unit (ReLU) activation functions. Three layers of perceptrons are used in training the teacher policy while four layers are used in training the student policy, due to larger number of input features and the increased complexity of the problem. The discount factor of the cumulative reward is set to 0.98 in all training and testing environments. The environments and neural network model summary of both the training and teaching scenarios are shown in Table 1.

Advice

In the student training environments, the advice given by the teacher agent refers to the recommended action output by the teacher policy. In all the following training algorithms, the advice budget is set to 500,000.

4.2. Simulation results

Sixteen workers, each having two rollouts, were used to improve the training efficiency and the parameters are updated after every episode. Each worker is fixed to a CPU core and the rollouts start with different initial and target conditions, but with the same policy. To achieve a stable optimization, a target network and a main network were deployed with the same architecture. The target network and the main network are in the same architecture, while the target network is updated at a slower pace than the main network using a Polyak-averaged version of the main network (Polyak and Juditsky, 1992). The target network parameters are stored globally and downloaded to the workers before running each episode. In this work, each epoch consists of generating 50 episodes and training the neural network with 40 batches of 256 transitions.

To compare the performance of the teaching framework for the sparse rewards and the continuous action space environment to obtain the teacher policy, we trained the deterministic policy from scratch without advice in the push, and pick and place environments using

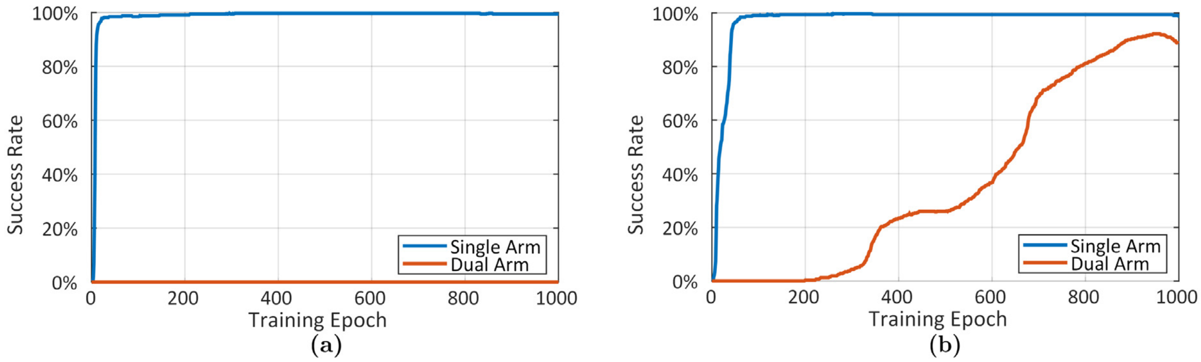


Fig. 4. The training process from scratch: (a) training process for pushing object and (b) training process for pick and place object.

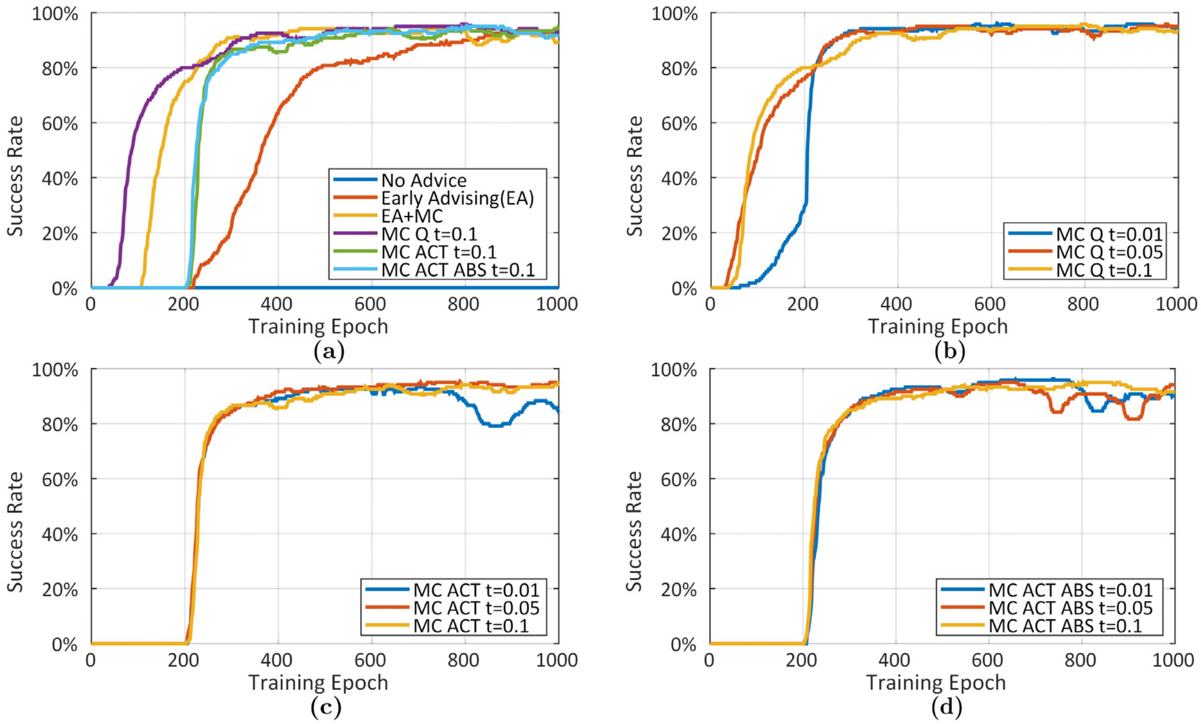


Fig. 5. Simulation results showing the performance of different advising strategies and thresholds in the push task: (a) comparison of the performance of different strategies, (b) comparison of the performance of MC-Q with different thresholds, (c) comparison of the performance of MC-ACT with different thresholds, (d) comparison of the performance of MC-ACTABS with different thresholds.

the single arm and dual-arm systems as shown in Fig. 3. The actor and the critic used in the single arm scenario consists of three layers with 256 neurons per layer. For the dual-arm scenario there are four layers with the same number of neurons. Both training processes used the HER buffer (Andrychowicz et al., 2017). Fig. 4 shows the success rate of the training procedures. The single arm manipulation achieved 90% success rate after 12 and 43 epochs in push, and pick and place tasks, respectively. The dual arm manipulation achieved 90% success rate after 898 epochs in the pick and place task, but failed in the push task. This reflects the fact that the task complexity grows rapidly as the dimensionality of the control space increases from the single arm to the dual-arm case as well as the interaction between agents increases. Compared to the pick and place task using dual-arm, the push task using dual-arm and locked grippers made it more likely to interact with each other in 2D than in 3D.

Figs. 5(a) and 7(a) show the performance of different strategies in the push, and the pick and place environments, including No Advice (NA), Early Advising (EA), Early Advising and Mistake Correcting (EAMC), Mistake Correcting with Q-value Filter with 0.1 threshold (MC-Q01), Mistake Correcting with Action Filter with 0.1 threshold (MC-ACT01) and Mistake Correcting with Action Filter with 0.1 absolute threshold (MC-ACTABS01). The EAMC procedure applies EA for the first half of the advice budgets and then applies MC for the other half of the advice budgets. It can be seen from the figure that NA converges much slower than all the others in both cases. The *inspired epoch* is defined as the epoch at which the strategy achieves 10% of the maximum success rate it achieved in its training process. Among all the strategies, the inspired epoch of the MC-Q01 is the lowest, 63 in push task and 86 in pick and place task. All the other strategies gained a lower inspired epoch as compared to NA. MC-ACT001 obtained the shortest rise time, which is defined as the number of epochs it takes to increase from 10% to 90% of the maximum success rate it achieved. MC-Q01 outperforms other strategies in the push task and achieves a maximum success rate of 0.9583. EA is the best strategy in the pick and place task with a maximum success rate of 0.9750. Figs. 6 and 8 show the mean Q-value in push and pick and place tasks. Similar to the success rate, the Q-value converges faster when using advising framework.

To estimate the impact of the threshold of Mistake Correcting on learning performance, three different threshold values were picked,

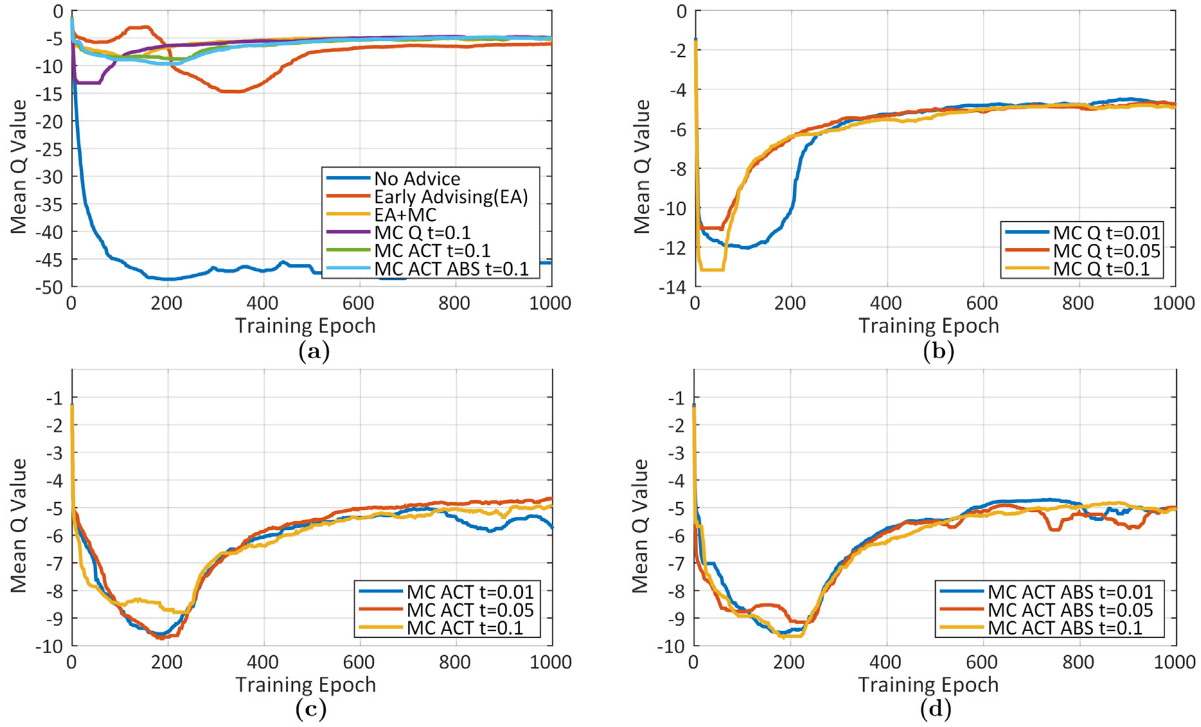


Fig. 6. Simulation results showing the mean Q-value of different advising strategies and thresholds in the push task: (a) comparison of the mean Q-value on different strategies, (b) comparison of the mean Q-value of MC-Q with different thresholds, (c) comparison of the mean Q-value of MC-ACT with different thresholds, (d) comparison of the mean Q-value of MC-ACTABS with different thresholds.

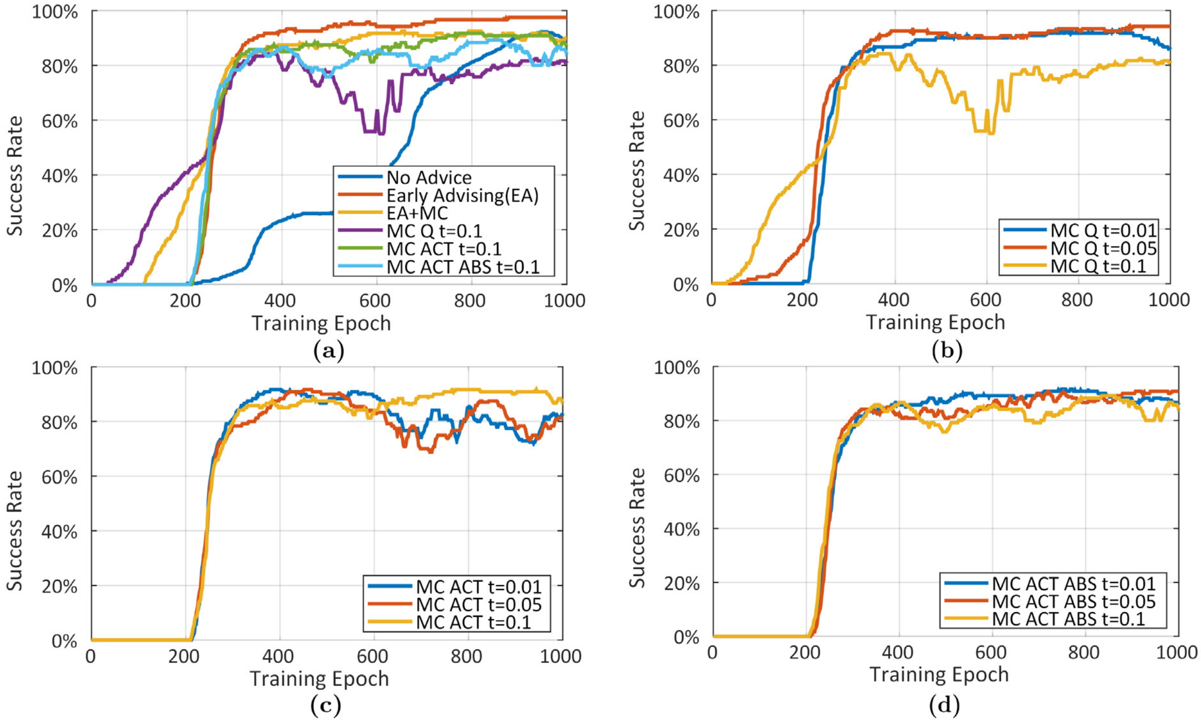


Fig. 7. Simulation results showing the performance of different advising strategies and thresholds in the pick and place task: (a) comparison of the performance on different strategies, (b) comparison of the performance on MC-Q with different thresholds, (c) comparison of the performance on MC-ACT with different thresholds, (d) comparison of the performance on MC-ACTABS with different thresholds.

$t = \{0.01, 0.05, 0.1\}$ and the convergence speed was analyzed. As the threshold changes from 0.01 to 0.1, the inspired epoch of Mistake Correcting with Q-value Filter decreases significantly from 140 to 63 in push tasks and from 220 to 86 in pick and place tasks, while it does not have a great impact on the Mistake Correcting with Action Filter methods. This may be due to the fact that the range of action is constrained by the motor while the Q value can be ranged from negative infinity to positive infinity such that Mistake Correcting with Q-value is more sensitive to the threshold value. This shows that the more loosely the teacher advises the students in the new environment,

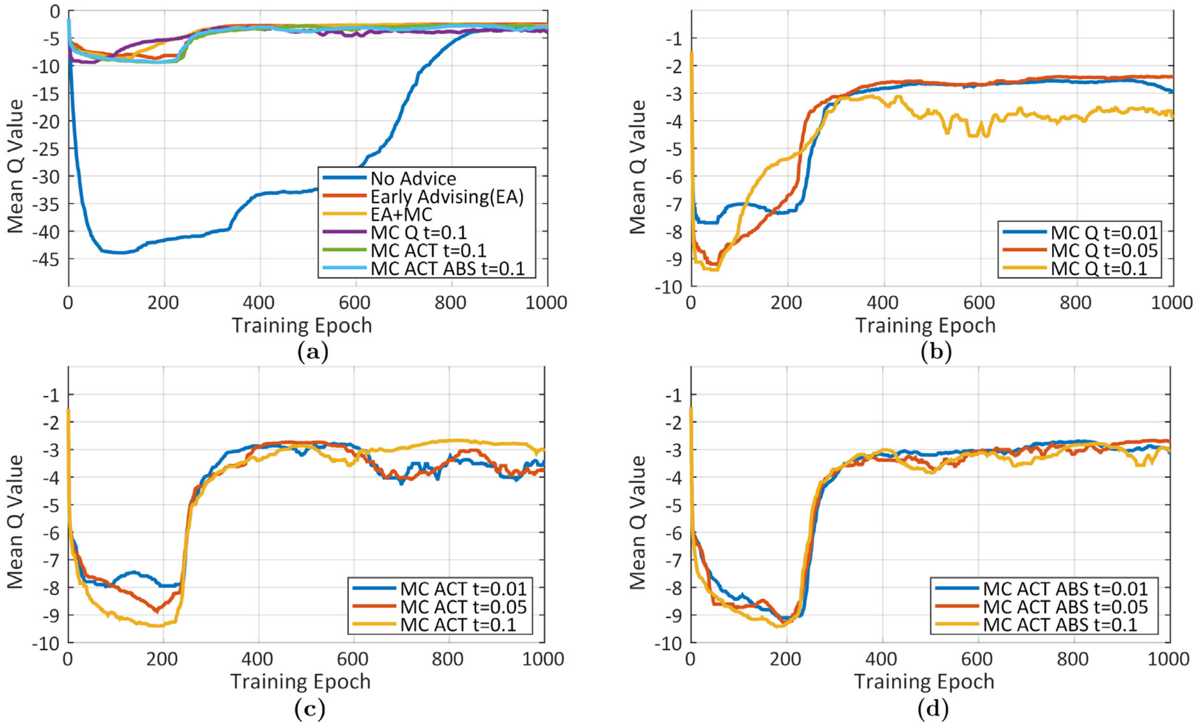


Fig. 8. Simulation results showing the mean Q-value of different advising strategies and thresholds in the pick and place task: (a) comparison of the mean Q-value on different strategies, (b) comparison of the mean Q-value on MC-Q with different thresholds, (c) comparison of the mean Q-value on MC-ACT with different thresholds, (d) comparison of the mean Q-value on MC-ACTABS with different thresholds.

the more freely the student explores the new environment and thereby obtains knowledge about the new environment as compared to agents with a more strict teacher. This observation could be attributed to the fact that the student with a loose teacher gains more opportunities to try new actions as compared to the students with a more strict teacher. This allows the student to gain positive samples via its own policy and knowledge of the new environment, after obtaining a certain amount of advice. Another concern in transfer learning is the time that one spends to achieve a decent success rate. *Effective learning rate* is defined as 90% of the maximum success rate divided by the number of epochs it takes to reach that success rate. In push task, MC-Q005 achieved a higher effective learning rate as compared to other strategies, while the effective learning rate of all advising strategies in pick and place task are almost equal and they outperform NA. This may be due to the intrinsic complexity of the push task where the trajectories are constrained on the table surface and attention is provided to avoid collision, resulting in strategies with diverse performance. The details of the impact factor on the performance of different advising strategies is summarized in Table 2.

It is important to note that the EA slightly outperforms all the other strategies with large advising threshold in the pick and place task. The strategies with a large threshold result in more fluctuation in the success rate and lowers it in general for the pick and place task. However, in the push task this phenomenon is not apparent.

4.3. Experimental results

This section presents the real world experimental validation of the trained policy obtained from the simulation on two Kinova Mico arms with Quanser SDK. The algorithm used to generate the workspace trajectories for the arms to complete the tasks is agnostic to the robot. This claim is experimentally validated by transferring the knowledge obtained from a 6-DOF arm, in simulation to a 4-DOF arm used for the experiments.

Two Quanser Mico arms (Kinova, 2018) used for the push, and the pick and place experiments are shown in Fig. 9. The Mico arm is a 4-DOF robotic arm with a 2-finger gripper, retrofitted by Quanser with a real-time control interface capable of running in MATLAB and Simulink (MathWorks). The Simulink blocks, provided by the Quanser SDK, allows for precise joint and torque control. The 2-finger gripper is programmed to be controlled by one input during the pick and place experiment while it is kept locked in the push tasks. The specifications of the robotic arms can be found in Table 3.

Table 2
Performance summary of different advising strategies.

	Push task				Pick and place task			
	IE[a]	RT[b]	MSR[c]	ELR[d]	IE	RT	MSR	ELR
NA	NaN	NaN	0	0	336	489	0.9219	0.0010
EA	252	363	0.9333	0.0014	228	96	**0.9750**	0.0027
EAMC	116	139	0.9417	0.0033	135	178	0.9250	0.0027
MC-Q001	140	105	0.9583	0.0035	220	93	0.9250	0.0027
MC-Q005	**46**	195	0.9583	**0.0036**	174	156	0.9417	0.0026
MC-Q01	63	229	0.9583	0.0030	**86**	209	0.8417	0.0026
MC-ACT001	214	**71**	0.9333	0.0029	225	**82**	0.9167	**0.0027**[e]
MC-ACT005	215	96	0.9500	0.0027	224	134	0.9167	0.0023
MC-ACT01	216	80	0.9500	0.0029	226	85	0.9167	0.0027
MC-ACTABS001	217	101	**0.9625**	0.0027	225	104	0.9167	0.0025
MC-ACTABS005	214	91	0.9500	0.0028	229	85	0.9250	0.0027
MC-ACTABS01	213	104	0.9500	0.0027	222	108	0.9083	0.0025

[a] IE: Inspired Epoch.
[b] RT: Rise Time.
[c] MSR: Maximum Success Rate.
[d] ELR: Effective Learning Rate.
[e] Compared in double data type

The desired trajectories, the position of the end effector and the open/close commands for the gripper, are generated by the well-trained dual-arm student policy. The Quarc simulator from Quanser was used to check the feasibility of the proposed trajectories before they were executed by the real robots. Object positions were kept to be the same for both the tasks in the simulation and real world experiments. But

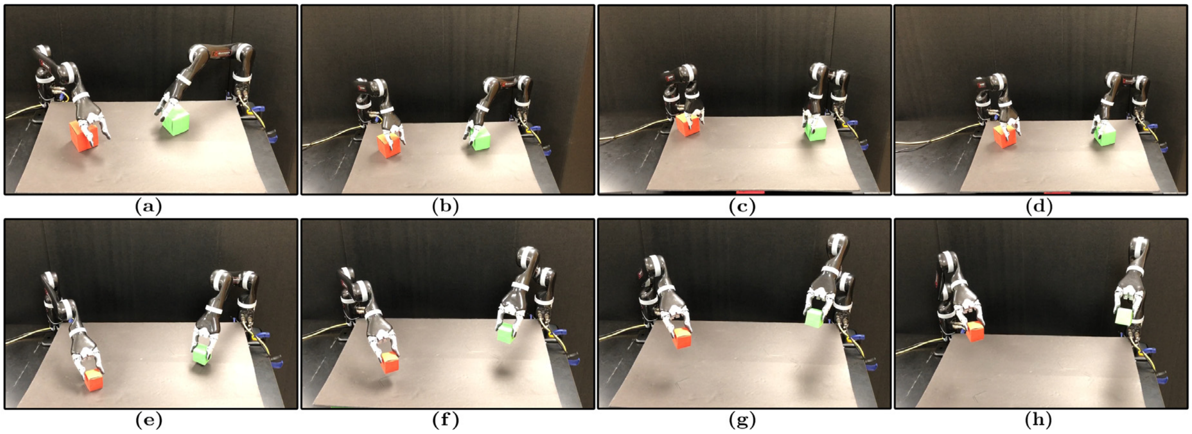


Fig. 9. Real world experiment using dual arm: (a–d) pushing objects to the target positions using dual arms, and (e–f) picking and placing objects at the desired locations using dual arms.

Table 3
Details of robot specification.

	Mico robot arm	Jaco robot arm
Arm DOF	4	6
Gripper DOF	2	3
Max reach	55 cm	70 cm
Max linear arm speed	20 cm/s	20 cm/s
Joint range	±27.7 rev	±27.7 rev
Max payload	0.75 kg full extension 1.25 kg mid-range	0.8 kg full extension 1.3 kg mid-range
Control interface	Matlab	ROS and Kinova SDK
Joint sensor	Encoder (index, absolute), torque sensor, motor current, motor temperature	
Finger sensor	Encoder (relative)	

Table 4
Performance summary of real-world experiments.

	Push task	Pick and place task
Baseline method[a]	0/50	30/50
Proposed method[b]	40/50	38/50

[a] Baseline method refers to the method of training from scratch without the advising framework.
[b] Proposed methods refers to the methods of training with the proposed advising framework.

the size of the objects for the push task were modified from 5.6 cm to 9 cm in real world experiments for compatibility with the grippers.

To evaluate the improvements brought about by the proposed methods, we used the from scratch method of training as the baseline method for this work. We applied the best well-trained policies generated from the proposed methods and the baseline method to the robotic manipulators to perform the tasks, respectively. For push, as well as pick and place tasks, 50 trials of the real-world experiments were performed with different initial and target conditions generated randomly using either policy. In the push task, 40 out of 50 trials ended with success using the proposed methods while all trials failed using the baseline method. In the pick and place task, 38 out of 50 trials resulted in success using the proposed methods while 30 out of 50 trials resulted in success using the baseline method. The experimental results are presented in Table 4.

For the 10 failed trials of the push task using the proposed methods, the robot arm failed to move the object in the right direction as the object rolled away from the gripper during the push process. This could be due to the difference in the grippers used in the experiment as compared to the training environment. Inaccuracies in modeling physical properties of objects inside the simulated environment could also result in poor performance of the trained policy. All trials using the baseline method ended in either case of exceeding the testing time or breaking the safety rules such as collision with each other or with the table.

In the 12 failures in the pick and place tasks using the proposed methods, the gripper failed to grasp the object while moving in 3D. This could also be due to the difference in physical properties such as deformation and friction of the objects in real life as compared to the simulation. Even though the real world experiments sometimes failed due to inaccuracies in modeling object–gripper contact and interaction, no collisions of the robot arms were seen in the real-world experiments. This indicates that the teacher–student framework assists the student in learning the additional requirements that come with the multi-agent problem (in this case collision avoidance). Among the 20 failures in the pick and place tasks using the baseline method, 8 failures were caused by reaching the target position inaccurately, while the other failures were caused by failing to grasp the object firmly during the movement.

5. Conclusion and future work

Solving problems using multiple agents is an effective approach toward tackling complicated tasks that typically involve high dimensional workspace. Unlike the classical method of stacking the individual controller with explicit mathematical model to create a hierarchical controller, stacking multiple well-trained single-agent neural networks is prone to make the overall system redundant and expensive for real-time computation. Training a new super agent for multiple agent control from scratch is computationally intensive and requires careful design of hyperparameters to guarantee convergence.

In order to handle the scaling problem in training multiple agents to perform a specified task, this paper extends prior research on advising framework by leveraging the knowledge of a well-trained-single agent. Furthermore, this work adapts the previous advising framework to allow for continuous control and presents a set of advising strategies that accelerate the learning process as compared to training from scratch. Furthermore, the proposed training framework avoids bias and convergence to a sub-optimal solution by using demonstration data. The performance resulting from different strategies are analyzed using different hyperparameters and demonstrated using physical experiments with a dual-arm robotic system. The experimental validations demonstrated the benefits in using different advising strategies. “Mistake Correcting with Q-value Filter” outperformed all the others in the push task, “Early Advising” gained the highest success rate in the pick and place task, “Mistake Correcting with Action Filter with 0.01 threshold” obtained the shortest rise time in both tasks. This shows that different strategies have different benefits and they need to be chosen based on the requirements of the task at hand. This provides directions for future applications of the above techniques. Even though the threshold may need to be tuned for different tasks to obtain the best performance, all the strategies showed the benefit of a faster convergence rate under the proposed approach.

While the strategies proposed in this work were implemented in a deterministic environment, they could also be applied in a stochastic case. However, a more robust solution could be developed for the purely stochastic case. For example, deciding whether or not to give an advice could be evaluated based on the similarity between the policy distributions of the teacher and the student agents.

Having addressed the scaling problem effectively, the next step in multi-agent learning is to enable cooperative behavior at a higher level by implementing behavioral constraints that either persist or are conditional. An example case would be to use the dual-arm system to collaboratively move a single object. Another case would be to enable the arms to safely operate in an environment with moving objects. Developing these capabilities as part of future work will be the next step in realizing safe casualty extraction using the SAVER system.

CRediT authorship contribution statement

Hailin Ren: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing - original draft, Writing - review & editing, Visualization, Supervision. **Pinhas Ben-Tzvi:** Conceptualization, Methodology, Validation, Investigation, Resources, Writing - original draft, Writing - review & editing, Visualization, Supervision, Project administration, Funding acquisition.

Acknowledgments

The authors would like to thank Bijo Sebastian, Raghuraj Chauhan, Vinay Kamidi and Jiteng Yang for their help in this work. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. This work was supported in part by the US Army Medical Research & Material Commands Telemedicine & Advanced Technology Research Center (TATRC), under Contract No. W81XWH-16-C-0062. The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official Department of the Army position, policy, or decision unless so designated by other documentation.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.engappai.2020.103515.

References

Amir, O., Kamar, E., Kolobov, A., Grosz, B.J., 2016. Interactive teaching strategies for agent training. In: IJCAI International Joint Conference on Artificial Intelligence.

OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., Zaremba, W., 2018. Learning Dexterous in-hand manipulation. pp. 1–27, arXiv preprint abs/1808.0. doi: arXiv:1808.00177v2. URL http://arxiv.org/abs/1808.00177.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., Zaremba, W., 2017. Hindsight experience replay. In: Conference on Neural Information Processing Systems. URL http://arxiv.org/abs/1707.01495.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W., 2016. OpenAI gym. arXiv preprint. URL http://arxiv.org/abs/1606.01540.

Fachantidis, A., Taylor, M., Vlahavas, I., 2017. Learning to teach reinforcement learning agents. Mach. Learn. Knowl. Extr. 1 (1), 21–42. http://dx.doi.org/10.3390/make1010002, URL http://arxiv.org/abs/1707.09079 http://www.mdpi.com/2504-4990/1/1/2.

Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P.H.S., Kohli, P., Whiteson, S., 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In: International Conference on Machine Learning, Sydney, Australia, URL https://arxiv.org/pdf/1702.08887.pdf http://arxiv.org/abs/1702.08887.

Ghalamzan, A.M.E., Ragaglia, M., 2018. Robot learning from demonstrations: Emulation learning in environments with moving obstacles. Robot. Auton. Syst. 101, 45–56. http://dx.doi.org/10.1016/j.robot.2017.12.001.

Gupta, A., Devin, C., Liu, Y., Abbeel, P., Levine, S., 2017. Learning invariant feature spaces to transfer skills with reinforcement learning. In: International Conference on Learning Representations. URL http://arxiv.org/abs/1703.02949.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., Leibo, J.Z., Gruslys, A., 2017. Deep Q-learning from demonstrations. arXiv preprint. URL http://arxiv.org/abs/1704.03732.

Ho, J., Ermon, S., 2016. Generative adversarial imitation learning. In: 30th Conference on Neural Information Processing Systems, Barcelona, Spain, pp. 4565–4573. URL http://arxiv.org/abs/1606.03476.

Kinova, Jaco - Robotic arm. URL https://www.kinovarobotics.com/en/products/assistive-technologies.

Kinova, 2018. MICO - robotic arm. URL https://www.kinovarobotics.com/en/knowledge-hub/all-kinova-products.

Leno Da Silva, F., Glatt, R., Reali Costa, A.H., 2017. Simultaneously learning and advising in multiagent reinforcement learning. In: AAMAS '17 Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, São Paulo, Brazil, pp. 1100–1108. URL www.ifaamas.org.

Levine, S., Wagener, N., Abbeel, P., 2015. Learning contact-rich manipulation skills with guided policy search. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), Vol. 2015-June. IEEE, pp. 156–163. http://dx.doi.org/10.1109/ICRA.2015.7138994, URL http://ieeexplore.ieee.org/document/7138994/.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint. URL http://arxiv.org/abs/1509.02971.

MathWorks, Simulink - Simulation and Model-Based Design - MATLAB. URL https://www.mathworks.com/products/simulink.html.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv Preprint. URL http://arxiv.org/abs/1312.5602.

Munemasa, I., Tomomatsu, Y., Hayashi, K., Takagi, T., 2018. Deep reinforcement learning for recommender systems. In: International Conference on Information and Communications Technology 2018-Janua. pp. 226–233. http://dx.doi.org/10.1109/ICOIACT.2018.8350761, URL http://arxiv.org/abs/1807.06613.

Omidshafiei, S., Kim, D.-K., Liu, M., Tesauro, G., Riemer, M., Amato, C., Campbell, M., How, J.P., 2019. Learning to teach in cooperative multiagent reinforcement learning. In: Association for the Advancement of Artificial Intelligence. URL www.aaai.org.

Parisotto, E., Ba, J.L., Salakhutdinov, R., 2016. Actor-Mimic: Deep multitask and transfer reinforcement learning. In: International Conference on Learning Representations, San Juan, Puerto Rico, http://dx.doi.org/10.1007/s11187-005-6455-x. URL http://arxiv.org/abs/1511.06342.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., Zaremba, W., 2018. Multi-goal reinforcement learning: Challenging robotics environments and request for research. arXiv preprint. URL http://arxiv.org/abs/1802.09464.

Polyak, B.T., Juditsky, A.B., 1992. Acceleration of stochastic approximation by averaging. SIAM J. Control Optim. 30 (4), 838–855. http://dx.doi.org/10.1137/0330046, URL http://epubs.siam.org/doi/10.1137/0330046.

Pong, V., Gu, S., Dalal, M., Levine, S., 2018. Temporal difference models: Model-free deep RL for model-based control. In: International Conference on Learning Representations. URL http://arxiv.org/abs/1802.09081.

Popov, I., Heess, N., Lillicrap, T., Hafner, R., Barth-Maron, G., Vecerik, M., Lampe, T., Tassa, Y., Erez, T., Riedmiller, M., 2017. Data-efficient deep reinforcement learning for Dexterous manipulation. http://dx.doi.org/10.1051/0004-6361/201527329, arXiv preprint (section V). URL http://arxiv.org/abs/1704.03073.

Ren, H., Kumar, A., Wang, X., Ben-Tzvi, P., 2018. Parallel deep learning ensembles for human pose estimation. In: Dynamic Systems and Control Conference. ASME, Atlanta, Georgia, V001T07A005. http://dx.doi.org/10.1115/DSCC2018-9007.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Van de Wiele, T., Mnih, V., Heess, N., Springenberg, J.T., 2018. Learning by playing - solving sparse reward tasks from scratch. In: Proceedings of Machine Learning Research. pp. 4344–4353. http://dx.doi.org/10.1641/0006-3568(2000)050, URL http://proceedings.mlr.press/v80/riedmiller18a.html.

Schaul, T., Horgan, D., Gregor, K., Silver, D., 2015. Universal value function approximators. In: Proceedings of the 32nd International Conference on Machine Learning, Lille, France, pp. 1312–1320. URL http://jmlr.org/proceedings/papers/v37/schaul15.html.

Silva, F.L.D., Costa, A.H.R., 2019. A survey on transfer learning for multiagent reinforcement learning systems. J. Artificial Intelligence Res. 64, 645–703. http://dx.doi.org/10.1613/jair.1.11396, URL https://www.jair.org/index.php/jair/article/view/11396.

Silver, D., Lever, G., Technologies, D., Lever, G.U.Y., Ac, U.C.L., 2014. Deterministic policy gradient algorithms. In: Proceedings of the 31 St International Conference on Machine Learning.

Taylor, M.E., Stone, P., Liu, Y., 2007. Transfer learning via inter-task mappings for temporal difference learning. J. Mach. Learn. Res. 8, 2125–2167, URL https://www.cs.utexas.edu/~pstone/Papers/bib2html-links/JMLR07-taylor.pdf.

Todorov, E., Erez, T., Tassa, Y., 2012. MuJoCo: A physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 5026–5033. http://dx.doi.org/10.1109/IROS.2012.6386109, URL http://ieeexplore.ieee.org/document/6386109/.

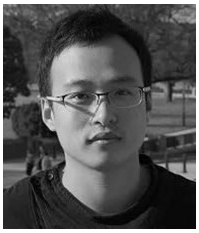
Torrey, L., Taylor, M., 2013. Teaching on a budget: Agents advising agents in reinforcement learning. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems. pp. 1053–1060, URL http://dl.acm.org/citation.cfm?id=2485086.

Vasquez, D., Okal, B., Arras, K.O., 2014. Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, no. Iros. IEEE, pp. 1341–1346. http://dx.doi.org/10.1109/IROS.2014.6942731, URL http://ieeexplore.ieee.org/document/6942731/.

Williams, A., Sebastian, B., Ben-Tzvi, P., Review and analysis of search, extraction, evacuation, and medical field treatment robots. http://dx.doi.org/10.1007/s10846-019-00991-6.

Zhan, Y., Taylor, M.E., 2015. Online transfer learning in reinforcement learning domains. arXiv preprint. URL http://arxiv.org/abs/1507.00436.

Zimmer, M., Viappiani, P., Weng, P., 2014. Teacher-student framework: A reinforcement learning approach. In: AAMAS Workshop Autonomous Robots and Multirobot Systems, Paris, France, URL https://matthieu-zimmer.net/publications/ARMS2014.pdf.



Hailin Ren (S'16) received the B.S Degree in Nanjing University of Science and Technology in 2013 and M.S. in Columbia University in 2014. He is currently pursuing the Ph.D. degree at the Virginia Polytechnic Institute and State University (Virginia Tech) under the supervision of Prof. P. Ben-Tzvi. His research interests include Artificial Intelligence, Computer Vision and Autonomous Robotics.

Pinhas Ben-Tzvi (S'02-M'08-SM'12) received the B.S. degree (summa cum laude) in mechanical engineering from the Technion-Israel Institute of Technology and the M.S. and Ph.D. degrees in mechanical engineering from the University of Toronto. He is currently an Associate Professor of Mechanical Engineering and Electrical and Computer Engineering, and the founding Director of the Robotics and Mechatronics Laboratory at Virginia Tech. His current research interests are in robotics and intelligent autonomous systems, human-robot interactions, robotic vision and visual servoing/odometry, machine learning, mechatronics design, systems dynamics and control, mechanism design and system integration, and novel sensing and actuation. He has authored and co-authored more than 155 peer-reviewed journal articles and refereed papers in conference proceedings and is the named inventor on at least twelve U.S. patents and patent applications. Dr. Ben-Tzvi is the recipient of the 2019 Virginia Tech Teaching Excellence Award, 2018 Virginia Tech Faculty Fellow Award, the 2013 GWU SEAS Outstanding Young Researcher Award and the GWU SEAS Outstanding Young Teacher Award, as well as several other honors and awards. Dr. Ben-Tzvi is Technical Editor for the IEEE/ASME Transactions on Mechatronics, Associate Editor for ASME Journal of Mechanisms and Robotics, Associate Editor for IEEE Robotics and Automation Magazine, and an Associate Editor for the Int'l Journal of Control, Automation and Systems and served as an Associate Editor for IEEE ICRA 2013, 2014, 2016, 2017, and 2018. Dr. Ben-Tzvi is a senior member of IEEE and a member of ASME