IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 31 January 2023, accepted 8 March 2023, date of publication 13 March 2023, date of current version 21 March 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3256724*

RESEARCH ARTICLE

# Boosting Performance of Visual Servoing Using Deep Reinforcement Learning From Multiple Demonstrations

**ALI AFLAKIAN, ALIREZA RASTEGARPANAH, AND RUSTAM STOLKIN, (Member, IEEE)**
Department of Metallurgy and Materials Science, University of Birmingham, B15 2TT Birmingham, U.K.
The Faraday Institution, Quad One, Harwell Science and Innovation Campus, OX11 0RA Didcot, U.K.

Corresponding author: Alireza Rastegarpanah (a.rastegarpanah@bham.ac.uk)

This work was supported in part by the project "Reuse and Recycling of Lithium-Ion Batteries (RELIB)," and in part by The Faraday Institution under Grant FIRG005.

**ABSTRACT** In this study, knowledge of multiple controllers was used and combined with deep reinforcement learning (RL) to train a visual servoing (VS) technique. Deep RL algorithms were successful in solving complicated control problems, however they generally require a large amount of data before they achieve an acceptable performance. We developed a method that generates online hyper-volume action bounds from demonstrations of multiple controllers (experts) to address the issue of insufficient data in RL. The agent then continues to explore the created bounds to find more optimized solutions and gain more rewards. By doing this, we cut out pointless agent explorations, which results in a reduction in training time as well as an improvement in performance of the trained policy. During the training process, we used domain randomization and domain adaptation to make the VS approach robust in the real world. As a result, we showed a 51% decrease in training time to achieve the desired level of performance, compared to the case when RL was used solely. The findings showed that the developed method outperformed other baseline VS methods (image-based VS, position-based VS, and hybrid-decoupled VS) in terms of VS error convergence speed and maintained higher manipulability.

**INDEX TERMS** Visual servoing, reinforcement learning, online action bounding, reinforcement learning from demonstrations, manipulability.

## I. INTRODUCTION

Robot vision is significantly important in perception as it endows the robot with the ability to perceive the surrounding environment without making direct contact with the object. One of the well-developed approaches in robot vision is visual servoing (VS) which converts visual errors into actuator commands [1]. Despite improvements, the traditional VS methods still have some limitations in terms of stability, convergence, and gain selection [2]. Some of these issues are associated with calculating the image Jacobian (Interaction matrix), including singularities and local minima. Other prominent problems are based on the fact that there is no direct control over the robot joint velocities. As a result, the controller is unaware of the limitations of the robot arm and its performance, especially when employing stationary manipulators such as industrial arms. Other essential aspects of VS are robustness to camera calibration, image noise, disturbances, errors in modeling of the problem, errors in robot kinematics, sensor data, and keeping visual features in the camera field of view (FOV).

Thanks to improved learning methods, new generations of robots can learn and prevent issues of this kind [3]. Supervised learning techniques aid in anticipating data output based on prior experiences; therefore, the dataset must be labeled ahead of time. However, labeled data are scarce for many real world applications, whereas unlabeled data are plentiful. Unsupervised learning methods and reinforcement learning (RL) algorithms could solve the problem of limited labeled data. Researchers use RL approaches to solve vision

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

tracking issues that are difficult to model or computationally expensive [4], [5], [6].

However, RL methods necessitate a great deal of trial and error, making it difficult to adapt to actual robot control [7]. This is supported by the fact that the agent neither has any prior knowledge about the environment, nor pre-existing data to rely on. Therefore, it necessitates a significant amount of time to spend exploring the environment. To alleviate the problem of insufficient data, the data of expert demonstrations would combine with RL algorithms, resulting in an intuitive framework for the agent that reduces the deployment costs [8]. However, several challenges remain with the use of experts, including the expensive cost of data collection, the inability to generalize to different scenarios and the fact that the agent exploration is restricted by blindly following a single expert [9], [10], [11].

This study proposes a method that avoids forcing the agent to blindly follow the expert behavior, but intelligently restricts actions based on multiple controllers in an online manner. As a result, the agent continues to learn while ignoring actions that are outside the adaptive bounds.

### A. RELATED WORKS

Classical image-based visual servoing (IBVS) and position-based visual servoing (PBVS) methods tend to fail in the presence of noises, disturbances and modelling errors [12]. The literature suggested a number of methods for preventing issues of this type [13], [14], [15], [16]. In [13], the authors developed a method for retaining visual features in the camera FOV that involves switching between PBVS and backward motion approaches. However, the controller suffers from discontinuities when switching happens, and the sensitivity of the PBVS to the camera calibration remains.

The author in [14] presented a visual servoing approach that decouples translation and rotation around the $Z$-axis from the interaction matrix in order to address drawbacks of IBVS method, such as the so-called Chaumette Conundrum and the camera retreat. However, non-optimized trajectories are generated for the manipulator, as a result of converging rotation errors directly to zero from the camera image [15]. Unnecessary movements are reduced in 2 1/2 D visual servoing systems by separating translations from rotations [17]. Nevertheless, these approaches are also computationally expensive and need homography construction, which is susceptible to image noise [14]. In our previous study [18], we proposed a decoupled-hybrid technique that is more robust to image noise and functions, similar to a 2 1/2 D VS. Additionally, the approach enhanced the manipulability (controllability) of the robot which is a crucial factor in stationary robots. Nevertheless, the time-consuming computation of the pseudo-inverse remained a challenge in calculating camera velocity, and consequently joint velocity in classical methods. That is why other studies are vastly focusing on machine learning approaches to solve the aforementioned vision tracking issues [4], [5], [6]. In another study [1], we introduced a hybrid VS technique called hybrid-decoupled VS (HDVS) to address the issue of complex computations. The method has the capability of estimating the pseudo-inverse of the decoupled VS interaction matrix using LoLiMoT (i.e. a neuro-fuzzy neural network). However, HDVS does not take into account the controllability of the robot separately because the outputs of the LoLiMoT neural network are camera velocities and have nothing to do with the joint velocities. In this study, we employed deep reinforcement learning to directly map image features to the joint velocities and make the trained policy robust to image noises and calibration as well as to enhance the performance of VS in both image and robot space. There are several papers in the literature that apply VS using deep RL.

Deep RL-based VS methods are used for various applications such as for multi-rotor aerial robots [2], [19], [20], for mobile robots [5], [21], [22], [23], and for stationary industrial robots [4]. Nevertheless, RL approaches acquire knowledge through extensive trial and error. Demonstrations aid in speeding up and enhancing the learning process.

In the literature, online and offline demonstration information was integrated with RL for robot vision applications. For example, a model-free deep reinforcement learning approach was proposed which uses demonstration data to support a reinforcement learning agent [24]. In [25], a unique tensor-based model was proposed to predict the unseen actions of the expert state sequences by combining RL and data of a demonstrator.

Our proposed method varies from all of these methods in that we create an online hyper-cube over the candidate actions of multiple controllers. By doing so we adaptively reduce the explorations of RL to develop policies that directly map inputs of the camera image to joint velocities. We demonstrate that the learning process is not only greatly accelerated (compared to pure RL), but the policy will also improve the performance of each controller technique that was used to train the policy. More specifically, the joint velocity data of three experts; image-based VS (IBVS), position-based VS (PBVS), and hybrid-decoupled VS (HDVS), are used in the proposed method to limit the action region of the agent and avoid inapplicable actions. In addition, the agent explores further the achieved action bounds from the experts to uncover even more optimal candidates capable of boosting the cumulative rewards. Defining these limitations has nothing to do with the reward function. As a result, the procedure could still be applied even without an explicit reward function using inverse RL method. Our proposed method addresses the issue of limited real-time data availability by itself and has shown promising results. However, it is important to note that there are still opportunities for further improvement in terms of training efficiency. Combining our method with other techniques could potentially enhance the sample efficiency of the training process, leading to a more efficient overall training process.

Domain randomization (DR) is also utilized during the process of learning to adapt the trained policy to real world environments and to make the policy robust in terms of

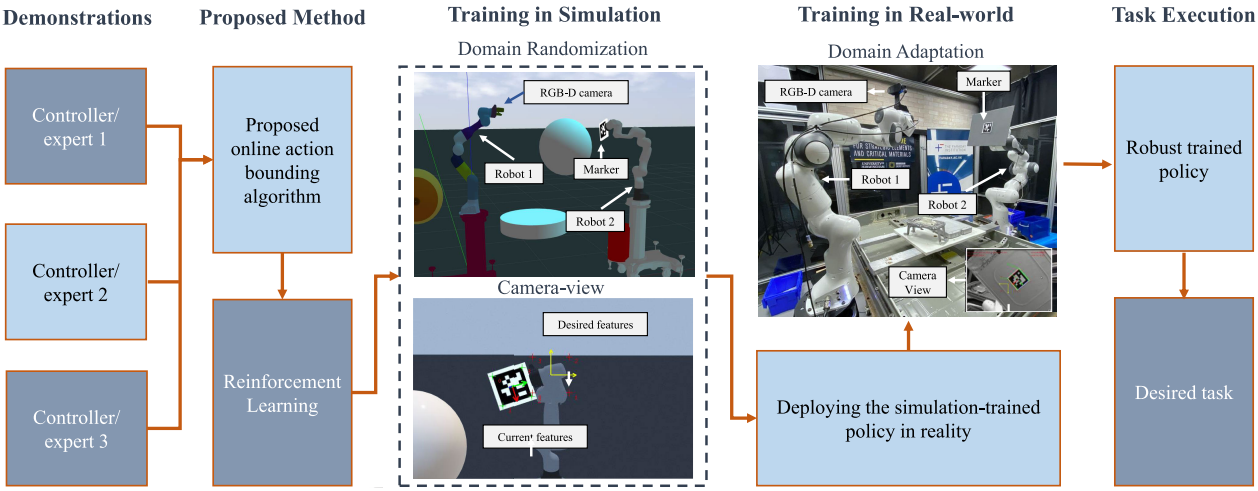


**FIGURE 1.** The outline of the proposed strategy combined with the RL method for a VS application. The Deep Deterministic Policy Gradient (DDPG) agent will be trained by using the DR method. The proposed method employs a combination of three different methods (PBVS, IBVS, and HDVS) as demonstrators to accelerate training and enhance VS performance. Following that, the policy will be further trained with the real robots to adapt to the real world, and the RL agent will be deployed to complete the visual servoing task.

noise, light, and also in the presence of random objects in the camera scene. Therefore, the agent will be trained in simulation, and then the knowledge will be transferred to the real world. Furthermore, domain adaptation (DA) is used as a method of deploying the RL agent to the real world. We demonstrated that the proposed method successfully accelerated the process of learning and enhanced the performance of the controller in both robot base and image base spaces.

Figure 1 illustrates the graphical overview of the proposed method, and the contributions of this study are summarized as follows:

- The proposed method avoids forcing the agent to blindly follow the behavior of a controller, but intelligently restricts actions based on multiple controllers.
- The agent inherits the advantages of each expert while avoiding their shortcomings.
- The agent will explore the constrained action space more thoroughly in order to find even more optimal solutions based on the defined rewards. As a result, implementing the suggested strategy has the potential to perform better than baseline techniques that were employed in the training to develop action bounds.
- Because the action space is achieved using mathematically proven control methods, the possibility of stacking in the local minima while training the agent in the RL algorithm is greatly reduced.

The remainder of this study is structured as follows: Section II provides a brief background about the deployed methods (i.e. VS and RL), followed by a detailed explanation of the proposed method. In Section III, the simulation environment and experimental setup will be introduced. Thereafter, in Section IV, the experimental outcomes and results will be reported and discussed. Finally, the paper is concluded in Section V.

## II. METHODOLOGY

The next section gives a general overview of the baseline visual servoing methodologies and RL algorithm, and then the proposed strategy is thoroughly explained.

### A. VISUAL SERVOING

In our proposed method, the results of three different methods (IBVS, PBVS, HDVS) were combined and used as a supervisor to provide data to the learner. An overview of the methodologies utilized in this paper is detailed in the following paragraphs. The features in the image space immediately provide feedback for the IBVS controller. By considering $\mathbf{v}_{cam}$ as the camera velocity vector, and $\mathbf{e}_i$ as the difference between the current and desired position of $i^{th}$ feature, the control law of the IBVS is calculated as follows [26]:

$$\mathbf{v}_{cam} = -k_i \mathbf{L}_i^{+} \mathbf{e}_i \tag{1}$$

where $k_i$ is the controller gain and $\mathbf{L}_i$ is the image Jacobian matrix used to connect pixel velocity to the camera velocity ($\mathbf{L}_i^{+}$ is the pseudo-inverse representation of $\mathbf{L}_i$). The feedback in PBVS is received from the environment pose reconstruction. The pose estimate will be computed using Euclidean algorithms and camera parameters [27]. The control law of the PBVS is defined as follows:

$$\mathbf{v}_{cam} = -k_p \mathbf{L}_p^{-1} \mathbf{e}_p \tag{2}$$

where $k_p$ is the controller gain and $\mathbf{e}_p$ is the position difference between the current and the desired position of the camera, in the task space. $\mathbf{L}_p(t)$ is a $6 \times 6$ Euclidean matrix to do the camera 3D position estimation; as defined in [27].

For the third method, we used HDVS method that is detailed in [1]. The HDVS method employs both 2D information from image features and their estimated 3D pose. The control law in the HDVS method is calculated by solving the

following equations simultaneously:

$$\mathbf{v}_{xy} = \mathbf{L}_{xy}^{+}\{-k_h\mathbf{e} - \mathbf{L}_r\mathbf{v}_r\} \tag{3}$$

$$\mathbf{v}_r = \mathbf{L}_{Pr}^{+}\left\{-k_h\mathbf{e}_p - \mathbf{L}_{Pxy}\mathbf{v}_{xy}\right\} \tag{4}$$

where $\mathbf{v}_{xy} = [v_x\ v_y]^T$, $\mathbf{v}_r = [v_z\ w_x\ w_y\ w_z]^T$, $k_h$ is the controller gain, and $\mathbf{L}_{xy}$, $\mathbf{L}_r$, $\mathbf{L}_{Pr}$, and $\mathbf{L}_{Pxy}$ are subsets of $\mathbf{L}_i$ and $\mathbf{L}_p$ which are defined in reference [1]. A LoLiMOT neural network was then trained with the collected data from (3) and (4) to provide an accurate estimation of the camera velocities from the feature errors [1].

After calculating the End-Effector (EE) velocities using the robot kinematics, the joint velocities ($\dot{\mathbf{q}}$) will be computed [28]:

$$\dot{\mathbf{q}} = \mathbf{J}^{+\lambda}\xi_c^e\mathbf{v}_{cam}^c \tag{5}$$

where $\lambda$ is a positive scalar known as damping factor [28]. $\xi_c^e$ is a transformation matrix that maps the velocities in the robot end effector (EE) frame to the camera frame (c) [29].

### B. REINFORCEMENT LEARNING

Reinforcement learning is well known for its applications in controlling complex, potentially non-linear systems. The policy is a function that connects observations with actions [30]. The policy parameters are continuously updated by the reinforcement learning algorithm. The goal is to find an optimal policy that is able to maximise the cumulative rewards. The reward indicates how successful an action is in terms of fulfilling the task goal. An RL agent interacts with the actual environment to perform an action $a$ from an action space $A$ and receive state $s$ from a state space $S$, both of which are determined by a policy known as $\pi(a|s)$. A scalar reward $r$ and the preceding state $s'$ are the results of mapping from state $s$ to actions $a$. The rewards function $R(s, a)$, the environment model, and the state transition probability $T(s, a, s') = P(s'|s, a)$ provide the basis for this mapping between states and actions. This process is continued in an episodic problem until the agent reaches a terminal state.

The value function $Q^*(s, a)$, which delivers maximal values in all states, is determined using the Bellman equation [31]:

$$Q^*(s,a) = \mathbb{E}\left[R(s,a) + \gamma\sum_{s'}P\left(s' \mid s,a\right)\max_{a'}Q^*\left(s',a'\right)\right] \tag{6}$$

where $\gamma \in (0, 1]$ is the discount factor, $\mathbb{E}$ is the Bellman expectation, and $Q^\pi(s, a)$ is an estimate of the expected future reward.

DDPG is an algorithm that learns a Q-function and a policy at the same time. The Q function is used to learn the policy using the Bellman equation and off-policy data. Two different NNs are used in DDPG: An actor (target policy) $\pi : S \rightarrow A$, and a critic (action value function approximator) $Q : S \times A \rightarrow R$. Approximating the action-value function $Q^\pi$ of the actor is the duty of the critic [32].

The actor trains using the following loss function:

$$\mathcal{L}_a = -\mathbb{E}_s Q(s, \pi(s)) \tag{7}$$

in which $s$ is sampled from the replay buffer. Furthermore, mini-batch gradient descent on the loss $\mathcal{L}$ is used to train the critic network, which promotes the estimated Q-function to meet the Bellman equation:

$$\mathcal{L} = \mathbb{E}\left(Q\left(s_t, a_t\right) - y_t\right)^2 \tag{8}$$

In (8), $y_t$ is computed using the action which is the output of the actor network:

$$y_t = r_t + \gamma Q\left(s_{t+1}, \pi\left(s_{t+1}\right)\right) \tag{9}$$

Back-propagation through the combined critic and actor networks is used to calculate the gradient of $\mathcal{L}_a$ with respect to the actor parameters [33].

In Figure 2, the structure of the proposed technique for the visual servoing task is outlined. As illustrated in Figure 2, the proposed method takes in current and desired features extracted from a vision sensor as inputs. These features are used to constrain the action space using knowledge from different approaches, namely HDVS, PBVS, and IBVS. This helps to ensure that the actions taken by the system are within acceptable bounds. To enforce these bounds, an action filter block is employed in the DDPG policy. The DDPG policy comprises two blocks: the actor and the critic. The actor block maps the current state to an action, while the critic block evaluates the quality of the action based on the expected reward. The actor and critic blocks are trained using actor and critic loss functions, respectively. During training, if the action generated by the actor block falls outside the bounds set by the knowledge of HDVS, PBVS, and IBVS approaches, the action filter block will remove the action from consideration. This ensures that only actions within the acceptable bounds are considered, leading to better learning outcomes. The joint velocity actions that pass through the action filter block are then applied to the training environment, and the resulting average rewards are calculated.

### C. PROPOSED METHOD: ACTION CONSTRAINED STRATEGY

As described in Section I-A, our proposed method uses implemented control algorithms to limit the action space of the RL agent rather than letting the AI agent to learn from scratch through their own exploration. Joint velocities are defined as RL agent actions (VS commands). The observations are the positions of image feature points, camera poses, and the Jacobian of the robot. Three reward functions are combined to provide the agent reward. The feature errors are driven to zero in the first reward, $r_1$:

$$r_1 = -\sum_{i=1}^{4}\sqrt{(u_i - u_{id})^2 + (v_i - v_{id})^2} \tag{10}$$

where $(u, v)$ denotes the coordinates of a point in the picture and $(u_d, v_d)$ is the desired coordinates of that point. $i = 1$ to

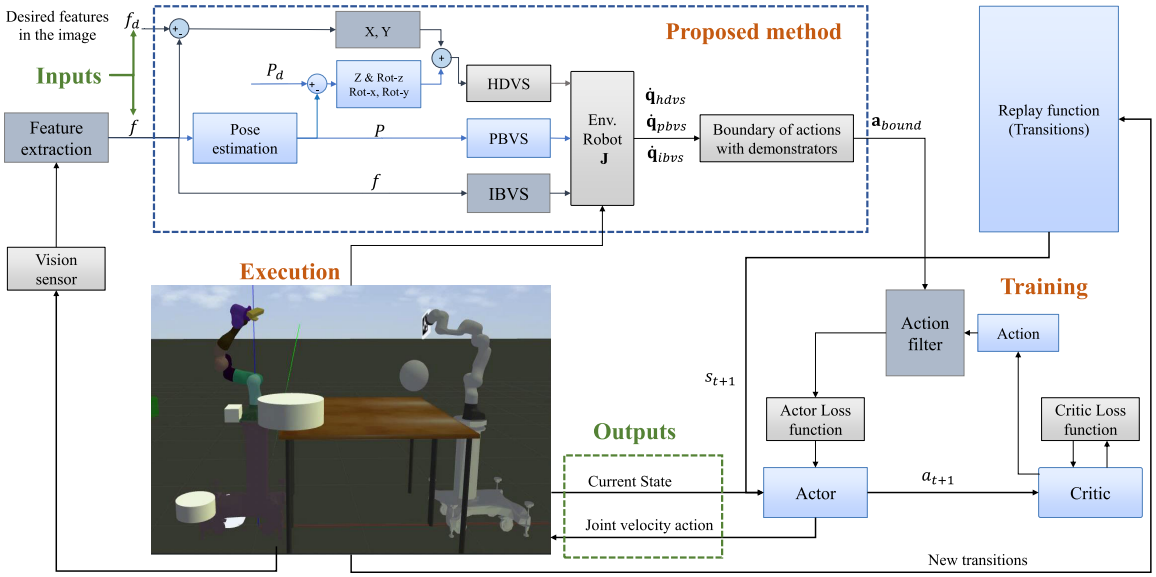


**FIGURE 2.** The proposed method block receives current and desired features that have been extracted from the vision sensor as inputs. Consequently, the knowledge of HDVS, PBVS, and IBVS approaches is used at each episode to constrain the action space. Actions outside the created bounds are filtered using the Action filter block in the DDPG policy. The joint velocity actions will be applied to the training environment, and accordingly average rewards will be calculated.

$n$ is the number of features (in this case $n = 4$ features). The second reward function ($r_2$) is defined to avoid joint limits [34]:

$$r_2(\mathbf{q}) = -\frac{1}{2n}\sum_{j=1}^{n}\left(\frac{q_j - \bar{q}_j}{q_{jM} - q_{jm}}\right)^2 \tag{11}$$

where $\bar{q}_j$ is center of $j^{th}$ joint range. $q_{jM}$ and $q_{jm}$ are the maximum and minimum angles of the $j^{th}$ joint respectively, and $n = 7$ is the number of joints. Singularity avoidance is introduced in the final reward component [34]:

$$r_3(\mathbf{q}) = \sqrt{\det\left(\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q})\right)} \tag{12}$$

where $\mathbf{J}$ is the robot Jacobian. The overall procedure of the proposed learning method is detailed in Algorithm 1. In the proposed acceleration method, the robot achieves the desired joint velocities at the beginning of each step from IBVS, PBVS, and HDVS methods. Thereafter, from combination of these demonstrators, a set of bounds for the action will be defined ($\mathbf{a}_{bound}$): the lower limits of the bound for the $i_{th}$ joint would be $min(\dot{\mathbf{q}}_{ibvs}[i], \dot{\mathbf{q}}_{pbvs}[i], \dot{\mathbf{q}}_{hdvs}[i])$ and the upper limit of the bound would be: $max(\dot{\mathbf{q}}_{ibvs}[i], \dot{\mathbf{q}}_{pbvs}[i], \dot{\mathbf{q}}_{hdvs}[i])$. In this way, the actions which are out of bounds and created from the critic network would be filtered, and the agent continues exploring by trial and error, within the created bounds. We defined a hypercube at each time step (i.e. a subset of the whole space), as restrictive bounds for action space. There are relevant actions inside of that, however the hyper cube is one of the simplest spaces which encapsulate all of the action vectors.

## III. EXPERIMENTAL SETUP

In order to train the policy, an environment was modelled in the simulation (i.e. ROS/Gazebo). Simulations are preferred over real world trials because they provide inexpensive and fast experiments. In addition, using the simulation environment helps to mitigate the risks of damaging the robot setup due to unexpected movements during training. Since the real system is assumed to be one instance in a vast distribution of training variations, the trained model with DR can adapt to the real world environment. As mentioned in I-A, DR is a technique for training a model that works in a variety of simulated settings with randomized properties [35].

The DDPG algorithm was used as a ROS node, and policies were taught using Matlab reinforcement Learning Toolbox [36]. Each control method (IBVS, PBVS, and HDVS) was employed as a distinct ROS node to deliver the actions based on the respective observations. Figure 1 depicts the simulation environment in Gazebo, and the real world setup. The simulation platform includes two Franka robot manipulators; one with eye-in-hand configuration, and another one with a tag marker attached to its EE. The reason for using the second arm was to move the marker into different positions.

An Intel RealSense depth camera D435i was employed as a vision sensor. Two systems were linked together using an Ethernet connection in the same network. One system with the following specifications was utilized for the simulation: AMD Ryzen 7 3700 × 8-core CPU with × 16 threads and a 3.6 GHz base clock. The graphics card (GPU) of another system with the following specifications was used for reinforcement learning and policy training: NVIDIA GTX 1080Ti GPU, Intel (R) Core (TM) i7-8086K 6-core CPU with × 12 threads and a 4GHz base clock with 32 GB installed RAM. Moreover, to accelerate the process of learning, parallel training was used with the help of Parallel Computing Matlab Toolbox [36]. In this study, 12 workers were deployed to create a simulation of the agent in the environment and send data back to the client.

**Algorithm 1** Action Constrained Approach

**Inputs:**
- Joint velocities from IBVS ($\dot{\mathbf{q}}_{ibvs}$), PBVS ($\dot{\mathbf{q}}_{pbvs}$), HDVS ($\dot{\mathbf{q}}_{hdvs}$)

**Outputs:**
- Optimized joint velocities $\mathbf{a}_t$ (actions)

**Given:**
- RL algorithm DDPG
- The strategy for sampling goals from replay
- The reward function

Initialize actor and critic weights randomly ;
Initialize replay buffer R ;
**while** *VS error not converged* **do**
  **for** *episode i=1 to M* **do**
    Sample g (goal) and initial $s_0$ (state);
    **for** *t=0 to T−1* **do**
      **for** *k=1 to 7* **do**
        Get $\mathbf{L}_{xy}$, $\mathbf{L}_r$, $\mathbf{L}_{Pr}$ and $\mathbf{L}_{Pxy}$, $\mathbf{e}_p$, $\mathbf{e}_i$ ;
        Get joint velocities $\dot{\mathbf{q}}_{ibvs}$ from IBVS ;
        Get joint velocities $\dot{\mathbf{q}}_{pbvs}$ from PBVS ;
        Get joint velocities $\dot{\mathbf{q}}_{hdvs}$ from HDVS ;
        Make bounds: $\mathbf{a}_{bound}$ = $[min(\dot{\mathbf{q}}_{ibvs}[i], \dot{\mathbf{q}}_{pbvs}[i], \dot{\mathbf{q}}_{hdvs}[i]), max(\dot{\mathbf{q}}_{ibvs}[i], \dot{\mathbf{q}}_{pbvs}[i], \dot{\mathbf{q}}_{hdvs}[i])]$;
        Sample $a_t$ (action) using DDPG policy and filter actions out of the $\mathbf{a}_{bound}$ bound: $\pi(s_t|g) \rightarrow a_t$ ;
        Execute $a_t$ and observe $s_{t+1}$ (new state) ;
        $r_t := r(s_t, a_t, g)$ ;
        Store $(s_t|g, a_t, r_t, s_{t+1}|g)$ (transition) in R ;
        Sample $g'$ (additional goal) for replay $G : S$(current episode) ;
      **end**
      **for** $g' \in G$ **do**
        $r' := r(s_t, a_t, g')$ ;
        Store $(s_t|g', a_t, r', s_{t+1}|g')$ in R ;
      **end**
      **for** *t=1 to N* **do**
        Sample B (mini-batch) from the R (replay buffer) ;
        Execute one step of optimization using DDPG and B ;
      **end**
    **end**
  **end**
**end**

## IV. RESULTS AND DISCUSSIONS

To show the efficacy of our suggested strategy (detailed in Section II-C), we compare the training progress and results of RL without the proposed strategy (agent-1) and with the proposed strategy (agent-2).

**TABLE 1. Employed RL and noise parameters in the training.**

| RL parameters | | Noise options | |
|---|---|---|---|
| Target smooth factor | 0.001 | Mean | 0 |
| Target update freq | 1 | Mean Attraction constant | 5 |
| Sample time | 0.025 | Variance decay rate | 0.00001 |
| Discount factor | 0.95 | Variance | 0.5 |

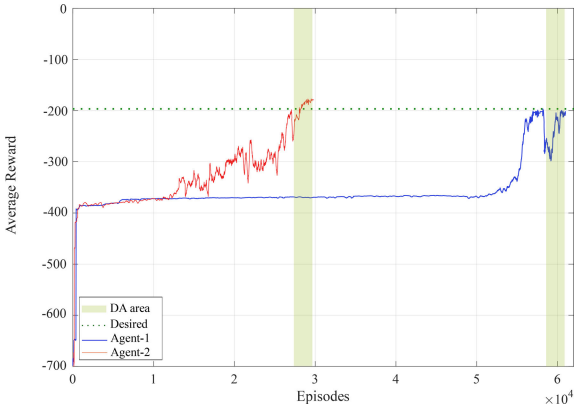


**FIGURE 3. Training graph without using any expert (agent-1, blue line), and with using three experts to constrain the action space (agent-2, red line).**

For the sim-to-real task, DR was used which has been identified as the most commonly used strategy for improving simulation realism. The training was carried out for the task of VS (i.e. tracking the features of a target in the camera screen). The initial position of the first robot (the robot with the camera mounted on its wrist) was randomized in each episode in order to generalize the trained policy better. The desired threshold of the average reward was defined to be −200 (determined from preliminary experiments). The agent restarted the episode in case of meeting one of the following criteria during training: (I) when the robot is close to the joint limits, (II) when the features are 10% close to the image boundary, (III) when the robot Jacobian manipulability is too small (less than 0.01), and finally (IV) when the number of steps in each episode surpasses 400. The used RL parameters are defined in Table 1.

As illustrated in Figure 3, the agents have learned to maximize the cumulative reward over time. According to Figure 3, it takes the agent approximately 57800 episodes and 6 million steps for the average reward to exceed the desired threshold (−200). However, it is shown that approximately 28400 episodes and 3 million steps are required to achieve the same average reward of −200 by combining the proposed method with RL. Thereafter, using the so-called domain adaptation approach, we let the trained policies train for 1500 more real world episodes using actual robots (green area in Figure 3). Both agent-1 and agent-2 reached the award of −200 before domain adaptation, however agent-2 achieved a higher reward after domain adaptation. Due to the change of environment (i.e. sim-to-real), the reward values dropped for both agents after DA, as shown in Figure 3. Results suggest that agent-2 is faster and achieved a more effective policy (higher average reward) than agent-1.

**TABLE 2.** The performance comparison of visual servoing techniques derived by averaging 50 trials (10 trails per method). The bold results represent the best candidate for each column.

| Method | RMSE of 2D Errors | Feature Error Range | RMSE of Position ($m$) | RMSE of Orientation ($deg$) | Camera Travelled Distance ($m$) | Manipulability Mean | Manipulability Range | Iterations | Average Reward |
|---|---|---|---|---|---|---|---|---|---|
| IBVS | **0.0222** | **[-0.36, 0.31]** | 0.036 | 9.43 | 0.942 | 0.0407 | [0.014, 0.081] | 292 | -309.42 |
| PBVS | 0.0383 | [-0.44, 0.51] | **0.022** | **6.54** | **0.722** | 0.0446 | [0.024, 0.080] | 387 | -316.23 |
| HDVS | 0.0273 | [-0.45, 0.49] | 0.034 | 8.41 | 0.917 | 0.0396 | [0.021, 0.081] | 424 | -275.19 |
| Agent-1 | 0.0458 | [-0.52, 0.62] | 0.046 | 15.3 | 1.140 | 0.0321 | [0.013, 0.075] | 383 | -223.03 |
| **Agent-2** | 0.0258 | [-0.34, 0.34] | 0.030 | 7.82 | 0.839 | **0.0492** | **[0.029, 0.082]** | **253** | **-184.21** |

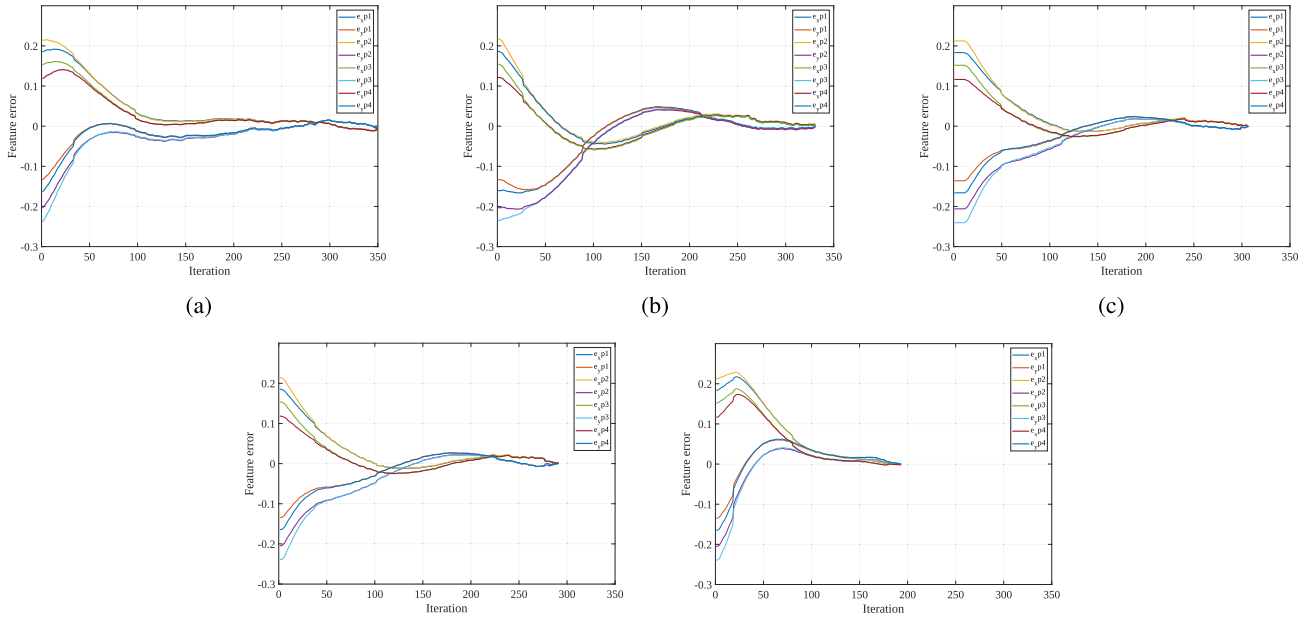


**FIGURE 4.** Comparison of visual servoing feature errors with various approaches for one random trial. The error between the desired and the current features (a) in the IBVS method, (b) in the PBVS method, (c) in the HDVS method, (d) with agent-1, (e) with agent-2. It should be noted that $e_{xPi}$ and $e_{yPi}$ represent the x and y components of the $i^{th}$ feature, respectively.

Table 2 compares the effective parameters in the performance of individual VS methods and the trained policies. These parameters are derived by averaging 50 trials with 10 different initial positions of the robot equipped with the camera. These initial positions were chosen randomly with the condition of having all four features visible in the image frame. All experiments are duplicated under the same condition for IBVS, PBVS, HDVS, agent-1, and agent-2. As shown in Table 2, IBVS shows an optimized behaviour in 2D image space, because it has the smallest Root Mean Square Error (RMSE) than other methods. Moreover, the smaller range of feature errors in IBVS confirms that the chance of losing the target object from the camera FOV in this method is lower than in the other four methods. However, agent-2 is still faster than IBVS based on the number of iterations. Not to mention that in VS both image space and robot space should be considered, and this is where the suggested method surpasses the IBVS. Following IBVS, the trained policy with agent-2 performs more optimally (smaller RMSE, smaller feature error range, and fewer iterations) than other methods. As a result, agent-2 outperformed PBVS, HDVS, and agent-1 in the image space.

From Table 2, the average mean value of manipulability was 0.0492 for agent-2, 0.0407 for IBVS, 0.0446 for PBVS, 0.0396 for HDVS, and 0.0321 for agent-1. Therefore, agent-2 performs better in terms of manipulability in comparison with the other four approaches. From Table 2, it would also be inferred that the robot has better controllability with agent-2 while tracking the target due to higher manipulability bounds created by agent-2.

According to Table 2, the policy trained with agent-2 has a shorter camera path than IBVS, HDVS, and agent-1. The robot end-effector (EE) travelled distance is 0.839m with agent-2. This value is 0.942m, 0.917m, and 1.14m, in IBVS, HDVS, and agent-1, respectively. In PBVS, the camera travelled distance was 0.722m, and thus provided the shortest EE trajectory. Agent-2 outperforms IBVS, HDVS, and agent-1 in the Cartesian (robot) space. However, still better results are achieved using the PBVS method in terms of EE travelled distance. It should be mentioned that the PBVS method has other drawbacks like losing the object from the camera FOV, high sensitivity to the camera parameters, and its sub-optimal performance in the image space, while the trained policy with agent-2 has solved these drawbacks. Another inference from Table 2 is that the RMSE of position and orientation with the

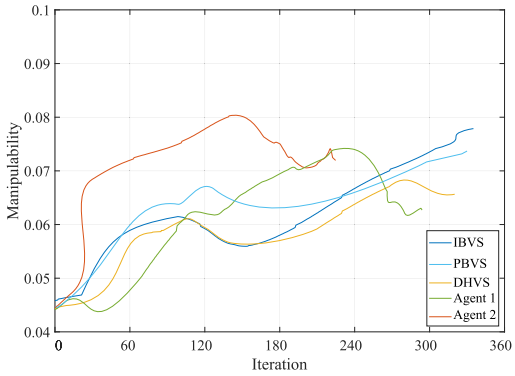


**FIGURE 5.** The manipulability of five different VS methods.

trained policy of agent-2 is smaller than IBVS, HDVS, and agent-1 policy which indicates a more optimized trajectory created by agent-2.

In conclusion, agent-2 achieved an optimal overall performance over the image space and cartesian space, where PBVS and IBVS suffer respectively. Additionally, using our suggested approach the agent offers the best controllability among the other techniques. Furthermore, comparing the average rewards in Table 2, agent-2 performs 17.4% better than agent-1, 33.1% better than HDVS, 41.7% better than PBVS, and 40.5% better than IBVS.

We selected a random trial in order to demonstrate the error convergence and robot manipulability for all five VS methods. Figure 4 illustrates the convergence error for all VS methods. By comparing the RMSE for this trial, IBVS achieved an RMSE of 0.024; comparatively, this value was 0.041 for PBVS, 0.0276 for HDVS, 0.043 for agent-1, and 0.0224 for agent-2. The RMSE for agent-2 is even better than this value for the IBVS approach. Moreover, the task is completed in 350 iterations with IBVS, 341 iterations with PBVS, 308 iterations with HDVS, 292 iterations with agent-1, and 197 iterations with agent-2. As a result, agent-2 offers a faster solution than all other approaches, demonstrating that the trained policy with agent-2 not only inherits good IBVS performance in terms of RMSE but also learns to operate faster.

Furthermore, the manipulability of the robot arm for all five methods is plotted in Figure 5 for the same trial illustrated in Figure 4. In Figure 5, the manipulability of the RL methods with agent-1 and agent-2 are higher than other IBVS, PBVS, and HDVS methods. Agent-2 offers the highest manipulability compared to the other four methods in most robot configurations. As a result, there would be more controllability for the robot joints using agent-2. In Figure 5, the mean manipulability for IBVS, PBVS, HDVS, agent-1, and agent-2 were 0.062, 0.0613, 0.642, 0.0662, and 0.0742, respectively.

## V. CONCLUSION

This study is an extension to our previously proposed method in [1]. We enhanced the performance of VS by combining the knowledge of multi-controllers with RL. In our proposed approach, the online action space of the agent is constrained with the help of multi-control approaches. Thereafter, the agent explores further within the created bounds to find more optimised solutions and accumulate more rewards. Therefore, the learning process will be significantly accelerated to achieve a reasonable level of performance. Additionally, domain randomisation and domain adaptation are used to make the strategy more robust for the sim-to-real transition. The results show that the proposed method achieves better overall performance in terms of feature trajectories in the 2D image screen, and robot trajectory in the 3D task space, with the higher Jacobian and manipulability of the robot. Trained visual servoing policy also incorporates the capabilities of RL methods to eliminate modeling and calculation difficulties. The proposed action constrained strategy would be used when there are multiple algorithmic control demonstrators (from two to an infinite number). This strategy improves the performance of an RL method and reduces training time by providing various samples from demonstrators.

In future work, we aim to generalize our proposed multi-expert method to be used for a variety of applications which will significantly reduce training episodes by bounding the action apace of each agent. Moreover, Exploring the use of additional experts with varying levels of competence or from different domains could be a potential avenue for future research.

## ACKNOWLEDGMENT

The authors would like to thank Christopher Gell and Jamie Hathaway for their technical support. *(Ali Aflakian and Alireza Rastegarpanah contributed equally to this work.)*

## REFERENCES

[1] A. Rastegarpanah, A. Aflakian, and R. Stolkin, "Optimized hybrid decoupled visual servoing with supervised learning," *Proc. Inst. Mech. Eng. I, J. Syst. Control Eng.*, vol. 236, no. 2, 2021, Art. no. 09596518211028379.

[2] C. Sampedro, A. Rodriguez-Ramos, I. Gil, L. Mejias, and P. Campoy, "Image-based visual servoing controller for multirotor aerial robots using deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 979–986.

[3] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[4] F. Castelli, S. Michieletto, S. Ghidoni, and E. Pagello, "A machine learning-based visual servoing approach for fast robot control in industrial setting," *Int. J. Adv. Robot. Syst.*, vol. 14, no. 6, 2017, Art. no. 1729881417738884.

[5] Z. Jin, J. Wu, A. Liu, W.-A. Zhang, and L. Yu, "Policy-based deep reinforcement learning for visual servoing control of mobile robots with visibility constraints," *IEEE Trans. Ind. Electron.*, vol. 69, no. 2, pp. 1898–1908, Feb. 2022.

[6] A. M. Farahmand, A. Shademan, M. Jagersand, and C. Szepesvári, "Model-based and model-free reinforcement learning for visual servoing," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 2917–2924.

[7] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2017, *arXiv:1707.08817*.

[8] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, p. 17, 2018.

[9] T. Takeda, Y. Hirata, and K. Kosuge, "Dance step estimation method based on HMM for dance partner robot," *IEEE Trans. Ind. Electron.*, vol. 54, no. 2, pp. 699–706, Apr. 2007.

[10] S. Krishnan, A. Garg, R. Liaw, B. Thananjeyan, L. Miller, F. T. Pokorny, and K. Goldberg, "SWIRL: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards," *Int. J. Robot. Res.*, vol. 38, nos. 2–3, pp. 126–145, Mar. 2019.
[11] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 4565–4573.
[12] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control*. Springer-Verlag, 1998, pp. 66–78.
[13] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino, "Keeping features in the field of view in eye-in-hand visual servoing: A switching approach," *IEEE Trans. Robot.*, vol. 20, no. 5, pp. 908–914, Oct. 2004.
[14] P. I. Corke and S. A. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Trans. Robot. Autom.*, vol. 17, no. 4, pp. 507–515, Aug. 2001.
[15] N. R. Gans and S. A. Hutchinson, "Stable visual servoing through hybrid switched-system control," *IEEE Trans. Robot.*, vol. 23, no. 3, pp. 530–540, Jun. 2007.
[16] A. Rastegarpanah, J. Hathaway, and R. Stolkin, "Vision-guided MPC for robotic path following using learned memory-augmented model," *Frontiers Robot. AI*, vol. 8, Jul. 2021, Art. no. 688275.
[17] E. Cervera, A. P. D. Pobil, F. Berry, and P. Martinet, "Improving image-based visual servoing with three-dimensional features," *Int. J. Robot. Res.*, vol. 22, nos. 10–11, pp. 821–839, Oct. 2003.
[18] A. Rastegarpanah, A. Aflakian, and R. Stolkin, "Improving the manipulability of a redundant arm using decoupled hybrid visual servoing," *Appl. Sci.*, vol. 11, no. 23, p. 11566, Dec. 2021.
[19] H. Shi, L. Shi, G. Sun, and K.-S. Hwang, "Adaptive image-based visual servoing for hovering control of quad-rotor," *IEEE Trans. Cogn. Develop. Syst.*, vol. 12, no. 3, pp. 417–426, Sep. 2020.
[20] C. Hu, W. Cao, and B. Ning, "Visual servoing with deep reinforcement learning for rotor unmanned helicopter," *Int. J. Adv. Robot. Syst.*, vol. 19, no. 2, 2022, Art. no. 17298806221084825.
[21] C. Gaskett, L. Fletcher, and A. Zelinsky, "Reinforcement learning for visual servoing of a mobile robot," in *Proc. Austral. Conf. Robot. Autom. (ACRA)*, 2000, pp. 1–6.
[22] Y. Li and J. Košecka, "Learning view and target invariant visual servoing for navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 658–664.
[23] R. S. Sharma, R. R. Nair, P. Agrawal, L. Behera, and V. K. Subramanian, "Robust hybrid visual servoing using reinforcement learning and finite-time adaptive FOSMC," *IEEE Syst. J.*, vol. 13, no. 3, pp. 3467–3478, Sep. 2019.
[24] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess, "Reinforcement and imitation learning for diverse visuomotor skills," 2018, *arXiv:1802.09564*.
[25] X. Guo, S. Chang, M. Yu, G. Tesauro, and M. Campbell, "Hybrid reinforcement learning with expert state sequences," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 3739–3746.
[26] G. Hu, N. R. Gans, and W. E. Dixon, "Adaptive visual servo control," Dept. Mech. Aerosp. Eng., Univ. Florida, Gainesville, FL, USA, Tech. Rep., 2009, pp. 42–63.
[27] E. Malis, F. Chaumette, and S. Boudet, "2 1/2 D visual servoing," *IEEE Trans. Robot. Autom.*, vol. 15, no. 2, pp. 238–250, Apr. 1999.
[28] P. Baerlocher and R. Boulic, "Task-priority formulations for the kinematic control of highly redundant articulated structures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Innov. Theory, Pract. Appl.*, vol. 1, Oct. 1998, pp. 323–329.
[29] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Hoboken, NJ, USA: Wiley, 2006.
[30] L. Wells and T. Bednarz, "Explainable AI and reinforcement learning—A systematic review of current approaches and trends," *Frontiers Artif. Intell.*, vol. 4, p. 48, May 2021.
[31] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Leibo, and A. Gruslys, "Deep *Q*-learning from demonstrations," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–8.
[32] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," 2017, *arXiv:1707.01495*.
[33] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[34] L. Huo and L. Baron, "The joint-limits and singularity avoidance in robotic welding," *Ind. Robot, Int. J.*, vol. 35, no. 5, pp. 456–464, Aug. 2008.
[35] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.
[36] *MATLAB Version 9.9.0 (R2020b)*, MathWorks, Natick, MA, USA, 2021.

**ALI AFLAKIAN** received the B.Sc. degree in mechanical engineering from Isfahan University of Technology (IUT), Iran, and the M.Sc. degree in mechatronics engineering from the University of Tehran, Iran. He is currently pursuing the Ph.D. degree in robotics with the University of Birmingham, U.K. He is working on a project called "Reuse and Recycling of Lithium-ion Batteries (RELIB)" with a focus on automating the process of disassembly of Lithium-ion batteries using advanced robotics and AI techniques. He is aspiring to use the most state-of-the-art artificial intelligence approaches in the tasks done by robots. His research interests include robotics, machine learning, computer vision, and human–robot interaction.

**ALIREZA RASTEGARPANAH** is currently a Senior Research Fellow in robotics with The Faraday Institution, University of Birmingham. He is an interdisciplinary engineer with diverse research interests, including robotics, physical human–robot interaction, machine vision, and machine learning. He is leading the Control and Manipulation Team, Extreme Robotics Laboratory, which focuses on automating the process of disassembly of complex products, such as electric vehicle batteries. He is actively involved in applied robotics projects within the industry. He has funded collaborations with the industry and currently acting as a CoI of REBELION project—"Research and development of a highly automated and safe streamlined process for increased lithium-ion battery repurposing and recycling."

**RUSTAM STOLKIN** (Member, IEEE) is currently the Chairperson of the Expert Group on Robotic and Remote Systems, the Chair of Robotics, a Royal Society Industry Fellow, a Professor in robotics at the University of Birmingham, and the Head of the Extreme Robotics Laboratory. He is an interdisciplinary engineer with diverse research interests, although mainly focuses on robotics. His main research interests include vision and sensing, robotic grasping and manipulation, robotic vehicles, human–robot interaction, AI, and machine learning. He is well-known internationally, with an extensive track record in leading major robotics research programs that are directly linked to industrial challenges and industrial stakeholders.