



# A Learning-based Control Framework for Fast and Accurate Manipulation of a Flexible Object

Junyi Wang<sup>1,2</sup> · Xiaofeng Xiong<sup>2</sup> · Silvia Tolu<sup>1</sup> · Stanislav N. Gorb<sup>3</sup>

Received: 11 October 2023 / Revised: 3 April 2024 / Accepted: 17 April 2024 / Published online: 21 May 2024  
© The Author(s) 2024

## Abstract

This paper presents a learning-based control framework for fast ( $< 1.5$  s) and accurate manipulation of a flexible object, i.e., whip targeting. The framework consists of a motion planner learned or optimized by an algorithm, Online Impedance Adaptation Control (OIAC), a sim2real mechanism, and a visual feedback component. The experimental results show that a soft actor-critic algorithm outperforms three Deep Reinforcement Learning (DRL), a nonlinear optimization, and a genetic algorithm in learning generalization of motion planning. It can greatly reduce average learning trials (to  $< 20\%$  of others) and maximize average rewards (to  $> 3$  times of others). Besides, motion tracking errors are greatly reduced to 13.29% and 22.36% of constant impedance control by the OIAC of the proposed framework. In addition, the trajectory similarity between simulated and physical whips is 89.09%. The presented framework provides a new method integrating data-driven and physics-based algorithms for controlling fast and accurate arm manipulation of a flexible object.

**Keywords** Deep reinforcement learning · Deformable object manipulation · Variable impedance control · Sim2real · Visual tracking

## 1 Introduction

Humans and animals exhibit remarkable skills in manipulating tools and their bodies, such as striking whips or tails to accurately hit targets less than 1.5 s. These rapid and precise movements with complex dynamics are not fully understood.

Deciphering these dynamics can not only advance robotic arm control, but also offer new insights into the mechanisms behind fast, accurate movement control.

Biologically, the manipulation of elongated structures like arms, tails, or objects is common. These systems typically consist of many flexibly connected segments, often powered by muscles, as seen in snake bodies [1], reptile and mammal tails [2, 3], and the legs of harvestmen [4]. Research such as that whip and hand movements had been investigated and analyzed in arm targeting [5], yet the intricacies of motion planning and muscle-mimicking control are still largely unexplored.

In robotics, manipulating flexible objects represents a significant challenge due to computational and dynamic complexities introduced by their high flexibility. While most research has concentrated on slower, general manipulations, like folding laundry [6], setting tablecloths [7], and performing surgical operations [8, 9], few studies have addressed fast, goal-directed control with flexible objects [10–12]. Furthermore, these studies have largely overlooked the potential of reinforcement learning and optimization algorithms.

Our work contributes to the State-Of-The-Art (SOTA) by merging data-driven and physics-based algorithms to facilitate fast and accurate movement control. It reviews and

---

✉ Xiaofeng Xiong  
xizi@mmmi.sdu.dk

Junyi Wang  
s212633@dtu.dk

Silvia Tolu  
stolu@dtu.dk

Stanislav N. Gorb  
sgorb@zoologie.uni-kiel.de

<sup>1</sup> Department of Electrical and Photonics Engineering, Technical University of Denmark, Anker Engelsevej 55, Kongens Lyngby, Zealand 2800, Denmark

<sup>2</sup> SDU Biorobotics, Mærsk Mc-Kinney Møller Institute, University of Southern Denmark (SDU), Campusvej 55, Odense M, Funen 5230, Denmark

<sup>3</sup> Department Functional Morphology and Biomechanics, Zoological Institute, University of Kiel, Am Botanischen Garten 1–9, D-24118 Kiel, Schleswig-Holstein, Germany

**Table 1** Comparison between our proposed and SOTA methods

Method	Ours	Zhang et al.	Nah et al.	Zimmermann et al.	Chi et al.
Year	2023	2021	2020	2021	2023
DoFs <sup>1</sup>	2	6	2	7	2 <sup>2</sup>
Whip targeting	✓	✗	✓	✓	✓
OIAC	✓	✗	✗	✗	✗
DRL	✓	✗	✗	✗	✗
NLO <sup>3</sup>	✓	✗	✗	✗	✗
GA <sup>4</sup>	✓	✓	✓	✓	✓
Visuomotor control	✓	✓	✗	✗	✓
Real robot	✓	✓	✗	✓	✓

*DoFs* Degrees of Freedom

Only two DoFs of a UR robot were controlled

*NLO* Non-Linear Optimization

*GA* Genetic Algorithm

compares existing works on targeted robotic manipulation of flexible objects (see below), with a particular focus on the control of a 2-jointed arm manipulating a flexible whip. While this research specifically addresses whip manipulation, the proposed control framework is versatile enough to be applied to end-effector space control of robots with a high degree of freedom, broadening our understanding and capabilities in both biological and robotic domains of fast, accurate manipulation.

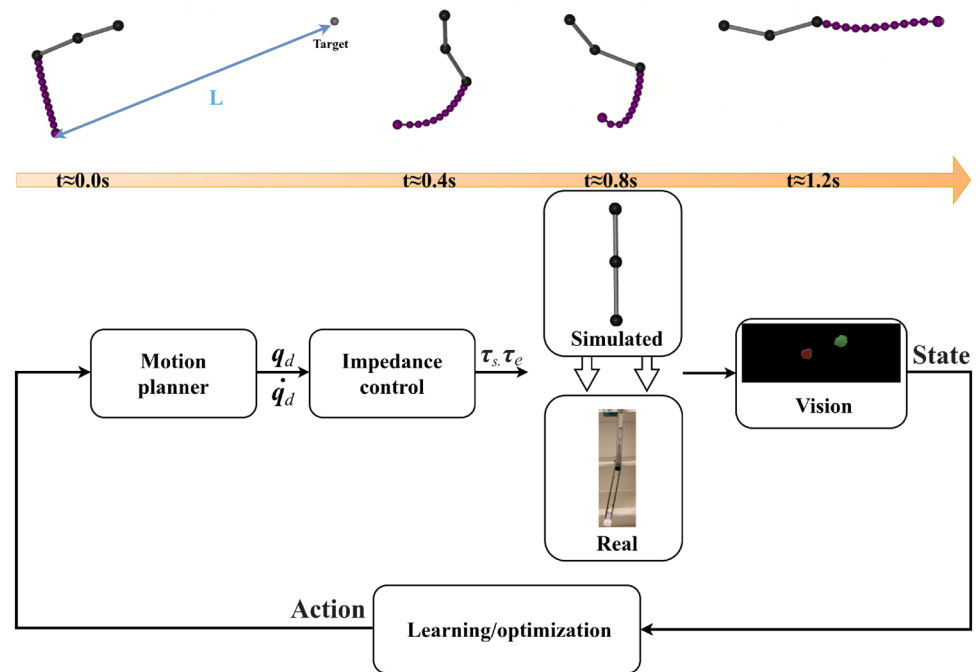
The research in robotic manipulation of deformable objects focuses mainly on slow control tasks such as folding cloth and surgery operations [8, 9]. In this paper, we focus on fast and accurate whip targeting. “Accurate” means that a human arm is controlled for striking a whip tip to hit a target in space, rather than other parts of a whip. The related SOTA works of manipulating flexible objects (e.g., cables) to hit targets are introduced as follows and compared to our proposed framework (see Table 1). A self-supervised learning framework was proposed for a robot arm to manipulate cables to knock cups [10]. Their method is based on the use of iteration learning with a convolutional neural network. Moreover, the movement is not accurate and fast ( $< 1.5$  s) targeting, since the manipulated cable tip was fixed to a wall. To achieve whip targeting, a simple (non-learning) control framework based on dynamic primitives was presented, in which a DIRECT global method was applied to optimize joint motion parameters (e.g., duration) with constant impedance control [13]. The framework was validated only in a simulated robot arm. Besides, there is no investigation into motion learning and adaptation. A batch Newton method and differential dynamic programming were compared in robotic trajectory optimization of manipulating a soft rod to hit wooden blocks [14]. The rod flexibility is not comparable to a whip used for accurate and fast targeting. In addition, their trajectory optimization methods are based on predefined wooden block positions, rather than visual feedback. They were not compared with learning approaches, e.g., reinforcement learning [15]. Their optimization methods only fit to the fixed targets and one of them can not reach convergence. The iterative residual policy, a general learning

framework was proposed for controlling two joints (of a UR5 robot) to manipulate ropes to hit targets [11]. However, the framework was validated with prior constraint knowledge in physical robot whipping, since learning converges only after three iterations in some experimental scenarios. Moreover, the framework is not data-efficient in simulated whipping control, since it heavily depends on big training datasets, i.e., 54 million trajectories. Furthermore, the learned motion trajectories may fall into local optimality, because a simple (greedy) motion update rule was adopted. In addition to control, how to bridge the gap between simulated and real robot whip targeting remains an open issue in deformable object manipulation [16]. This is because training learning-based controllers is effortfully costly in real, and distorted in simulated modeling of flexible objects. Sim2real of robotic whip targeting control will be presented and analyzed in this paper. Other related works and their sim2real have been introduced in comprehensive reviews [17–19]. Most of them focus mainly on slow manipulation of deformable objects (e.g., cloth), rather than fast ( $< 1.5$  s) and targeted movement control guided by visual feedback in this paper.

The main motivation of this work is to reverse-engineer motion learning and muscle control in a fast targeting behavior by a flexible tool, which has not yet been well investigated. It will push not only scenario boundary from slow and non-targeted to fast and targeted manipulation control of flexible tools, but also methodology from a data-driven or physics-based algorithm to their integration. These progresses advance the understanding of motion learning and muscle-like control on sophisticated manipulation of bionic systems. Its significance depends on a novel learning-based framework for achieving fast and accurate manipulation by combining data-driven and physics-based algorithms. It consists of a motion planner learned or optimized by an algorithm, an online impedance adaptation controller, a sim2real mechanism, and a visual feedback mechanism (see Fig. 1). Its specific novelties lie in,

1. Integration of data-driven and physics-based methods guided by visual feedback. Six data-driven methods are applied, consisting of four SOTA DRLs [20–23], a deterministic optimization [24], and a genetic algorithm [25]. Each of them is integrated into a physics-based control method, i.e., OIAC. The integration imitates human arm motion planning and (online) adaptation in whip targeting;
2. Sim2real for accurate and fast movement learning and control. Our proposed framework is validated from simulated and real whip targeting control. The trajectories of the whip tips are 89.09% similar in the simulated and real environment, therefore, validating a feasible transfer from simulated to real whip targeting control.

**Fig. 1** The LEARNING-based MOvement coNtrol (LEMON) framework for a whip-targeting task. The motion planner provides the desired joint positions  $q_d \in \mathbb{R}^{1 \times 2}([q_s, q_e])$  and velocities  $\dot{q}_d \in \mathbb{R}^{1 \times 2}([\dot{q}_s, \dot{q}_e])$  that are sent to an online adaptive impedance controller to generate the joint torque commands  $[\tau_s, \tau_e]$ . The learning/optimization objective is to minimize the distance  $L$  between a whip tip and a target, guided by the visual feedback (state)



The rest of this article is organized as follows. The proposed methods are described in Sect. 2. The experimental results of the simulated and real whip targeting are introduced in Sect. 3. Sections 4 and 5 contain a comprehensive discussion and conclude the article, respectively.

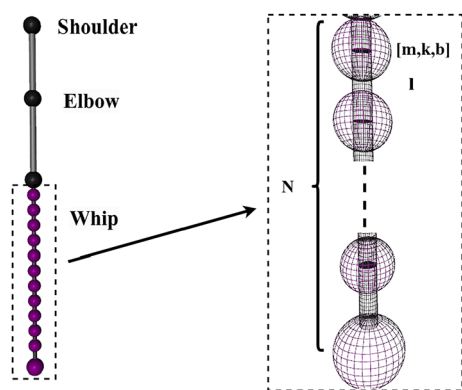
## 2 Methods

Simulated and physical two-jointed arms were controlled to strike a whip to hit targets, respectively. In the controller (see Fig. 1), arm joint movements and duration are learned or optimized by an algorithm, while the joint impedance (control) is online modulated. The learning or optimization process was performed in simulation and later validated in the physical whip targeting.

### 2.1 Simulation

A whip targeting control environment was built into the advanced physics simulator MuJoCo, which outperforms its counterparts in more accurate and faster modeling [26]. Moreover, it is more efficient to perform Sim2real in DRL,

i.e., to train DRL algorithms in simulations and then to transfer knowledge into real robots [27, 28]. A ‘lumped-parameter’ model (Fig. 2) was built to simulate complex whip continuum dynamics, which consists of a series of  $N$  rotational springy and damped nodes. Each sub-model is comprised of a rotational joint and an ideal point-mass  $m$  [kg], which connects a massless cylinder with length  $l$  [m]. A rotational spring with  $k$  [ $N \cdot m/rad$ ] and a rotational damper with  $b$  [ $N \cdot m \cdot s/rad$ ] consists of a rotational joint, and  $N$  of these discrete sub-models are connected serially to comprise the experimentally-fitted whip [13]. The whip parameters  $[N, l, m, k, b]$  can be seen in Table 2. The parameters are adopted from [29], which have been numerically validated in real whip (targeting) kinematics [5]. The  $N$ -node whip model is connected to the upper-limb model with a freely-rotating hinge. Note that complex whip dynamics are unknown for the following motion planner and control system.



**Fig. 2** The combined planar upper-limb and whip model (see its dimension in Table 2)

## 2.2 Motion Planner and Control

### 2.2.1 Zero-Torque Trajectory

The desired upper-limb joint angles follow a minimum-jerk trajectory given by,

$$\mathbf{q}_d(t) = \mathbf{q}_i + (\mathbf{q}_f - \mathbf{q}_i) \left\{ 10 \left( \frac{t}{D} \right)^3 - 15 \left( \frac{t}{D} \right)^4 + 6 \left( \frac{t}{D} \right)^5 \right\}, \quad (1)$$

where  $\mathbf{q}_d(t) \in \mathbb{R}^2$  is the joint position;  $\mathbf{q}_i$  and  $\mathbf{q}_f$  denote the initial and final zero-torque joint positions, respectively;  $t$  is the time;  $D$  means the trajectory duration time; If the current time  $t$  is longer than  $D$ , the zero-torque trajectory of joints would remain at the final positions  $\mathbf{q}_f$ . The planned motion parameters  $\mathbf{q}_i$ ,  $\mathbf{q}_f$ , and  $D$  were learned or optimized by one of six applied algorithms (see them in Sect. 2.3).

### 2.2.2 OIAC[30]

A first-order impedance controller with gravity compensation is used in the upper-limb model,

$$\boldsymbol{\tau} = \mathbf{K}(\mathbf{q}_d - \mathbf{q}) + \mathbf{B}(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \boldsymbol{\tau}_G, \quad (2)$$

$$\boldsymbol{\tau}_G = \mathbf{J}_s^T \mathbf{f}_s + \mathbf{J}_e^T \mathbf{f}_e + \mathbf{J}_w^T \mathbf{f}_w, \quad (3)$$

where  $\mathbf{q} \in \mathbb{R}^2$  is two joints displacement vector;  $\boldsymbol{\tau}_G$  denotes gravity compensation for the upper-limb and whip model;  $\mathbf{J}$  and  $\mathbf{f}$  denote the jacobian matrices and the gravitational forces on each part; The subscripts  $s$ ,  $e$ , and  $w$  correspond to the shoulder joint, elbow joint, and whip, respectively;  $\mathbf{K} \in \mathbb{R}^{2 \times 2}$ ,  $\mathbf{B} \in \mathbb{R}^{2 \times 2}$ , which represent both the shoulder joint and elbow joint neuromuscular mechanical impedance; The proposed OIAC aims at online adapting  $\mathbf{K}$  and  $\mathbf{B}$  impedance matrices to the joint tracking error reduction,

$$\mathbf{K}(t) = \begin{bmatrix} k_{ss}(t) & k_{se}(t) \\ k_{es}(t) & k_{ee}(t) \end{bmatrix} \quad \mathbf{B}(t) = \begin{bmatrix} b_{ss}(t) & b_{se}(t) \\ b_{es}(t) & b_{ee}(t) \end{bmatrix} \quad (4)$$

where the diagonal and off-diagonal terms correspond to the impedance generated by the monoarticular and biarticular muscles of the upper limb, respectively.

$$\mathbf{J} = \mathbf{J}_c + \mathbf{J}_p, \quad (5)$$

$$\mathbf{J}_c = \frac{1}{2} \int_0^D \left( \|\text{vec}(\tilde{\mathbf{K}})\|_{\mathbf{Q}_K}^2 + \|\text{vec}(\tilde{\mathbf{B}})\|_{\mathbf{Q}_B}^2 \right) dt, \quad (6)$$

$$\mathbf{J}_p = \int_0^D \dot{V}(t) dt \quad V(t) = \frac{1}{2} \boldsymbol{\varepsilon}^T(t) \mathbf{I}(\mathbf{q}) \boldsymbol{\varepsilon}(t), \quad (7)$$

where the cost function of OIAC is  $J$ ;  $J_c$  is the weighted norms of matrices  $\mathbf{K}$  and  $\mathbf{B}$  from the zero-torque trajectory;  $\mathbf{Q}_K$  and  $\mathbf{Q}_B \in \mathbb{R}^{4 \times 4}$  are positive symmetric weighting matrices, which affect optimization speed through dimension [31];  $D$  denotes the duration time;  $J_p$  is the task error function;  $\mathbf{I}(\mathbf{q})$  is the upper-limb model inertial matrix;  $\boldsymbol{\varepsilon}(t)$  is the time-varying tracking error given by,

$$\boldsymbol{\varepsilon}(t) = \mathbf{e}(t) + \beta \dot{\mathbf{e}}(t), \quad \mathbf{e}(t) = \mathbf{q}_d(t) - \mathbf{q}(t) \quad (8)$$

$$\mathbf{K}(t) = \mathbf{F}(t) \mathbf{e}^T(t), \quad \mathbf{B}(t) = \mathbf{F}(t) \dot{\mathbf{e}}^T(t), \quad \mathbf{F}(t) = \frac{\boldsymbol{\varepsilon}(t)}{\gamma(t)}, \quad (9)$$

$$\gamma(t) = \frac{a}{1 + C \|\boldsymbol{\varepsilon}(t)\|^2}, \quad (10)$$

**Table 2** Experimentally-fitted whip model dimension [5, 13]

Whip parameters					
Description	Number	Length	Mass	Spring	Damping
Notation	$N$	$l$	$m$	$k$	$b$
Units	$[-]$	$[m]$	$[kg]$	$[NN \cdot m / rad]$	$[N \cdot mN \cdot s / rad]$
Values	12	0.072	0.012	0.242	0.092

**Table 3** Arm model dimension

Arm parameters				
Description	Mass of upper arm	Mass of lower arm	Length of upper arm	Length of lower arm
Notation	$M_0$	$M_1$	$L_0$	$L_1$
Units	$[kg]$	$[kg]$	$[m]$	$[m]$
Values	0.25	0.274	0.31	0.34

where  $\mathbf{e}(t)$  and its derivative calculate the joints position and velocity error separately comparing with the desired ones;  $\beta$  is 0.6; (Equation (9)) is used for the real  $\mathbf{K}$ ,  $\mathbf{B}$ ; (Equation (10)) shows the adaptation scale factor;  $\alpha$  is 0.2, and  $C$  is 5; The mathematical derivation and other details can be seen in [31].

a policy for the whip targeting task, the pseudocode of LEMON is listed in Algorithm. 1.

### 2.3 Learning/Optimization

Four RL algorithms and OIAC are integrated into the proposed LEMON control framework, respectively. To learn

**Algorithm 1** LEMON algorithm for whip targeting

---

```

1: Input: target position  $(x, y)$ ; stay time  $t_{\text{stay}}$ ; train episodes  $TE$ ; sample episodes  $SE$ ; time steps  $TS$ 
2: Initialize the policy parameters  $\pi_\theta$ ; Environment  $E$ ; Replay buffer  $R = \{\}$ 
3: Set episode counter  $ep$  to 1
4: for  $ep \in \{1, 2, \dots, TE\}$  do
5:   Set step counter  $ts$  to 1
6:   Observe initial state  $\mathbf{s}_{ts}$ :  $[X_{\text{tip}}, Y_{\text{tip}}, V_x, V_y]$  from  $E$  and output  $\mathbf{a}_{ts} \sim \pi_\theta$ :  $[q_{\text{si}}, q_{\text{ei}}, q_{\text{sf}}, q_{\text{ef}}, D]$ 
7:   while  $ep \leq SE$  do
8:     for  $ts \in \{1, 2, \dots, TS\}$  do
9:       Set time counter  $t$  to 0 and initialize  $L = \{\}$ 
10:      while  $t \leq D + t_{\text{stay}}$  do
11:        Use OIAC to update  $\mathbf{K}$ ,  $\mathbf{B}$ , and  $\boldsymbol{\tau}$ 
12:        Compute and store distance  $l_t$ 
13:      end while
14:      Observe  $\mathbf{s}_{ts+1} = \min L \{l_1 \dots l_t\}$ , calculate reward  $r_{ts}$  and done signal  $d_{ts}$ 
15:      Store the transition tuple  $(\mathbf{s}_{ts}, \mathbf{a}_{ts}, r_{ts}, d_{ts}, \mathbf{s}_{ts+1})$  in  $\mathbf{R}$ 
16:    end for
17:    Save average reward  $\bar{R} = \frac{1}{TS} \sum_{ts=1}^{TS} r_{ts}$ 
18:     $ep \leftarrow ep + 1$ 
19:  end while
20:  for  $ts \in \{1, 2, \dots, TS\}$  do
21:    Sample a batch tuple data from  $\mathbf{R}$ 
22:    Calculate the loss  $Loss$  and update policy  $\pi_\theta$ 
23:    Select optimized  $\mathbf{a}_{ts}$  to execute and store the new transition tuple in  $\mathbf{R}$ 
24:  end for
25:  Save average reward  $\bar{R}$ 
26: end for

```

---

### 2.3.1 Proximal Policy Optimization (PPO) [20]

PPO is an algorithm for learning optimal control policies in environments with discrete and continuous action spaces. A trust region optimization algorithm is used to improve the stability and reliability of policy gradient methods in the actor-critic method. The current state  $s_t$  is fed into two same structure networks, actor and critic. Both feedforward networks have hidden layers of size 64 and 64. The learning rate is set to  $1e-4$ . The actor produces the decision Gaussian distribution  $\pi(s_t|\theta)$ , which the action  $a$  is sampled. The critic assesses the output  $V(s)$  corresponding to the  $s_t$ . The objective function of PPO is the expected return of the policy, with a term that encourages the new policy to be close to the old policy (Eq. (11)):

$$J_{\text{PPO}}(\theta) = \sum_{t=1}^T \left( \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\text{old}}(a_t | s_t)} \hat{A}_t - \beta \text{KL}[\pi_{\text{old}} | \pi_{\theta}] \right), \quad (11)$$

where  $\hat{A}_t$  is the advantage function;  $\text{KL}[\pi_{\text{old}} | \pi_{\theta}]$  is the KL-divergence constraint;  $\beta$  dynamically changes with the adaptive KL penalty.

### 2.3.2 Deep Deterministic Policy Gradient (DDPG) [21]

To some extent, it could be seen as an extension of the Deep Q Network (DQN), since it follows similar strategies of Q-learning. There are two kinds of neural networks, i.e., an actor that maps the states to actions and a critic that estimates the value of taking a specific action in a specific state. The Q-network is used to compute the gradient of the policy expected return with respect to the policy parameters, and the actor and critic networks are both updated by using the transitions stored in the replay buffer. Each network is a two-layer feedforward neural network with 256 hidden nodes. Rectified Linear Units (ReLU) [32] activation functions are employed between each layer for both the actor and critic networks. Additionally, a final *tanh* unit is applied following the actor network output. The critic network takes both the state and action as input for its first layer. The learning rate for these networks is set to  $3e-4$ . The objective function of DDPG is given by,

$$J_{\text{DDPG}}(\theta) = \sum_{t=1}^T (Q_{\phi}(s_t, a_t) - \ln \pi_{\theta}(a_t | s_t)) \quad (12)$$

### 2.3.3 Twin Delayed Deep Deterministic Policy Gradients (TD3) [22]

TD3 is an extension of DDPG that applies clipped double Q-learning, delayed policy updates, and target policy

smoothing, and the target Q-value for the Q-network update is the minimum of the two Q-network predictions.

There are two sets of critic networks and an actor network adopted in TD3. The state  $s_t$  is the input to the actor network, which is formed by three-layer feedforward neural network with (9, 256, 256) hidden nodes and ReLU activation functions. The input of critic network is a state-action pair to compute a target value  $Q$  and through two Q-network and reduce the risk of deviation, shown in (Eq. (13)). A soft updating factor  $\tau$  is imported TD3 to update network parameters, thus the delayed method brings a low variance action updates (Eq. (14)). Moreover, a Gaussian noise  $\epsilon$  is added to the target policy to improve TD3 capacity of exploration (see Eq. (15)). Therefore, the stability of the DDPG is facilitated by this network structure, and TD3 becomes more robust and more reliable in learning than DDPG in some tasks [33]. It also uses fixed target networks for the actor and the critic, which helps reduce the correlation between the target and the prediction, making the training more stable and consistent. TD3 also introduces a delay between the update of the actor and the critic networks to further improve the stability and robustness of the algorithm.

$$Q_{\text{target}} = r + \gamma \min_{i=1,2} Q_{\phi_i}(s_t, a_t), \quad (13)$$

$$\phi_{i'} \leftarrow \tau \phi_i + (1 - \tau) \phi_{i'}, \quad (14)$$

$$\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c) \quad (15)$$

### 2.3.4 Soft Actor-Critic (SAC) [23]

SAC is an off-policy algorithm, which is used to optimize the expected cumulative reward of a stochastic policy. The hidden layers neuros and learning rate are set the same as DDPG and TD3. Different from these two off-policies, it includes an entropy regularization term in the objective function. The term will balance exploration and exploitation, making it suitable for tasks with unknown dynamics, so it encourages exploration and stability in the learning process. The objective function of SAC is given by,

$$J_{\text{SAC}}(\theta) = \sum_{t=1}^T (r(s_t, a_t) + \alpha H(\pi(\cdot | \theta_t))), \quad (16)$$

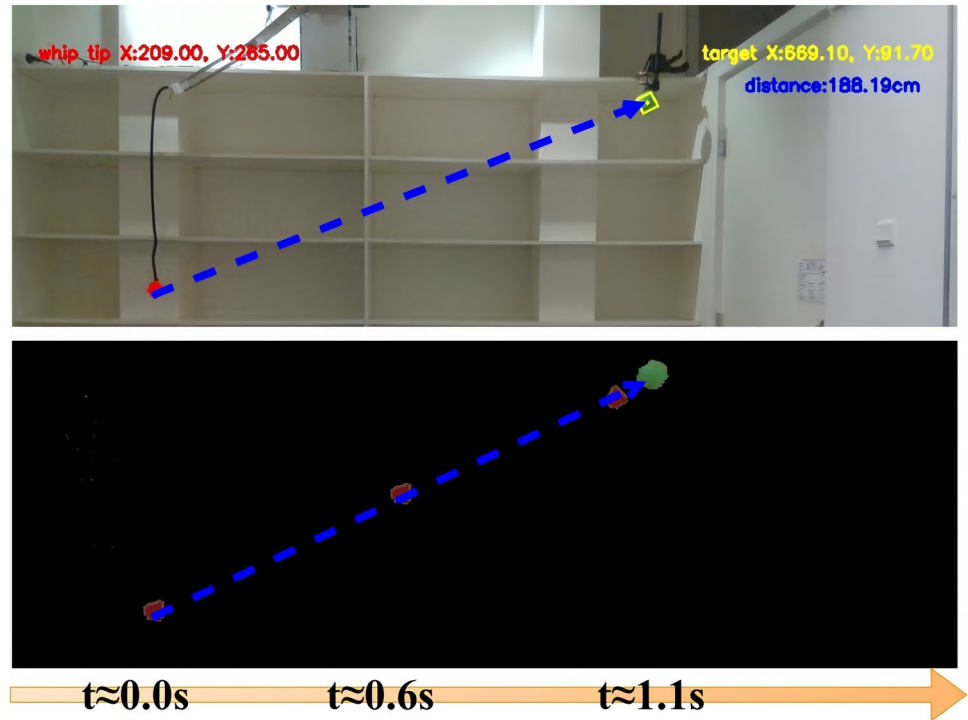
where  $H(\pi(\cdot | s_t))$  reflects how random the policy in state  $s$ ;  $\alpha$  is regularization coefficient to control the entropy, it is 0.2 in the whip task;

### 2.3.5 Reward Design

The reward function decides how to evaluate the action value, it affects the policy formulation and entire training



**Fig. 3** Vision tracking of the whip tip and targets over time  $t$



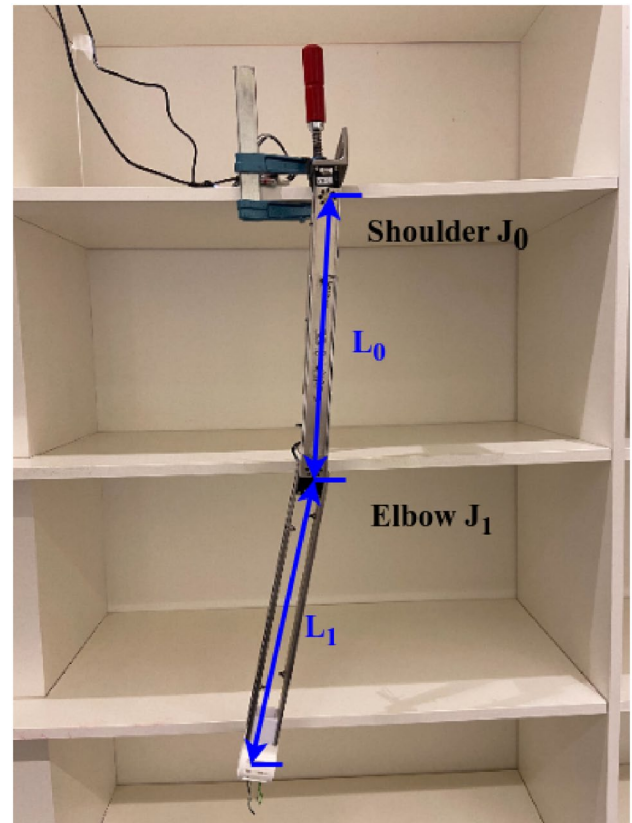
results. Here the distance is utilized in the reward function given by,

$$r(s_t, a_t) = \begin{cases} r = -10 * d & \text{if } 0.1 < d \\ r = -5 * d & \text{if } 0.06 < d \leq 0.1 \\ r = 50 & \text{if } d \leq 0.06, \end{cases} \quad (17)$$

where if the distance is smaller than  $0.06m$ , it would be considered as the tip hits the target successfully in the simulated and real environment;

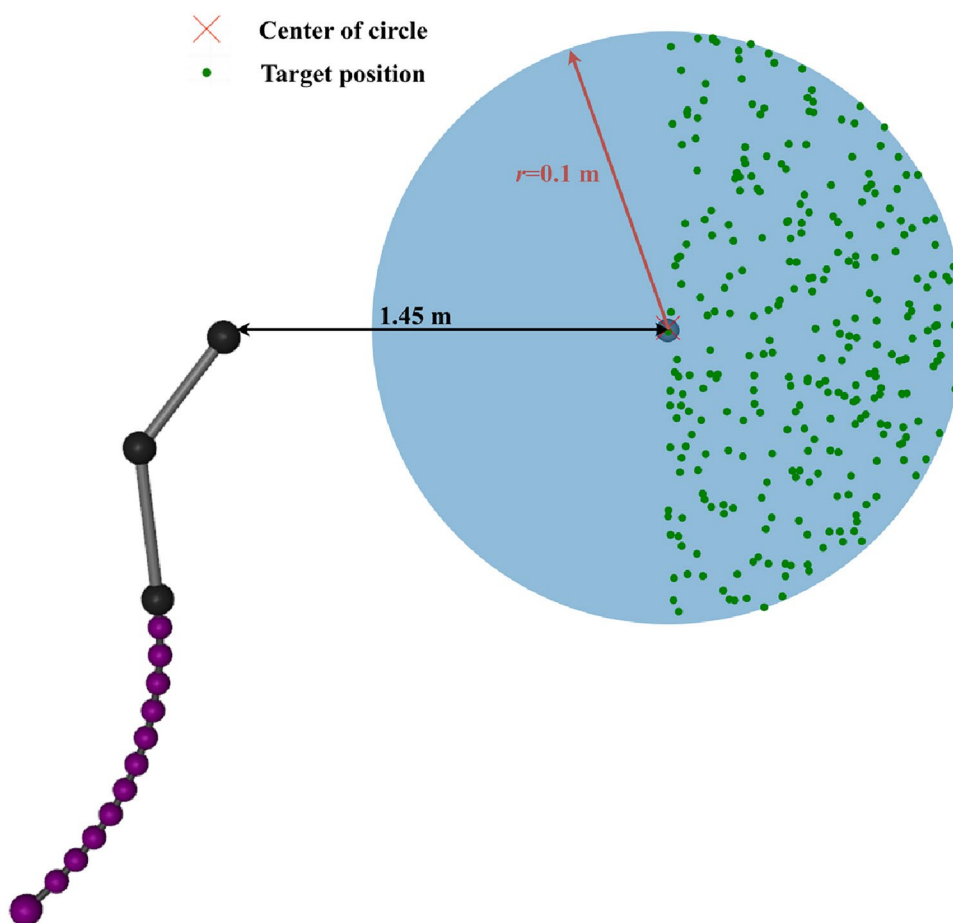
### 2.3.6 Locally-based Divide Rectangle (DIRECT-L)

A DIRECT-L algorithm was applied to control simulated whip targeting control [13]. It weights more toward the local search than the global and samples the objective function points at  $\Omega$  (Eq. (18)) to start its search. To simplify the analysis and reduce the algorithm running time, it is usually normalized as  $[0, 1]$ , and also considered as a bound-constrained optimization problem. It is used for sampling the new centers of the new hypercube. The DIRECT-L selects potential hypercube in  $S$ . Normally, the optimal potential hypercube exists either in the local search low function values at the centers or in the large enough to be good targets for global search. Thus, not all such hypercube is divided, selecting the smallest and biggest value of  $f$  to subdivide the hypercube into subgroups.



**Fig. 4** The real two-joint robot arm consisting of the joints  $J_{0,1}$  for 2D movement control. Detailed parameters of its weight and length refer to a real human arm model [37], which can be checked in Table 3

Fig. 5 Targets distribution zone



$$\Omega = \{x \in \mathbb{R}^N : lb \leq x \leq hb\} \sim \Omega = \{x \in \mathbb{R}^N : 0 \leq x \leq 1\}, \quad (18)$$

$$\min_{x \in \Omega} f(x), \quad f : \mathbb{R}^N \rightarrow \mathbb{R}, \quad (19)$$

where  $f$  is Lipschitz continuous on  $\Omega$ ;  $N$  is the number of dimension;  $lb$  is the low boundary of the action;  $hb$  is the high boundary of the action.

### 2.3.7 Genetic Algorithm (GA)[34]

Since the objective function is hard to express with the action space elements, another powerful optimization method GA is raised to solve the black box problem. A complete GA includes three main operators such as selection, crossover, and mutation. Each solution is represented as a chromosome and every parameter corresponds to a gene. Fitness ( $f$ ) is the GA evaluation standard. Generally, the value of  $f$  is higher, the individual is fitter for the environment, so the objective minimum distance is transferred as,

$$f = \frac{D_{\max} - d_{\text{real}}}{D_{\max}}, \quad (20)$$

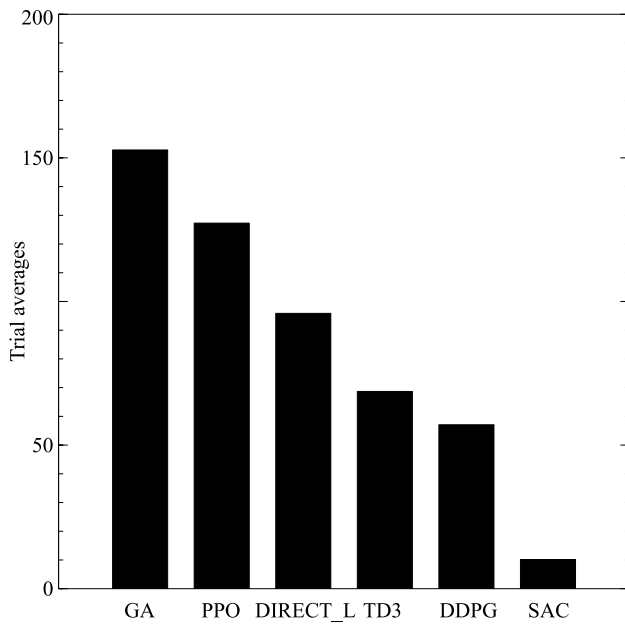
where  $D_{\max}$  means the maximum distance value from the tip to the target;  $d_{\text{real}}$  means the real distance value from the tip to the target.

## 2.4 Visual Tracking

The Intel camera D435<sup>1</sup> was used for the vision tracking mission, located 3.5 meters in front of the robot arm and targets. It provides the distance between the whip tip and a target as feedback to learning/optimization algorithms. Mean-shift is a popular method for object tracking in computer vision because it does not require any prior information about the object's shape or motion [35]. Therefore, the method was selected for real-time tracking deformable and fast whip movements. After the pre-processing of the Gaussian filtering, the target and whip tip were segmented by the different Hue, Saturation, and Value (HSV) thresholds in frames, respectively. Figure 3 shows how to track the real-time distance between the tip and the target.

<sup>1</sup> [https://www.intelrealsense.com/depth-camera-d435/?\\_ga=2.86552923.293598258.1670164775-171451401.1668173604](https://www.intelrealsense.com/depth-camera-d435/?_ga=2.86552923.293598258.1670164775-171451401.1668173604).





**Fig. 6** Trial averages for hitting 300 random targets (see Eq. 21) by six learning or optimization algorithms

## 3 Results

The length of the real robot arm is the same as the simulated one (see Fig. 4). Each joint of the upper limb model is driven by a Dynamixel motor<sup>2</sup> [36]. This simple actuation design has no external spring, damper, and gear. It follows an actuation trend in robotic movement control, i.e., lightweight and reduced inertia. Its resulting control challenges (e.g., adaptivity) will be addressed by our proposed LEMON framework (see Fig. 1). The experiments were run on Ubuntu 20.04 laptop with graphic card RTX3060, and each RL learning or optimization algorithm took seven training hours on average. The experimental video can be seen at the footnote.<sup>3</sup>

### 3.1 Comparisons between Learning/optimization Methods

#### 3.1.1 Task Definition

The task is to learn or optimize the planned movement parameters (see Eq. (1)) of the proposed LEMON control framework by one of the six algorithms. They aim at minimizing the distance  $L$  between a whip tip and a target. 300 random targets were generated to compare the generalization performance of the six algorithms in simulated whip targeting. The targets were situated in a circle with

a radius of 0.1  $m$  (see green dots in Fig. 5). Note that the hit threshold value  $L^*$  is set to 0.06  $m$ , i.e., the target is considered to be hit if  $L \leq L^*$ .

#### 3.1.2 Performance

The generalization performance of a learning or optimization algorithm is quantified by its average required trials of successfully hitting 300 targets,

$$\overline{N}_{\text{hit}} = \frac{1}{300} \sum_{i=1}^{300} n_{\text{hit}}, \quad (21)$$

where  $\overline{N}_{\text{hit}}$  means the average required trials;  $n_{\text{hit}}$  denotes the required trials of successfully hitting a target. A fewer  $\overline{N}_{\text{hit}}$  refers to a better learning generalization.

DRL algorithms excluding the PPO, perform better than the nonlinear optimization DIRECT-L and a genetic algorithm, since the on-policy strategy adopted in the PPO is low-efficient in exploration and exploitation. Besides, we can see that the SAC outperforms the other five counterparts in learning generalization (see Fig. 6). This is because entropy regularization is introduced to boost its exploring capacity in the SAC, therefore leading to better learning generalization with fewer average hits.

Figure 7 shows the average rewards of four DRL algorithms over 500 episodes. An episode reward was the average of 100 (trial) accumulated rewards. The first 50 episodes were used for allocating random data into the replay buffer. We can see that the SAC outperforms the three counterparts owing to its better exploration ability.

### 3.2 Comparison between the OIAC and Constant Impedance Controller

The comparison indicates the benefits of replacing constant impedance control with the OIAC. The stiffness matrix  $\mathbf{K}$  and damping matrix  $\mathbf{B}$  used for the constant impedance controller are given by [13]:

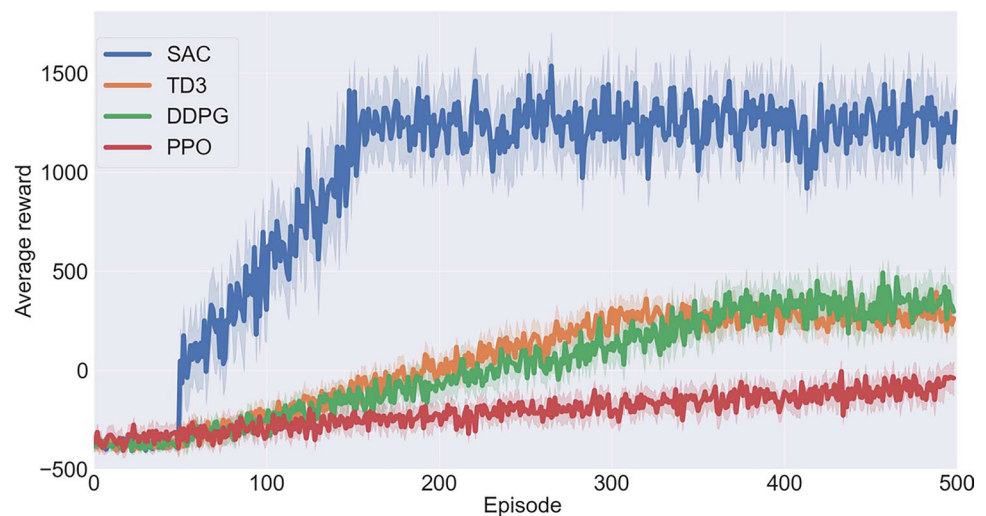
$$\mathbf{K} = \begin{bmatrix} 29.5 & 14.3 \\ 14.3 & 39.3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2.95 & 1.43 \\ 1.43 & 3.93 \end{bmatrix}. \quad (22)$$

We can see that the position tracking error governed by the OIAC is reduced to 22.36% of constant impedance control (see Fig. 8 and Table 4). Besides, the velocity tracking error guided by the OIAC is decreased to 13.29% of constant impedance control. Therefore, the integration of the SAC and OIAC outputs others in better learning generalization and higher tracking control accuracy.

<sup>2</sup> XM430-W350, <http://emanual.robotis.com/docs/en/dxl/x/xm430-w350/>.

<sup>3</sup> [www.youtube.com/watch?v=N4GWXABSno](http://www.youtube.com/watch?v=N4GWXABSno).

**Fig. 7** Average rewards of four DRL algorithms over 500 episode training



### 3.3 Sim2real

We can see that the whip trajectories in the simulated and real environments are 89.09% similar (see Fig. 9), which is quantified by the average similarity parameter  $\rho_{\text{similar}}$ ,

$$\rho_{\text{similar}} = \frac{1}{n} \sum_{k=1}^n \sum_{i=1}^m \left( 1 - \sqrt{\frac{(P_k^i - S_k^i)^2}{(P_k^i)^2}} \right), \quad (23)$$

where  $n$  denotes the number of simulated and physical whip tip trajectories, i.e.,  $n=18$  (trials), each of which has  $m$  data points;  $\sqrt{(P_k^i - S_k^i)^2}$  is the Euclidean distance between  $P_k^i$  of a physical whip tip trajectory and  $S_k^i$  of the related simulated one. Learned trajectories of hitting three different targets in simulated and physical whip targeting are 89.09% similar over iterations (see Fig. 9), respectively. The high similarity indicates that it is practicable to achieve sim2real in this task though some slight gaps between the simulated and real environments.

Thus, it is feasible to achieve sim2real because the whip movements are very similar in both environments. Figure 10 (b-d) shows the distance  $L$  between a whip tip and a target, stiffness, and damping values in a whip targeting task. The amplitudes of the shoulder joint stiffness and

damping values are higher than those of the elbow joint, as the shoulder joint is a leading joint in the control task.

## 4 Discussion

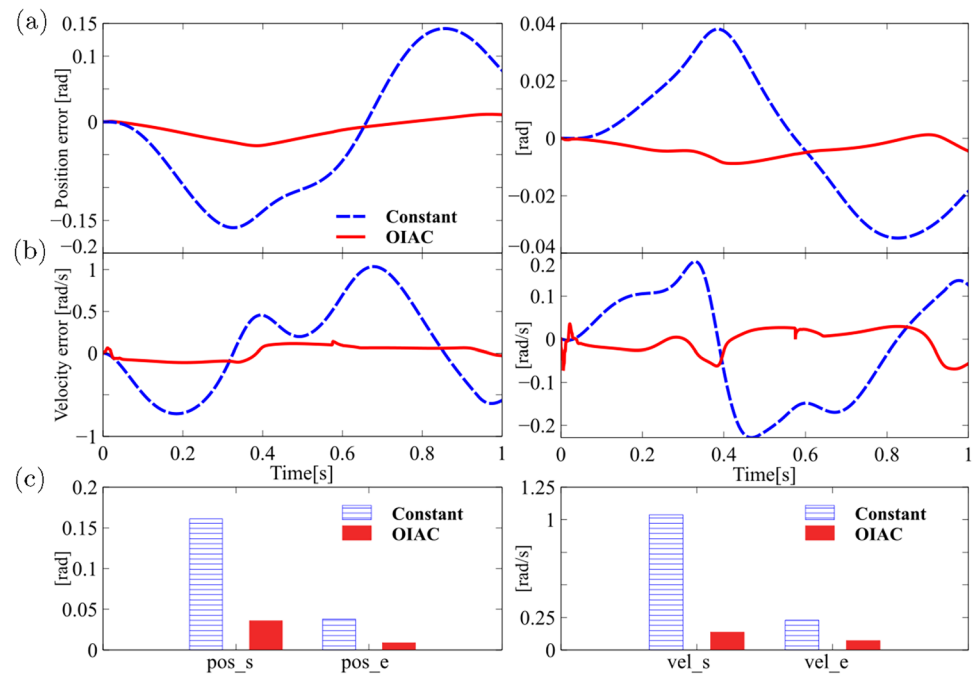
### 4.1 Algorithmic Comparison

GA and DIRECT-L optimization methods are worse than most DRL methods in learning generalization of whip targeting control. Required iterations by the GA depend heavily on the quality of initial population formulations. The new population is composed of operations such as replication, crossover, and mutation of the previous generation of individuals. There is no clear direction for optimization in the whip task, so the recursive iterations are more than other algorithms. For the DIRECT-L, the optimization process presents regularity, as the optimal solution is either a smallest or largest hypercube. Thus, it has a specific optimization direction for the fixed target and will change to the opposite one when the result is always divergent. However, the dividing hypercube method is usually better at the low-dimension discrete space. As the spatial dimension increases, the optimization becomes hard. Therefore, its performance is worse than off-policy DRL in continuous control tasks, e.g., whip targeting control. The PPO, as an on-policy DRL, is the same as an updated policy. Therefore, the data used in a previous policy will be replaced. It is a good algorithm to solve the continuous action problem, but not fit for the whipping task in this project. If a target is not hit in one single trial, a robot will try a new policy to generate new datasets. The historical data from previous policies is not kept, thus its optimization effect is worse than off-policy DRL. Off-policy DRL algorithms refer to the previous data experience saved by a replay buffer. The performance of DDPG and TD3 is similar in whip targeting control, since

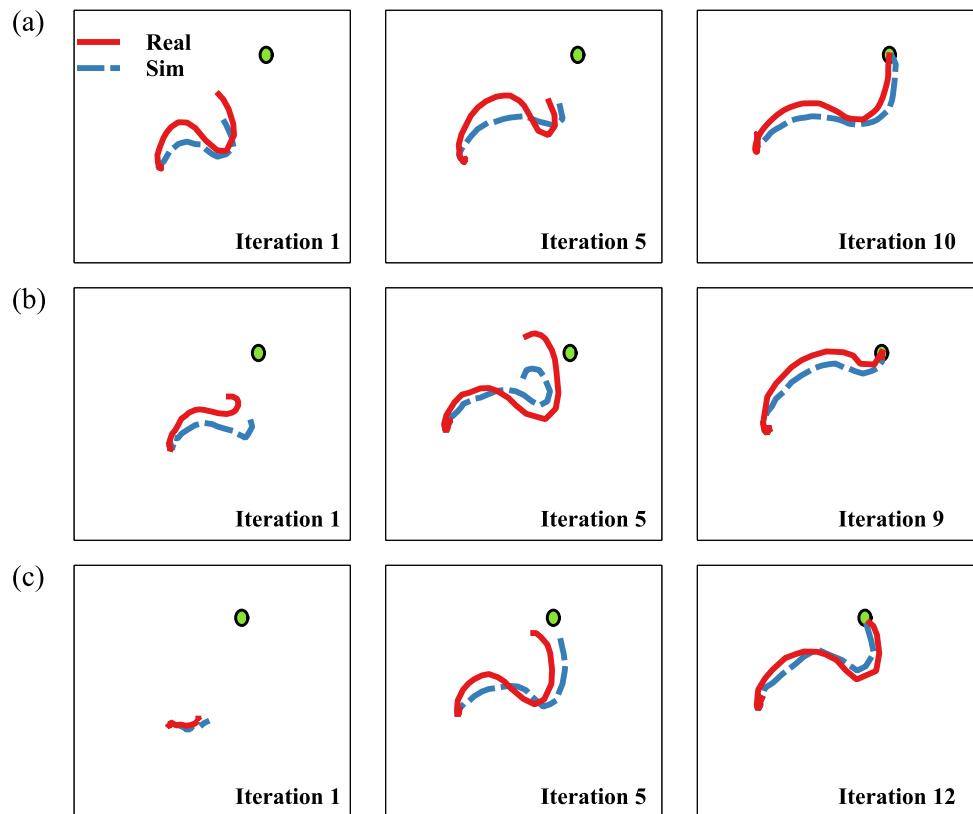
**Table 4** Performance of different impedance controllers with SAC

		SAC and constant	SAC and OIAC
Position error [rad]	Shoulder	0.161	<b>0.036</b>
	Elbow	0.038	<b>0.009</b>
Velocity error [rad/s]	Shoulder	1.038	<b>0.138</b>
	Elbow	0.229	<b>0.073</b>

**Fig. 8** Errors of the position and velocity in the shoulder and elbow joint; (a) left: shoulder joint position errors; right: elbow joint position errors; (b) left: shoulder joint velocity errors; right: elbow joint velocity errors; (c) compares the absolute maximum errors in two joints; pos\_s, vel\_s, pos\_e, and vel\_e are the abbreviations of position and velocity for shoulder, and elbow, respectively



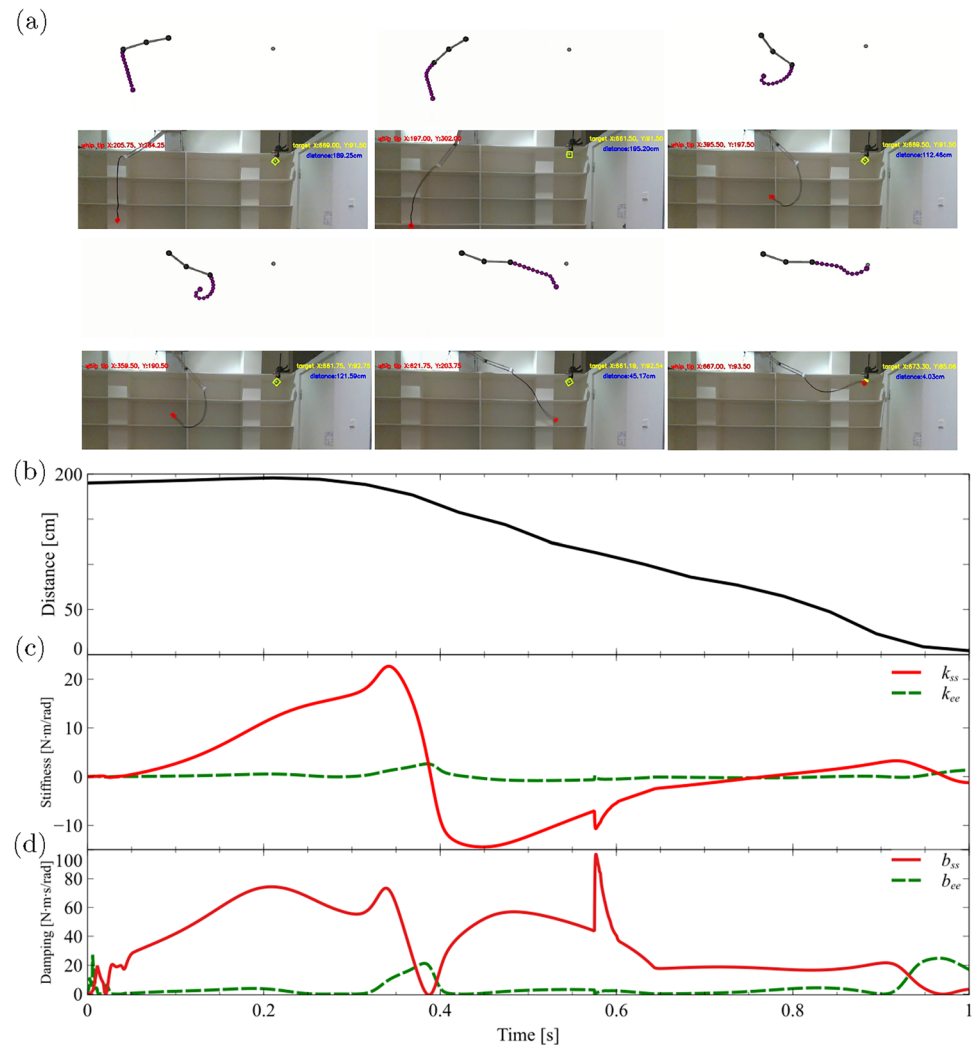
**Fig. 9** Simulated and physical whip tip trajectories over learning iterations; (a–c) refer to targets 1–3 (green dots)



both of them are based on a determined policy. Besides, the TD3 is an extension of the DDPG. Different from the DDPG and TD3, a central feature of the SAC is entropy regularization that prevents the policy from converging to a local optimum. Therefore, the exploration of the SAC is more

efficient than the DDPG and TD3. A Q-function and policy are used to estimate the action-value and determine the best actions, which allow for learning a policy and a value function in the SAC. Its output action is within the range of the  $\tanh$  formula, while the TD3 output action refers to the  $\tanh$

**Fig. 10** Sim2real: (a) 12 snapshots of a simulated whip targeting and a physical whip targeting; (b) distance  $L$  between a whip tip and a target; (c, d) stiffness  $K(t)$  and damping  $B(t)$ , respectively (parameters of the OIAC)



formula with noise. The actor-critic network will be more robust with the noisy formula, but its generated actions will be close to the boundary. The TD3 is a good strategy for other tasks, but not for the presented whip targeting. This is because the actions which can hit targets are usually not around the boundary. Based on the above reasons, SAC is more suitable to learn whip targeting control tasks.

## 4.2 Limitations

The proposed learning-based control framework is a prototype with several limitations that illustrate directions for future work. First, it was validated on joint space control of a two-DoF robot. Further work is required for end-effector space of a UR5 robot arm with the combination of reinforcement learning and online adaptive admittance/impedance control [15, 31, 38]. This will reduce control dimension to three, compared to six of UR5 joint space. Second, it assumes that the targets are fixed. An advanced vision algorithm can be developed to predict moving target trajectories, which is integrated into the presented control

framework. Third, its application is limited to flexible tool targeting control, which can be extended to industrial scenarios, e.g., pick and throw objects into boxes. Finally, its motivation is to explore the benefits of integrating data-driven and physics-based algorithms into fast and accurate robot manipulation with flexible tools. In future, experimental data of human whip targeting will be recorded to analyze and compare differences between human and machine intelligence. This may lead to next-generation robot arm manipulation and novel understandings on fast and accurate human motor control.

## 5 Conclusion

The upper limb model is used to reverse-engineer the fast movement control of a deformable object, e.g., whip targeting. The whole process includes an impedance controller to ensure the stability of the movement. When the coordinates of targets are fixed, the tracking error of OIAC is better than that of the constant impedance controller. At the same time,

a variety of optimization methods were tested to compare learning strategies for hitting different targets, where the model has to provide quickly adjustments. Finally, SAC and OIAC showed better effects and is a more suitable optimization method for this paper. Human muscle properties will be analyzed and understood by reproducing whip targeting in a 7-DoF robot with a muscle-like controller in future [38].

**Acknowledgements** This work was supported in part by the Brødrene Hartmanns (No. A36775), Thomas B. Thriges (No. 7648-2106), Fabrikant Mads Clausens (No. 2023-0210) and EnergiFyn funds.

**Funding** Open access funding provided by University of Southern Denmark.

**Availability of data and materials** The datasets generated and analyzed during the current study are not publicly available, because they are the parts of an ongoing study. However, they can be available from the corresponding author on a reasonable request.

## Declarations

**Conflict of interest** On behalf of all authors, the corresponding author states that there is no Conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Ryerson, W. G. (2020). Ontogeny of strike performance in ball pythons (*python regius*): A three-year longitudinal study. *Zoology*, 140, 125780.
2. Hickman, G. C. (1979). The mammalian tail: A review of functions. *Mammal Review*, 9(4), 143–157.
3. Matherne, M. E., Cockerill, K., Zhou, Y., Bellamkonda, M., & Hu, D. L. (2018). Mammals repel mosquitoes with their tails. *Journal of Experimental Biology*, 221(20), 178905.
4. Wolff, J. O., Wiegmann, C., Wirkner, C. S., Koehnsen, A., & Gorb, S. N. (2019). Traction reinforcement in prehensile feet of harvestmen (arachnida, opiliones). *Journal of Experimental Biology*, 222(3), 192187.
5. Krotov, A., Russo, M., Nah, M., Hogan, N., & Sternad, D. (2022). Motor control beyond reach-how humans hit a target with a whip. *Royal Society Open Science*, 9(10), 220581. <https://doi.org/10.1098/rsos.220581>
6. Hietala, J., Blanco-Mulero, D., Alcan, G., Kyrki, V. (2022). Learning visual feedback control for dynamic cloth folding. In *IEEE/RSJ international conference on intelligent robots and systems (iros)*, Kyoto, Japan. 1455–1462.
7. McConachie, D., Dobson, A., Ruan, M., & Berenson, D. (2020). Manipulating deformable objects by interleaving prediction, planning, and control. *The International Journal of Robotics Research*, 39(8), 957–982.
8. Khalil, F., & Payeur, P. (2010). Dexterous robotic manipulation of deformable objects with multi-sensory feedback—a review. *Robot Manipulators Trends and Development*. <https://doi.org/10.5772/9183>
9. Sanchez, J., Corrales, J.-A., Bouzgarrou, B.-C., & Mezouar, Y. (2018). Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey. *The International Journal of Robotics Research*, 37(7), 688–716.
10. Zhang, H., Ichnowski, J., Seita, D., Wang, J., Huang, H., & Goldberg, K. (2021). Robots of the lost arc: Self-supervised learning to dynamically manipulate fixed-endpoint cables. In *IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, 4560–4567.
11. Chi, C., Burchfiel, B., Cousineau, E., Feng, S., & Song, S. (2024). Iterative residual policy: For goal-conditioned dynamic manipulation of deformable objects. *The International Journal of Robotics Research*, 43(4), 389–404. <https://doi.org/10.1177/02783649231201201>
12. Nah, M. C., Krotov, A., Russo, M., Sternad, D., & Hogan, N. (2023). Learning to manipulate a whip with simple primitive actions—a simulation study. *iScience*, 26(8), 107395. <https://doi.org/10.1016/j.isci.2023.107395>
13. Nah, M. C., Krotov, A., Russo, M., Sternad, D., Hogan, N. (2020). Dynamic primitives facilitate manipulating a whip. In *8th IEEE Ras/embs International Conference for Biomedical Robotics and Biomechatronics (biorob)*, New York, USA, 685–691.
14. Zimmermann, S., Poranne, R., & Coros, S. (2021). Dynamic manipulation of deformable objects with implicit integration. *IEEE Robotics and Automation Letters*, 6(2), 4209–4216.
15. Lin, X., Wang, Y., Olkin, J., & Held, D. (2021). Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In: *Proceedings of the 2020 conference on robot learning*. 155. Boston, MA, USA: PMLR, 16–18 Nov, 432–448. <https://proceedings.mlr.press/v155/lin21a.html>.
16. Chang, P., & Padif, T. (2020). Sim2real2sim: Bridging the gap between simulation and real-world in flexible object manipulation. In *Fourth IEEE International Conference on Robotic Computing (IRC)*, Taichung, China, 62. <https://doi.org/10.1109/IRC.2020.00015>
17. Yin, H., Varava, A., & Kragic, D. (2021). Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, 6(54), 8803.
18. Zhu, J., Cherubini, A., Dune, C., Navarro-Alarcon, D., Alambeigi, F., Berenson, D., Ficuciello, F., Harada, K., Kober, J., Li, X., Pan, J., Yuan, W., & Gienger, M. (2022). Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics Automation Magazine*, 29(3), 67–77. <https://doi.org/10.1109/MRA.2022.3147415>
19. Miao, Q., Lv, Y., Huang, M., Wang, X., & Wang, F. Y. (2023). Parallel learning: Overview and perspective for computational learning across syn2real and sim2real. *IEEE/CAA Journal of Automatica Sinica*, 10(3), 603–631. <https://doi.org/10.1109/JAS.2023.123375>
20. Gu, Y., Cheng, Y., Chen, C. L. P., & Wang, X. (2022). Proximal policy optimization with policy feedback. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(7), 4600–4610. <https://doi.org/10.1109/TSMC.2021.3098451>
21. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. In: *4th international conference on learning representations*. San Juan, Puerto Rico, [arXiv:1509.02971](https://arxiv.org/abs/1509.02971)



22. Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In: International conference on machine learning. PMLR. Stockholm, Sweden, 1587–1596.
23. Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. PMLR. Stockholm, Sweden, 1861–1870.
24. Gablonsky, J. M., & Kelley, C. T. (2001). A locally-biased form of the direct algorithm. *Journal of Global Optimization*, 21(1), 27–37.
25. Sivanandam, S., Deepa, S., Sivanandam, S., & Deepa, S. (2008). Genetic algorithm optimization problems. *Introduction to Genetic Algorithms*, 165–209.
26. Erez, T., Tassa, Y., Todorov, E. (2015). Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, USA, 4397–4404. <https://doi.org/10.1109/ICRA.2015.7139807>
27. Choi, H., Crump, C., Duriez, C., Elmquist, A., Hager, G., Han, D., Hearl, F., Hodgins, J., Jain, A., Leve, F., Li, C., Meier, F., Negrut, D., Righetti, L., Rodriguez, A., Tan, J., & Trinkle, J. (2021). On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences*, 118(1), e1907856118. <https://doi.org/10.1073/pnas.1907856118>
28. Centurelli, A., Arleo, L., Rizzo, A., Tolu, S., Laschi, C., & Falotico, E. (2022). Closed-loop dynamic control of a soft manipulator using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 7(2), 4741–4748.
29. Nah, M. C., Krotov, A., Russo, M., Sternad, D., Hogan, N. (2021). Manipulating a whip in 3d via dynamic primitives. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2803–2808). Prague. <https://doi.org/10.1109/IROS51168.2021.9636257>
30. Xiong, X., Nah, M. C., Krotov, A., Sternad, D. (2021). Online impedance adaptation facilitates manipulating a whip. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 9297–9302). Prague.
31. Xiong, X., Manoonpong, P. (2018). Adaptive motor control for human-like spatiotemporal adaptation. In *IEEE International Conference on Robotics and Biomimetics (robio)* (pp. 2107–2112), Kuala Lumpur.
32. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84–90. <https://doi.org/10.1145/3065386>
33. Zhou, J., Xue, S., Xue, Y., Liao, Y., Liu, J., & Zhao, W. (2021). A novel energy management strategy of hybrid electric vehicle via an improved td3 deep reinforcement learning. *Energy*, 224, 120118.
34. Holland, J. H. (1992). Genetic algorithms. *Scientific American*, 267(1), 66–73.
35. Cariou, C., Le Moan, S., & Chehdi, K. (2022). A novel mean-shift algorithm for data clustering. *IEEE Access*, 10, 14575–14585.
36. Xiong, X., & Poramate, M. (2021). Online sensorimotor learning and adaptation for inverse dynamics control. *Neural Networks*, 143, 525–536. <https://doi.org/10.1016/j.neunet.2021.06.029>
37. Burdet, E., Tee, K. P., Mareels, I., Milner, T. E., Chew, C. M., Franklin, D. W., Osu, R., & Kawato, M. (2006). Stability and motor adaptation in human arm movements. *Biological Cybernetics*, 94, 20–32.
38. Xiong, X., Wörgötter, F., & Manoonpong, P. (2013). A simplified variable admittance controller based on a virtual agonist-antagonist mechanism for robot joint control. *Nature-Inspired Mobile Robotics*, 10(1142/9789814525534), 0037.

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.