SofToss

Learning to Throw Objects With a Soft Robot

By Diego Bianchi, Michele Gabrio Antonelli, Cecilia Laschi, Angelo Maria Sabatini, and Egidio Falotico

In this article, we present, for the first time, a soft robot control system (SofToss) capable of throwing life-size objects toward target positions. SofToss is an open-loop controller based on deep reinforcement learning (RL) that generates, given the target position, an actuation pattern for the tossing task. To deal with the high nonlinearity of the dynamics of soft robots, we deploy a neural network to learn the relationship between the actuation pattern and the target landing position, i.e., the direct model (DM) of the task. Then, an RL method is used to predict the actuation pattern given the goal position. The proposed controller was tested on a modular soft robotic arm, I-Support, by tossing four objects of different shapes and weights in 140-mm squared target boxes. We registered a success rate of almost 65% of the throws in two actuation modalities (i.e., partial, keeping one module of the soft arm passive, and complete, with both modules active). This performance raises to 85% if one can choose the number of modules to actuate for each throwing direction. Furthermore, the results show that the proposed learning-based, real-time controller achieves a performance comparable to that of an optimization-based, nonreal-time controller. Our study contributes to the foundations for bringing soft robots into everyday life and industry by performing more complex, dynamic tasks.

Digital Object Identifier 10.1109/MRA.2023.3310865
Date of current version: 29 September 2023

INTRODUCTION

Performing ballistic tasks, such as throwing objects, with a robot is an aspired ability. It increases the working capacity of the robot, for instance, in a weakly structured logistics sorting scene [1], [2], [3]. For a traditional robot, this ability is challenging to obtain for the high acceleration and speed required [4], which brings its joints to their limits. However, in [5] and [6], authors show that storing elastic energy by using compliant actuators helps rigid robots to perform ballistic tasks, such as throwing an item. Soft robots may be better suited for this task, thanks to their characteristics [7]: they can store potential elastic energy within their bodies and then transfer it to the held object before releasing it to perform the throw. This characteristic has been shown in [8]. In that work, the authors show that a soft robotic arm is potentially able to perform several tasks, such as the throwing one, by releasing the held object during a linear trajectory.

Indeed, due to the inherent deformability of the actuators and materials, soft robots are able to adapt to obstacles, terrain, etc. These characteristics make them suitable for various unstructured environments, which opens up new opportunities in various fields, such as search and rescue, human assistance, and applications in medicine [7]. The downside of having these complex soft body structures resides in their challenging control. Such problems affect both the low-level and high-level control. A low actuation frequency, especially in pneumatic-actuated soft robots, represents one of the major issues of low-level control. Actuating the robot at high frequency will generate undesired vibrations in the soft robot. The impossibility of applying the traditional rigid control theory (often based on rigidity assumptions) represents one of the main issues of the high-level controller.

Tossing an object toward a target requires reaching the releasing point in the robot workspace with a precise speed. This is not trivial to achieve with a soft arm since its dynamic model may not be available or, in general, accurate [9], mainly due to the high stochasticity in the fabrication process. Different alternatives have been proposed to overcome this limitation. Machine learning algorithms [10] have been deployed to approximate the model of a soft robot in static and dynamic conditions [11], but these strategies were tailored for a specific task, primarily trajectory tracking. Recently, this task has been performed with some payloads attached to the end effector of the soft robotic arm to assess the robustness of the controller with respect to external interferences [12], [13]. In addition, state-of-the-art soft robot controllers use a combination of machine learning techniques, often neural network based, and optimization methods. This procedure makes the controller unable to work in real time since it needs to wait for the optimization process to end.

Recent approaches to controlling a soft robot involve the application of RL [10]. This method has great potential to come up with a robust decision-making capability but can hardly be applied directly on a real platform since the prolonged use of the soft robotic platform will lead to reliability problems. To mitigate this problem, the RL agent is trained in a simulation environment where the robot is approximated with its model [14] or with a function approximator, i.e., a neural network [12], [15]. However, an RL agent trained in simulated environments often performs poorly when tested on a real-world environment, mainly due to the "sim2real gap," a lack of fidelity and accuracy of the environment in replicating real-world conditions, which are even more challenging with a soft robot compared with a traditional one.

In this work, we present SofToss, a model-free soft robot control system for the tossing task. SofToss consists of an open-loop controller in which an artificial agent is trained, through deep RL, to select the actuation pattern necessary to throw different objects into small boxes located outside of the robot workspace. This work introduces the learning of the control policy as a solution for carrying out complex ballistic tasks beyond the current soft manipulator controllers, which typically enable only tracking tasks.

MATERIALS AND METHODS

METHODOLOGY

Our controller can be classified as a model-free, open-loop approach to accomplishing the ballistic task of throwing objects with soft robots. More specifically, given a desired landing position, the controller learns through reinforcement to give the robot the appropriate actuation patterns to perform the throw toward the target. Figure 1(a) and (b) illustrates the targets and the objects employed, respectively.

Inspired by previous works [5], we generate the throw with a two-phase movement (backswing before the forward motion toward the target). This constraint applied to the movement generation speeds up the controller training phase since the grasped object trajectory is predefined and does not need to be predicted.

We defined that each trajectory lies in one of the planes of the sheaf of planes whose intersecting line is the z-axis passing along the manipulator's longitudinal backbone. The division of the movement into a run-up phase and a forward phase allows us to increase the robot's speed and, consequently, its throwing range. In our experiments, the robot first performs a backswing (run-up) and then a rapid movement toward the target (forward), as shown in Figure 1(c). The object is released when it reaches the lowest point of the trajectory in the forward movement after the backswing.

Our controller consists of three components: 1) an RL artificial agent, 2) a DM of the task, and 3) an actuation network (Figure 2). The RL agent learns to generate the actuation pattern of the forward-phase movement. Given the actuation input pattern, the DM, which is implemented with a feed-forward artificial neural network (ANN) with one hidden layer, predicts the resulting landing position. The actuation network is used to predict the actuation pattern for backward movement, given the one for the forward movement generated by the RL agent. It is implemented with a feed-forward ANN with one hidden layer. The possibility of defining these task characteristics and the reliance on a forward model derived from the robot data led us to a solution that speeds up the controller training phase. In fact, the trajectory is predefined and does not need to be learned.

Given the presence of several uncertainties, such as the approximations related to the neural networks employed, it was necessary to test the actuation pattern by performing the throw and recording the trajectory to assess if the task was accomplished or not (object in/object out). By attaching markers to the thrown object, we also tracked the trajectories of the real throws to obtain quantitative information. Each marker has a weight of 0.9 g and a characteristic dimension of 17 mm. This procedure introduced a slight discrepancy between the quantitative and the qualitative results, as the tossed object differed between the two conditions.

EXPERIMENTAL SETUP

We tested our deep RL-based controller on an experimental setup that consists of the following:

- *the robotic platform*: a serial concatenation of the I-SUPPORT robot arm [16] and a custom-designed soft gripper
- *the actuation unit*: custom-made electronics to provide electronic input signals to the hardware platform

- *sensors*: vision-based motion capture (VICON Motion Capture Ltd).

The I-Support robot arm [Figure 3(a)] is a pneumatically actuated modular soft robotic manipulator, where a single module [16] embeds three pairs of McKibben-like actuators arranged 120° from each other. In this work, two modules have been connected together through a ring-shaped interface that is made of Plexiglas (width: 3 mm), with an offset in the orientation by 60° with respect to each other. Furthermore, to reduce the overall weight of the system, the modules have a tapering shape. The robot is fixed onto an aluminum-based rectangular supporting frame of 20×20 mm in a vertically downward position to reduce gravitational effects. In the resting position, the robot arm has a total length of 409 mm and weight of 230.5 g and is suspended 1 m above the ground. Below the robot, we placed the targets as represented in Figure 1(a); each one is represented by a square container, whose dimensions are $140 \times 140 \times 100$ mm. In our experiments, we aimed at these containers with different objects, as shown in Figure 1(b), whose masses and dimensions can be different from each other—specifically, the following:

- a ping-pong ball, whose mass and diameter are equal to 0.8 g and 36 m, respectively
- a lemon (toy), which weighs 4.8 g and has a characteristic dimension of 62 mm
- a Vicon marker, whose mass and diameter are equal to 3.2 g and 26 mm, respectively
- a tomato (toy), which weighs 10.2 g and has a characteristic dimension of 63 mm.

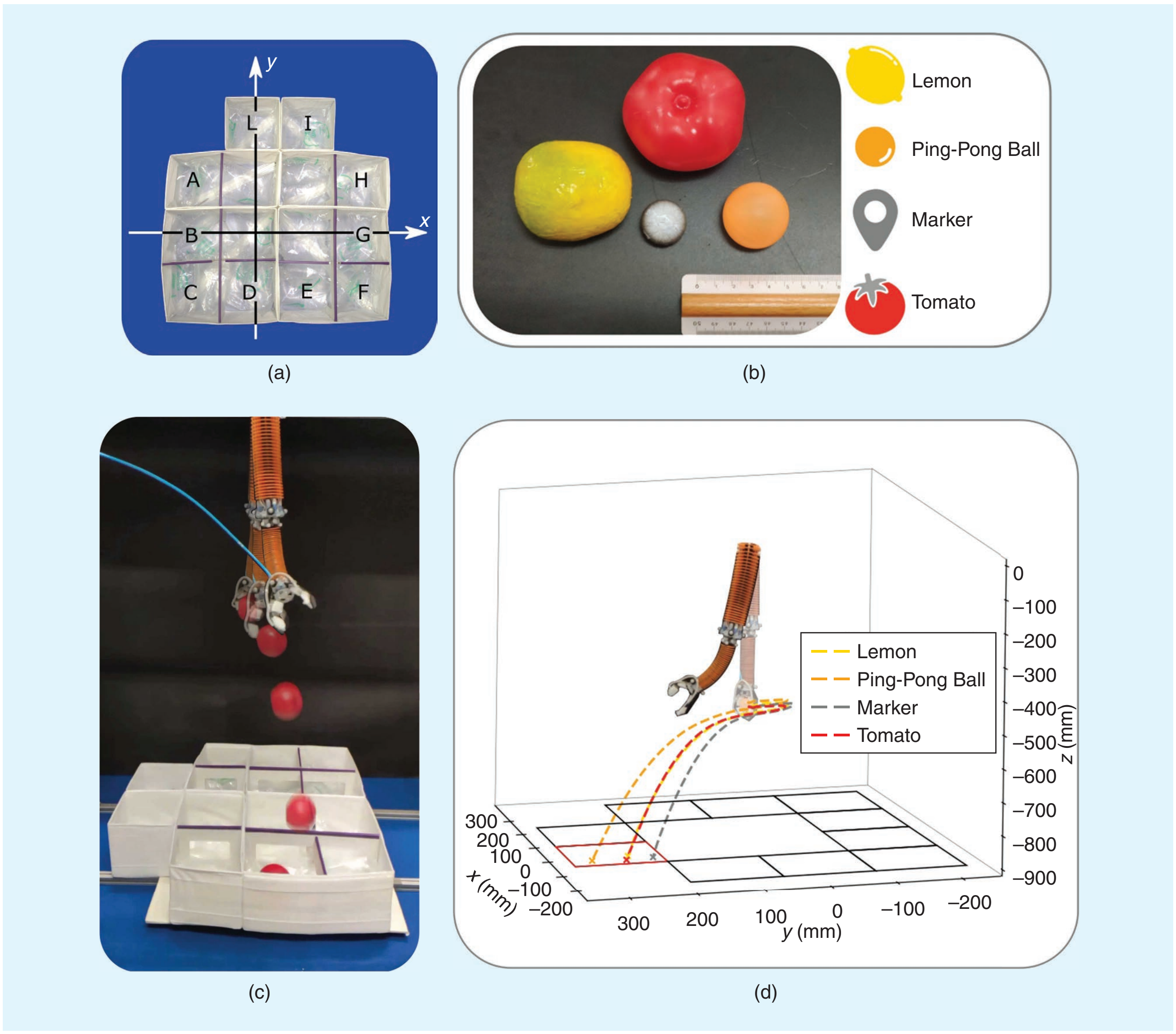

FIGURE 1. The experiments performed. (a) The targets used for the different trials. Ten goal points have been selected around the robot to explore the potentiality of our controller on the entire workspace of the soft robotic arm. Inside each box there are air pillows to avoid any damage to the tossed objects and the Vicon markers glued on them. (b) The objects thrown during the experiments. (c) The robot while performing a throw of the tomato toward target B. (d) The trajectories of the thrown objects toward target L using the complete actuation for each object.

More specifically, the last object, the "tomato," was selected to test the robustness of the open-loop controller since its mass and dimension are outside the set of the others used to collect the dataset.

The custom-designed gripper is screwed to the distal module. It comprises two fingers composed of 3D-printed surfaces, which are joined together by two thermoplastic polyurethane chambers realized with a heat press. The gripper has an overall weight of 27.5 g. The time required to open the gripper has a tremendous impact on the experiments; this motion has been characterized as follows: 50 closing and opening cycles have been carried out to quantify this delay. For each of them, we measured the time between the instant at which the opening command was sent ($t = 1$ s) and the instant at which the distance between the two appendixes is ~ 60 mm, which approximately corresponds to the greatest dimension of the thrown objects. From Figure 3(b), it is possible to appreciate the trends of the distance between the appendixes of the gripper and the one related to the pressure value applied to the component. We assessed that the opening time of the gripper is equal to ~ 0.2 s.

THROWING TASK SPECIFICATIONS

In our tests, the robot moves first in the opposite direction with respect to the target (run-up phase) and then toward it (forward phase). The division of the movement aims at increasing the speed of the I-Support arm during the throw and, consequently, expanding the range of the target positions.

In a module comprising a triad of radially arranged actuators, the activation of a single actuator will generate a bending in a specific direction. Consequently, a module will have a minimum of three major bending directions in the Cartesian space, each passing from the respective pair of pneumatic chambers. In this way, for the system under consideration [16], there will be a total of six bending directions, as illustrated in Figure 4. In particular, note that the bending motion has been

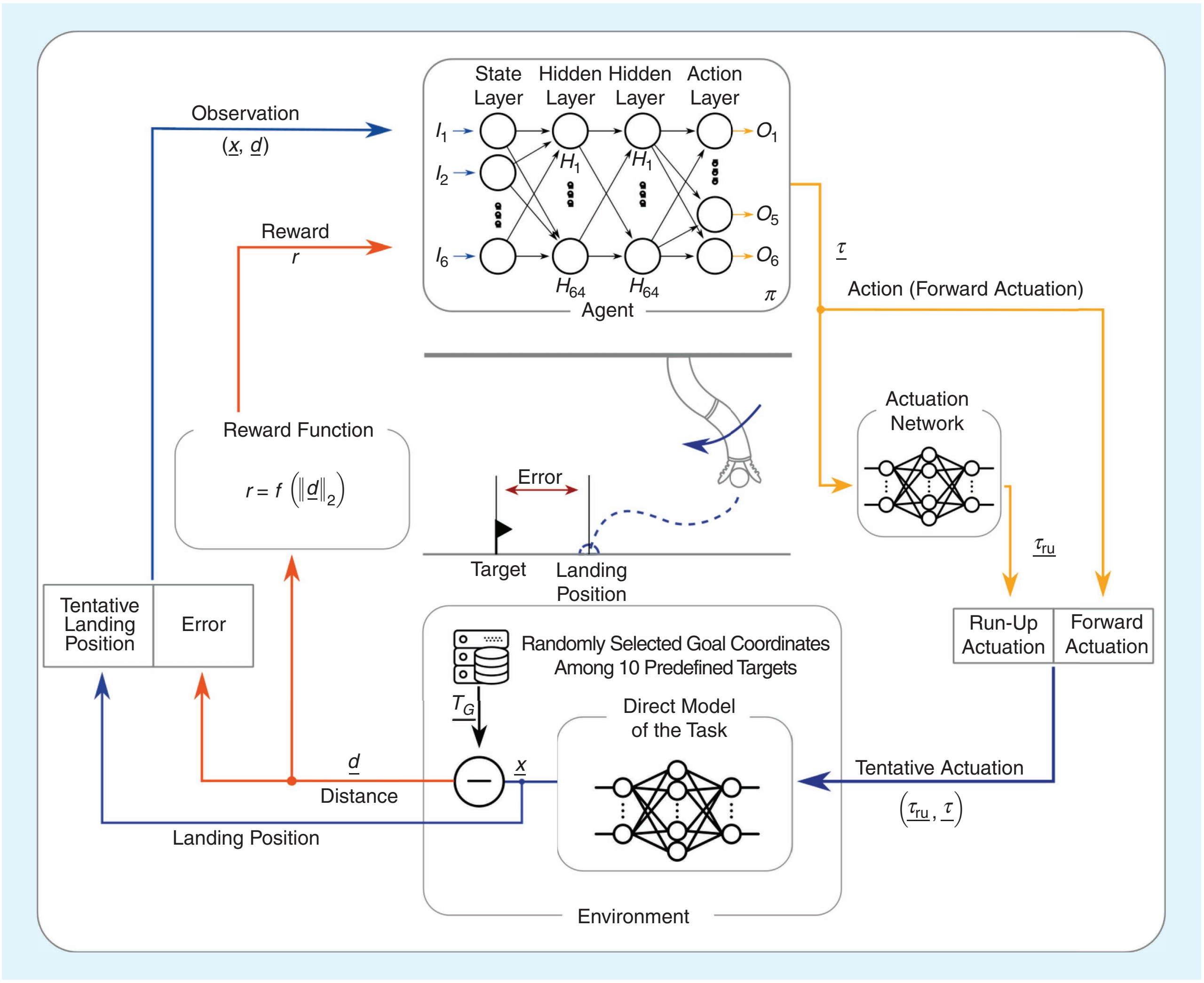

FIGURE 2. The RL framework. The agent generates the actuation given the observation. Thanks to the actuation network, we can respect the functional relation between the input patterns to the soft robot in the two phases in which we divided the movement (run-up/forward). Then, the DM of the task is used to predict the landing position associated with the actuation. The agent receives a reward based on the Euclidean distance between the landing position and the target. In this way, the agent learns to reduce the distance from the goal position by looking at the error and at the previously reached position.

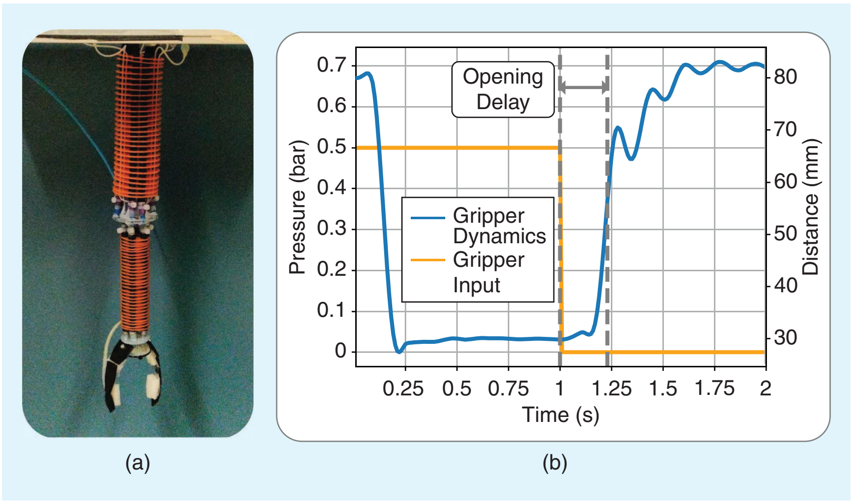

FIGURE 3. The soft robotic arm. (a) The I-Support arm with two modules and a gripper. (b) The gripper opening dynamics. The graph shows the trends of the air pressure input for the gripper and the distance between its two fingers.

considered with respect to a local planar reference frame associated with the cross section of the robot arm. For example, the inflation of chamber 1 results in a bending along the negative y-axis. In addition, it is important to avoid any interference between the proximal and distal modules in the case of simultaneous actuation of the chambers displaced along the same axis, e.g., chambers 1 and 5.

Given a desired goal $T_G(x_{T_G}; y_{T_G})$, the actuation required to throw an object is determined as follows: considering the desired goal T_G, an objective point is obtained that has coordinate values that are diametrically opposite to those of the desired goal, $T = (x_T; y_T) = (-y_{T_G}; -x_{T_G})$. Then, this objective point is projected on a unit circle, represented as $T'(x_{T'}; y_{T'})$, and is exploited to obtain the corresponding rectangular coordinates on all six major bending axes using the formulas summarized in Table 1. In particular, the length of each rectangular projection represents a direct proportionality between the projection on the axis and the pressure applied to the actuator. In other words, when the projection of the radius (OT') is equal to one, it means that the pressure applied to that specific actuator is maximum; in this case, it is restricted to $p_{\max} = 1$ bar.

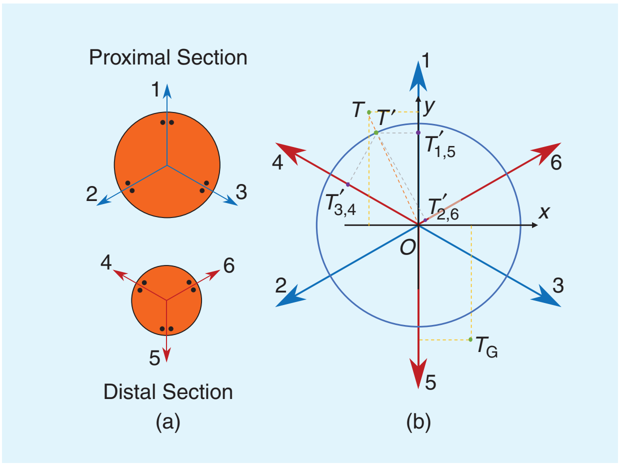

FIGURE 4. The geometrical model of the throwing trajectory lying on the plane passing through the z-axis. (a) The major bending directions of the proximal module are represented by directions 1–2–3, while those of the distal part are represented by directions 4–5–6. (b) Given a goal point (T_G), we first find its symmetric (T) and its projection onto a unit circle (T'). Then, projecting T' onto the major bending directions, it is possible to find the pressure inputs needed to move toward the goal.

DATASET COLLECTION

Having established the geometrical relationship between the input activation and the throwing direction, we can now start to collect a database of trajectories.

As stated previously, the movement of the manipulator is divided into two phases, i.e., the run-up phase and the forward phase. In particular, it has been empirically determined that the overall movement will last $t_{\text{total}} = 2$ s, of which the first $t_{\text{run-up}} = 0.5$ s is spent for the run-up phase. The object is released at $t = 0.97$ s because we observed that, around this time instant, the robot reaches the lower position of the forward phase, where the favorable effect of gravity on movement ends. In this way, each trajectory is essentially characterized by one time instant, i.e., the one where the object would be released. To create a dataset, more than 1,000 trajectories have been collected. For each one, we randomly selected a point and followed the procedure described in the "Throwing Task Specifications" section, summarized in Table 1, to find the actuation pattern for the two phases. The data collection process took two hours and 15 min. This work aims to teach a robot how to throw, so, from the dataset of position and

TABLE 1. Actuations given a point.

DIRECTION	x_{T_i}	y_{T_i}	CONDITION
1	0	$y_{T'}$	$y_1 \geq 0$
2	$\frac{3x_{T'} + \sqrt{3}y_{T'}}{4}$	$\frac{\sqrt{3}x_{T'} + y_{T'}}{4}$	$x_2 \leq 0;\ \ y_2 \leq 0$
3	$\frac{3x_{T'} - \sqrt{3}y_{T'}}{4}$	$\frac{y_{T'} - \sqrt{3}x_{T'}}{4}$	$x_3 \geq 0;\ \ y_3 \leq 0$
4	$\frac{3x_{T'} - \sqrt{3}y_{T'}}{4}$	$\frac{y_{T'} - \sqrt{3}x_{T'}}{4}$	$x_4 \leq 0;\ \ y_4 \geq 0$
5	0	$y_{T'}$	$y_5 \leq 0$
6	$\frac{3x_{T'} + \sqrt{3}y_{T'}}{4}$	$\frac{\sqrt{3}x_{T'} + y_{T'}}{4}$	$x_6 \geq 0;\ \ y_6 \geq 0$

The geometrical relationship between the actuation inputs of the I-Support robotic arm used to restrict the overall motion to a plane. T' represents the projection on a unit circle with coordinates that are diametrically opposite to the desired goal T_G.

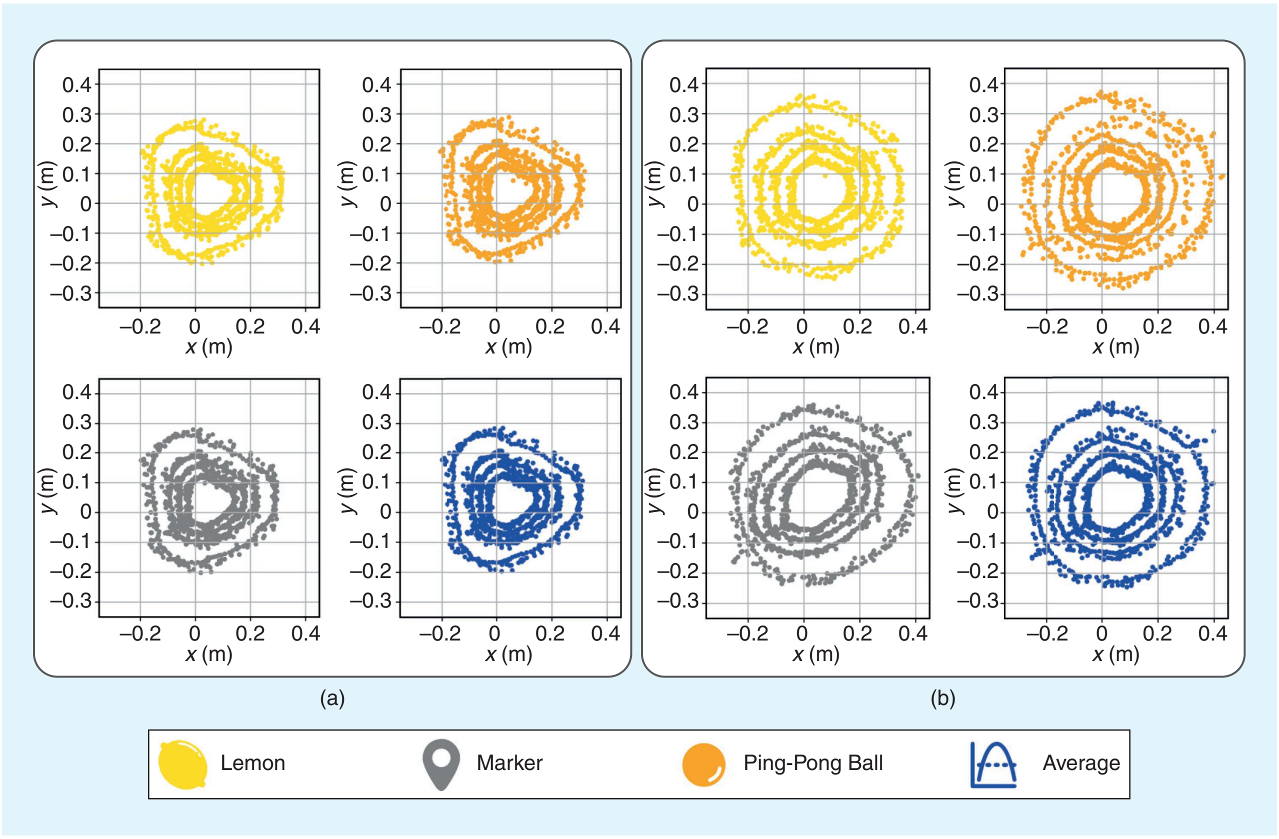

FIGURE 5. The training datasets collected for the (a) partial and (b) complete actuation scenarios, respectively. Each point is the result of the simulated throw, starting from a trajectory generated with the actuation relationships summarized in Table 1.

velocity of the end effector, a dataset of landing positions is generated through the projectile equations.

The dataset has been collected three times. In each case, the object that the gripper was holding has been changed from the ping-pong ball to the marker and then the lemon (toy). Then, the final dataset was obtained, which was used to train the networks, by averaging the landing positions in these three cases. This last operation aims to take into account the different weights of the held objects from the ones used to train the networks to build a controller that is not object specific. This procedure is repeated for both actuation scenarios, partial (just the proximal module) and complete (the joined actuation of the distal and proximal modules). Figure 5(a) and (b) shows, respectively, the datasets when one module or two modules were actuated with the different gripping objects and the final two datasets that then have been used to train the ANN to obtain the DM.

TABLE 2. The model selection.

	MODEL			
HYPERPARAMETERS	**DM OF THE TASK**		**ACTUATION NETWORK**	
Actuation	Partial	Complete	Partial	Complete
Number of units	64	64	128	128
Activation function	ReLU	ReLU	ReLU	ReLU
Normalization	Rescaling	Z-score	—	—
Error	4.59 mm	6.58 mm	0.07 bar	0.02 bar

The best combination of hyperparameters for the DM and the actuation network in the two actuation scenarios. ReLU: rectified linear unit.

ANN TRAINING

After the collection of the dataset, with an ANN, we mapped the input space with the resulting landing positions of the thrown object as in

$$\underline{x} = f(\underline{\tau_{\mathrm{ru}}}, \tau). \qquad (1)$$

Here, $(\underline{\tau_{\mathrm{ru}}}, \underline{\tau})$ are, respectively, the actuation input of the run-up and forward phases, and $\underline{x}$ represents the resulting landing position. Equation (1) represents the DM. It consists of an ANN with one hidden layer. We performed a model selection based on the average Euclidean distance from the desired target to choose the best set of hyperparameters. The results are presented in Table 2.

We trained the model (1) using the actuation inputs for both phases. To decrease its prediction error, it is necessary to have an input that is consistent with the inputs used during training. This led to the need for the actuation network (2). It is a one-hidden-layer ANN, which, given the forward actuation pattern, predicts the one for the run-up phase. In this case, the model selection, whose results are displayed in Table 2, was based on the average error on the actuation components:

$$\underline{\tau_{\mathrm{ru}}} = g(\underline{\tau}). \tag{2}$$

PROXIMAL POLICY OPTIMIZATION

These two neural networks were the building blocks of the RL environment, as shown in Figure 2. In our framework, the agent, a multilayer perceptron, learns how to set the actuation $(\underline{\tau})$ of the soft robot for tossing the object in the desired target. The agent has to guess the actuation responsible for the forward movement by looking at the landing position associated with the previous tentative actuation pattern $(\underline{x})$ and its distance from the desired target $(\underline{d})$. Once the agent infers the best action to take, the correspondent run-up actuation $(\underline{\tau_{\mathrm{ru}}})$ is predicted thanks to the actuation network (2). Then, with the DM (1), the toss is simulated. For each toss, the agent receives a reward based on the distance $(\| \underline{d} \|)_2$ between the landing point associated with the tentative actuation and the target position. More specifically, it is the sum of two contributions: a continuous part, which is always negative and equals to the distance, in millimeters, from the target, and a pseudostep part, which is positive. The overall reward can be written as

$$r(\| d \|) = -\| d \| + \begin{cases} 3{,}000 & \text{if } \| d \| < 2 \text{ mm} \\ 60(10 - \| d \|) + 1{,}200 & \text{if } 2 \le \| d \| < 10 \text{ mm} \\ 40(30 - \| d \|) + 400 & \text{if } 10 \le \| d \| < 30 \text{ mm} \\ 20(50 - \| d \|) & \text{if } 30 \le \| d \| < 50 \text{ mm} \end{cases}. \tag{3}$$

Such a reward function allows the agent to improve its performances quickly. Indeed, there is a massive reward when the target is hit.

We wanted to exploit the goal-direct learning and decision making that characterize every RL algorithm [17]. More specifically, we opted for the proximal policy optimization algorithm [18] since it is one of the most sampling-efficient algorithms with respect to other policy gradient-based approaches. In addition, it is less affected by the hyperparameter initialization, and it can be applied to a wide variety of RL tasks.

Once trained, the RL agent predicts, in a deterministic mode, the actuation pattern to toss the objects. The prediction errors tested on the DM for both the actuation scenarios are shown in Table 3.

Despite being within acceptable limits, the average errors associated with the RL predictions remain relatively high. We attribute this to the stochastic behavior of the soft arm and the approximation errors introduced by the building blocks of the controller, which relies on a DM based on an average dataset of the different objects. A comparison with object-specific controllers can be found in the supplementary material available at https://doi.org/10.1109/MRA.2023.3310865.

TABLE 3. The RL performances: errors in millimeters associated with the prediction of the RL agent tested in the DM.

SCENARIO	PARTIAL	COMPLETE
MINIMUM	2.43	19.07
AVERAGE	24.38	30.55
MAXIMUM	48.18	49.5

RESULTS

Our results show that the SofToss controller reaches a performance in successfully tossing objects to target locations as high as 62%–63% on average (depending on the actuation pattern used) and consistently more than 80% in the best-performing directions for each actuation pattern. Furthermore, it shows a performance similar to that of a nonreal-time controller that uses optimization [19]. SofToss shows a significantly shorter time spent for finding a new actuation pattern after changing the target, which makes our system work in real time.

In the experiments, we tossed four objects, different in size and weight, as shown in Figure 1(b). We selected goal points that cover the whole workspace and different directions. We tested the proposed controller in two scenarios: 1) partial actuation, where only the proximal module is actuated, and 2) complete actuation, where both modules are actuated. Both quantitative (involving the tracking of the object trajectory) and qualitative (without trajectory tracking) tests have been performed to assess the ability of the controller to generate an appropriate actuation pattern that allows tossing an object into the target. Figure 1(c) and (d) show the robot performing such tests. For each combination of parameters (the object type, qualitative/quantitative test, and target position), three trials have been recorded for a total of 240 throws for each actuation scenario.

The comparison between the partial and complete actuation scenario tests is summarized in Figure 6(a). In Figure 6(a), we show a box plot based on the average error, i.e., the Euclidean distance between the target and the actual landing position, across the three trials for each combination of object–target. This graph shows that the average performances of the controllers in the two actuation scenarios are remarkably similar. By measuring the error as the distance of the object landing point from the center of the target location, we obtain 68.3 mm for the complete actuation scenario and 61.7 mm for the partial one. For the qualitative results, we consider the success rate, i.e., the number of successful landings of the object inside the target box (with or without the bouncing of the object on its edge) with respect to the total number of throws. We obtain 62% for the complete scenario and 63% for the partial one.

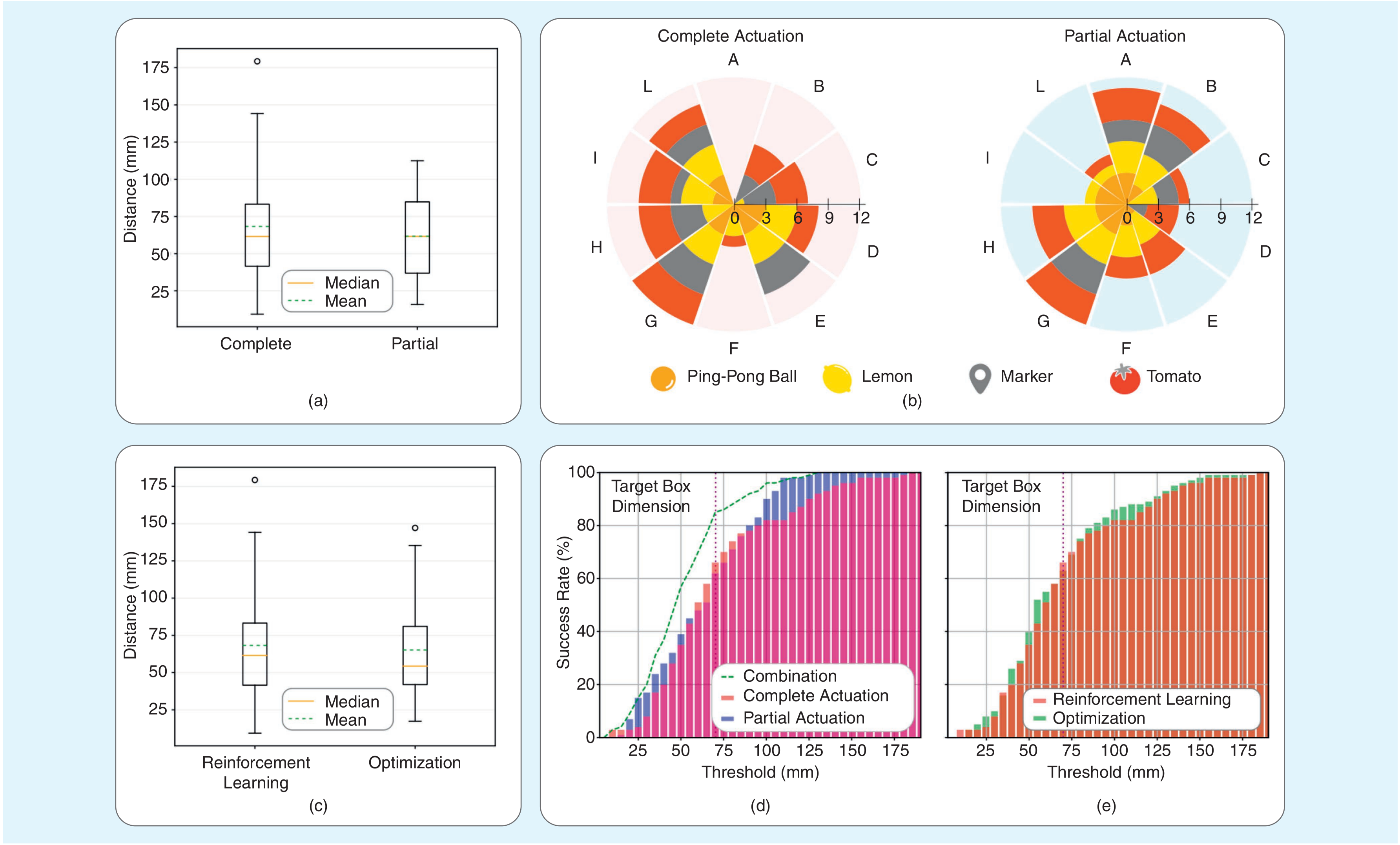

FIGURE 6. The performance of the controllers. (a) A comparison between the RL-based controllers in different actuation scenarios based on the distance from the target position, which was reconstructed by recording the falling trajectories of the object during a toss. (b) The qualitative results for the RL-based controllers, showing the number of successful trials for each target. This graph shows the ability of the RL-based controller to toss the objects in different directions in the two different actuation scenarios analyzed. The number of successful trials for each is indicated by the radius of the angular sector. Furthermore, the presence of empty sectors is due to the fact that the controller fails to launch any of the objects considered in that particular sector. (c) A comparison between our controller and an optimization-based nonreal-time approach previously evaluated on a soft robotic platform simulator [19]. In both cases, the data are represented as a box plot to represent the variability of the trials. (d) and (e) The success rates of the different controllers against the dimension of the threshold that we can select to decide if a trial is successful or not for (d) RL-based controllers and (e) the complete actuation scenario. In these graphs, the vertical dotted line represents half of the characteristic dimension of the target container (70 mm).

These latter results are obtained through the analysis of the recorded videos. In addition, Figure 6(d) shows the success rate of the two scenarios with respect to threshold errors. By considering the 70-mm threshold value as an approximation of the box size, it is possible to appreciate a success rate greater than 60%, confirming the success rate derived qualitatively.

A comparison with the nonreal-time controller [19], presented in Figure 7, is summarized in Figure 6(c). This controller is based on the basin-hopping optimization algorithm applied to the DM (1). The optimization aims at minimizing the value function represented by the difference between the desired position and the landing position predicted by (1) given the generated actuation pattern.

To compare the two controllers, we tested the optimization-based one over the same targets in the complete actuation scenario. We thus performed another 240 throws.

In Figure 6(c), it is possible to note that our strategy performances are close to the nonreal-time ones; indeed, their average error is equal to 65.2 mm. Figure 6(e) shows a comparison between the nonreal-time approach and the proposed controller in terms of the success rate. The controller performances are similar since the success rate for both strategies has a similar trend. The nonreal-time controller achieves a success rate of 63% considering the threshold of 70-mm, while the RL-based one reaches a success rate of 66%.

DISCUSSION AND CONCLUSION

We showed that, despite a smaller workspace and lower dynamics than a conventional robot, with our controller, a soft robot can perform the dynamic task of tossing an object inside a target area. We tested our deep RL-based controller in different actuation scenarios, where we obtained a success rate greater than 60%.

One source of error in our system is the stochastic behavior of the soft arm. We quantified it by observing the variations in output under the actuation input pattern used for throwing (here, we just consider the arm movement, not the gripper). By considering the releasing point in all trials for the same trajectory (used to throw in a specific target), we derived a parallelepiped that includes all of these points to have a measure of their variability. All of these parallelepipeds have been averaged (considering all of the targets in the two actuation scenarios), producing a variability volume that has the following size: $17 \times 13 \times 4$ mm. The same phenomenon occurs with the initial resting position of the robot gripper, which varied throughout the trials and was contained within a volume of $7 \times 5 \times 2$ mm. In addition, the initial position of the object in the gripper has a variability volume of $14 \times 11 \times 32$ mm. This is mainly due to the manual positioning of the object in the gripper. Additionally, since the neural networks are trained on an average dataset and not on the dataset related to a specific object, the controller generates a suboptimal condition for each tossed item.

Nevertheless, when the threshold is near 100 mm, the behavior of the controller in the complete actuation scenario presents a plateau with respect to the partial actuation case. This is due to the inability of the controller to generate appropriate actuation patterns for some specific directions. Indeed,

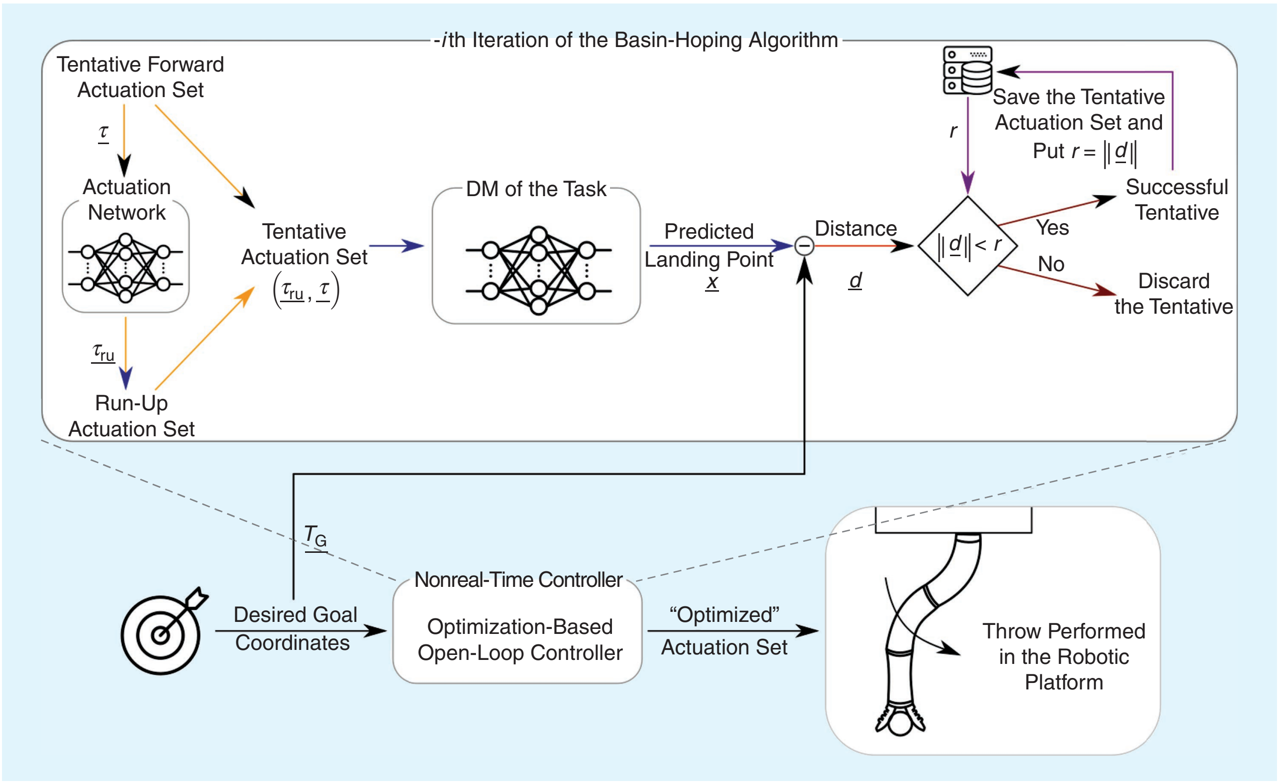

FIGURE 7. The nonreal-time controller. The control scheme of the optimization-based approach on which the controller is based is described in [19].

as shown in Figure 6(b), the controller is not able to toss any of the objects inside target A. Therefore, there are at least 12 cases in which the error is more significant than the one in the partial scenario, and the distance from the goal is bigger than the dimension of the target. The presence of preferred directions in which the controller is able to throw objects is also found in the partial actuation scenario. This characteristic can be associated with the controller generating some actuation patterns that make the soft robot move abnormally (e.g., generating higher acceleration). As a result, the held object is released in a nonoptimal condition, causing a failure. Additionally, some uncertainties can be due to the geometrical approximation we relied on for the controller definition.

By taking into account this peculiarity, we can realize a controller in which the actuation scenario also becomes a parameter. According to this strategy, given the target container position, we can select how many modules should be actuated to maximize the performance of the toss. This simple approach allows us to bring the success rate to 85% in correspondence with the characteristic dimension of the target (70 mm) as presented in Figure 6(d).

Throwing tasks, in both biological and robotic studies, can be found in the literature. For instance, in [20], the authors compared the throwing performances of capuchin monkeys and humans. Monkeys were required to throw stones weighing between 15 and 25 g into a target bucket (whose diameter was 300 mm) located at 300 or 600 mm from the animal. Similarly, the humans had to throw tennis balls weighing 57 g into a basket (whose diameter was 550 mm) at a distance of 3,000 or 6,000 mm. The success rates for the monkeys were 68% for the closest target and 34% for the furthest, while, for the humans, they were, respectively, 71% and 31%. From the robotic side, in [3], the authors used a rigid robot (UR5) to perform pick-and-toss tasks of objects into rectangular target boxes with dimensions of $250 \times 150 \times 100$ mm, achieving a success rate of 82.3%. The authors stated that this success rate was highly influenced by the precision of the grasping phase. Though the task space for our robot is smaller than the one presented in the studies described, the success rate is comparable to that of the rigid robot if one could choose the actuation pattern to be used. Alternatively, it could be compared to that of humans and monkeys.

Figure 6(c) and (e) show the comparison between the nonreal-time approach with our controller in terms of the success rate. As already discussed, their average error and variance behavior are similar, thanks to the presence of the same building blocks in the two strategies i.e., neural networks, and so they are affected by the same approximation errors. For this reason, as expected, in Figure 6(e), the curves that describe their behavior tend to overlap. In any case, the deep RL-based strategy presents the considerable advantage that it does not need to wait for the optimization process to end when we change the target. Indeed, in this case, we measured that less than 1 s was required to find the actuation with the deep RL strategy, while more than 1 min was necessary to change the target with the nonreal-time controller.

To increase the controller performance, it is critical to focus on the gripper, as it has a considerable effect on this task. This element influences the releasing point of the object and, consequently, the object initial kinematic conditions in its freefall toward the target. In this work, to develop the model of the task, we simulated several throws given the trajectories starting from the kinematic condition of the object held in the gripper in a precise time instant. At the same time, delay estimation characterization of the gripper was conducted only for a defined dimension of the object. This was because, as for the learning on the average dataset, we wanted to challenge the generalization capabilities of the controller. Indeed, small objects (e.g., markers) are released earlier than the others, thus in suboptimal conditions.

In conclusion, the performances of our system are affected not only by the stochasticity of the soft arm dynamics and the variability in the gripper realizing time but also by the learning method and the task model. The concurrent effect of all of these components makes it difficult to assess the effect of the proposed RL controller compared to the other factors. In addition, another limitation of our work is that we have not included methods to alleviate the simulation-to-real gap problem (the controller derived in silico is applied to the robot without any adjustment). Future works can address this issue by including online optimization methods to reduce the mismatch.

ACKNOWLEDGMENT

We acknowledge Andrea Centurelli for the help provided during the development of the proposed methodology and Muhammad Sunny Nazeer for the suggestions on the RL algorithms.

This work was partially supported by the European Union’s Horizon 2020 Future and Emerging Technologies-Open program under Grant Agreement 863212 [https://proboscis.eu (PROBOSCIS project)]. Diego Bianchi and Egidio Falotico are the corresponding authors.

This article has supplementary downloadable material available at https://doi.org/10.1109/MRA.2023.3310865, provided by the authors. This includes a PDF file with tables containing experimental raw data and additional information regarding the learning algorithm model. In addition, a video (SofToss_SMSummary.mp4) presents the work and some of the clips recorded during the experiments. This material is 3 MB in size.

AUTHORS

Diego Bianchi, The BioRobotics Institute, Scuola Superiore Sant’Anna, 56025 Pontedera, Italy, and the Department of Excellence in Robotics and Artificial Intelligence, Scuola Superiore Sant’Anna, 56127 Pisa, Italy. E-mail: diego.bianchi@santannapisa.it.

Michele Gabrio Antonelli, Department of Industrial and Information Engineering and Economics, University of L’Aquila, 67100 L’Aquila, Italy. E-mail: gabrio.antonelli@univaq.it.

Cecilia Laschi, Department of Mechanical Engineering, National University of Singapore, Singapore: 117575, Singapore. E-mail: mpeclc@nus.edu.sg.

Angelo Maria Sabatini, The BioRobotics Institute, Scuola Superiore Sant'Anna, 56025 Pontedera, Italy, and the Department of Excellence in Robotics and Artificial Intelligence, Scuola Superiore Sant'Anna, 56127 Pisa, Italy. E-mail: angelo.sabatini@santannapisa.it.

Egidio Falotico, The BioRobotics Institute, Scuola Superiore Sant'Anna, 56025 Pontedera, Italy, and Department of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, 56127 Pisa, Italy. E-mail: egidio.falotico@santannapisa.it.

REFERENCES

[1] F. Raptopoulos, M. Koskinopoulou, and M. Maniadakis, "Robotic pick-and-toss facilitates urban waste sorting," *in Proc. IEEE 16th Int. Conf. Automat. Sci. Eng. (CASE)*, Aug. 2020, pp. 1149–1154, doi: 10.1109/CASE48305.2020.9216746.

[2] Z. Fang, Y. Hou, and J. Li, "A pick-and-throw method for enhancing robotic sorting ability via deep reinforcement learning," *in Proc. 36th Youth Acad. Annu. Conf. Chin. Assoc. Automat. (YAC)*, May 2021, pp. 479–484, doi: 10.1109/YAC53711.2021.9486466.

[3] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "TossingBot: Learning to throw arbitrary objects with residual physics," May 2020, *arXiv:1903.11239.*

[4] D. Büchler, R. Calandra, and J. Peters, "Learning to control highly accelerated ballistic movements on muscular robots," *Robot. Auton. Syst*., vol. 159, Aug. 2022, Art. no. 104230, doi: 10.1016/j.robot.2022.104230.

[5] D. J. Braun, M. Howard, and S. Vijayakumar, "Exploiting variable stiffness in explosive movement tasks," *in Proc. Robot., Sci. Syst. VII*, 2012, vol. 7, pp. 25–32.

[6] A. Albu-Schaffer et al., "Soft robotics," *IEEE Robot. Autom. Mag*., vol. 15, no. 3, pp. 20–30, Sep. 2008, doi: 10.1109/MRA.2008.927979.

[7] D. Rus and M. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, May 2015, doi: 10.1038/nature14543.

[8] O. Fischer, Y. Toshimitsu, A. Kazemipour, and R. K. Katzschmann, "Dynamic task space control enables soft manipulators to perform real-world tasks," *Adv. Intell. Syst*., vol. 5, no. 1, Sep. 2022, Art. no. 2200024, doi: 10.1002/aisy.202200024.

[9] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft Robot*., vol. 5, no. 2, pp. 149–163, Apr. 2018, doi: 10.1089/soro.2017.0007.

[10] D. Kim et al., "Review of machine learning methods in soft robotics," *PLoS One,* vol. 16, no. 2, Feb. 2021, Art. no. e0246102, doi: 10.1371/journal.pone.0246102.

[11] F. Piqué, H. T. Kalidindi, L. Fruzzetti, C. Laschi, A. Menciassi, and E. Falotico, "Controlling soft robotic arms using continual learning," *IEEE Robot. Autom. Lett*., vol. 7, no. 2, pp. 5469–5476, Apr. 2022, doi: 10.1109/LRA.2022.3157369.

[12] A. Centurelli, L. Arleo, A. Rizzo, S. Tolu, C. Laschi, and E. Falotico, "Closed-loop dynamic control of a soft manipulator using deep reinforcement learning," *IEEE Robot. Autom. Lett*., vol. 7, no. 2, pp. 4741–4748, Apr. 2022, doi: 10.1109/LRA.2022.3146903.

[13] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Koopman-based control of a soft continuum manipulator under variable loading conditions," *IEEE Robot. Autom. Lett*., vol. 6, no. 4, pp. 6852–6859, Oct. 2021, doi: 10.1109/LRA.2021.3095268.

[14] G. Li, J. Shintake, and M. Hayashibe, "Deep reinforcement learning framework for underwater locomotion of soft robot," *in Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 12,033–12,039, doi: 10.1109/ICRA48506.2021.9561145.

[15] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Trans. Robot*., vol. 35, no. 1, pp. 124–134, Feb. 2019, doi: 10.1109/TRO.2018.2878318.

[16] M. Manti, A. Pratesi, E. Falotico, M. Cianchetti, and C. Laschi, "Soft assistive robot for personal care of elderly people," *in Proc. 6th IEEE Int. Conf. Biomed. Robot. Biomechatronics (BioRob)*, Jun. 2016, pp. 833–838, doi: 10.1109/BIOROB.2016.7523731.

[17] R. S. Sutton and A. G. *Barto, Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," Aug. 2017, *arXiv:1707.06347.*

[19] D. Bianchi, M. Antonelli, C. Laschi, and E. Falotico, "Open-loop control of a soft arm in throwing tasks," *in Proc. 19th Int. Conf. Informat. Contr., Automat. Robot*., Jul. 2022, pp. 138–145, doi: 10.5220/0011267100003271.

[20] G. Westergaard, C. Liv, M. Haynie, and S. Suomi, "A comparative study of aimed throwing by monkeys and humans," *Neuropsychologia*, vol. 38, no. 11, pp. 1511–1517, Oct. 2000, doi: 10.1016/S0028-3932(00)00056-7.

Open Access funding provided by 'Scuola Superiore "S.Anna" di Studi Universitari e di Perfezionamento' within the CRUI CARE Agreement