Learning-Based Inverse Dynamic Controller for Throwing Tasks with a Soft Robotic Arm

Diego Bianchi[1,2][a], Michele Gabrio Antonelli[3][b], Cecilia Laschi[4][c], Angelo Maria Sabatini[1,2][d] and Egidio Falotico[1,2][e]

[1]*The BioRobotics Institute, Scuola Superiore Sant'Anna, Pontedera, Italy*

[2]*Departement of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, Pisa, Italy*

[3]*Department of Industrial and Information Engineering and Economics, University of L'Aquila, L'Aquila, Italy*

[4]*Department of Mechanical Engineering, National University of Singapore, Singapore, Singapore*

Keywords: Soft Robotics, Throwing, Open-Loop Control, Neural Network, Ballistic Task.

Abstract: Controlling a soft robot poses a challenge due to its mechanical characteristics. Although the manufacturing process is well-established, there are still shortcomings in their control, which often limits them to static tasks. In this study, we aim to address some of these limitations by introducing a neural network-based controller specifically designed for the throwing task using a soft robotic arm. Drawing inspiration from previous research, we have devised a method for controlling the movement of the soft robotic arm during the ballistic task. By employing a feed-forward neural network, we approximate the relationship between the actuation pattern and the resulting landing position. This enables us to predict the input sequence that needs to be transmitted to the robot's actuators based on the desired landing coordinates. To validate our approach, we conducted experiments using a 2-module soft robotic arm, which was utilized to throw four different objects towards ten target boxes positioned beneath the robot. We considered two actuation modalities, depending on whether the distal module was activated. The results indicate a success rate, defined as the proportion of successful trials out of the total number of throws, of up to 68% when a single module was actuated. These findings demonstrate the potential of our proposed controller in achieving successful performance of the throwing task using a soft robotic arm.

1 INTRODUCTION

Soft robots, often inspired by biological systems (Laschi et al., 2016; Polygerinos et al., 2017), are constructed using hyper-elastic materials with a Young's modulus comparable to that of biological skin (Rus and Tolley, 2015), typically ranging from 10^4 to 10^9 Pa. These robots offer several advantages over traditional robots, including the potential for low-cost and safe human interaction due to their energy-absorbing capabilities during collisions. Despite their unique characteristics and wide range of potential applications, soft robots have yet to realize their full potential (Katzschmann et al., 2018; Holland et al., 2017), primarily due to sensing and control limitations.

While soft robots have been evaluated for dynamic tasks such as trajectory tracking, there are still numerous real-world dynamic tasks that remain unexplored (Fischer et al., 2022). In (Fischer et al., 2022), the authors demonstrate the potential of soft robots in various dynamic tasks, including ballistic tasks where objects are thrown along linear trajectories. However, there is a lack of control strategies specifically tailored for achieving ballistic tasks with soft robots, particularly in accurately tossing an object into a user-defined target box. Therefore, the objective of our work is to develop a control strategy to enable soft robots to perform ballistic tasks, precisely throwing objects into predetermined target boxes.

Despite classical robots, which are designed to maximise the performance of an operation, soft robots thanks to their materials are characterised by low-cost fabrication, high deformability, and compliance. However, this has a cost in terms of accuracy and

[a] https://orcid.org/0000-0001-7148-1612
[b] https://orcid.org/0000-0001-8437-9131
[c] https://orcid.org/0000-0001-5248-1043
[d] https://orcid.org/0000-0003-3306-6498
[e] https://orcid.org/0000-0001-8060-8080

Bianchi, D., Antonelli, M., Laschi, C., Sabatini, A. and Falotico, E.
Learning-Based Inverse Dynamic Controller for Throwing Tasks with a Soft Robotic Arm.
DOI: 10.5220/0012184200003543
In *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2023) - Volume 1*, pages 424-432
ISBN: 978-989-758-670-5; ISSN: 2184-2809

payload. These properties make them suitable for delicate handling(Li et al., 2017) and operation in unstructured environments as scenarios like assistive robotics(Manti et al., 2016) and search and rescue missions(Hawkes et al., 2017) can represent. However, the non-linearity of the materials combined with the fact that soft robots show virtually infinite degrees-of-freedom, and that their properties depend on the environment in which they work, require innovation in the control systems(Wang and Chortos, 2022) because it is challenging to apply directly the conventional robotic control theory which has been developed for rigid robots.

The main challenges that a soft robotic control system must face are high dimension morphology and the time-varying combined with the non-linear behaviour of the soft material. For these reasons, it is really challenging to model them. Furthermore, we need an accurate and computationally efficient representation to create a controller over the model. However, achieving both characteristics is difficult for a soft robotic model. Indeed, with Finite Element Analysis(Ilievski et al., 2011) (FEM) we can achieve the biggest accuracy among the other type of models which make some exemplificative assumptions, such as the Constant Curvature (Webster and Jones, 2010) (CC), the Piecewise Constant Curvature(Runge and Raatz, 2017) (PCC) and Cosserat rod models (Gazzola et al., 2018) (Alessi et al., 2023).

A promising alternative is represented by Machine Learning (ML) algorithms which could discover the underlying structure of the data that in our case is the model of the robot without any assumptions. ML techniques have already been used to create static and dynamic controllers (Laschi et al., 2023) even if there are some limitations. Firstly, since it is a data-driven approach to work properly it requires a (large) dataset whose collection may be time-consuming. Then, data might not represent the whole behaviour of the robot but just part of it i.e., the one captured in the dataset. In (Giorelli et al., 2013), the authors developed the earliest machine-learning controller for a non-redundant soft robot. They demonstrated the utilization of a neural network as an approximation of the inverse kinematics of the robot.

Later, this work has been extended in (George Thuruthel et al., 2017) to account for redundancies based on previous works (Vannucci et al., 2014; Vannucci et al., 2015). However, achieving a dynamic task with a quasi-static model or in general with a model that relies on the steady-state assumption is impossible since its accuracy during fast movements is limited. Due to severe limitations in speed, throwing would not be possible with controllers based on these models.

To overcome these limitations, model-free dynamic controllers have been developed. Different open-loop strategies (Thuruthel et al., 2019), (Centurelli et al., 2021) are present in the literature to control the manipulator dynamic in a trajectory tracking task. Both works represent a model-free approach in which recurrent neural networks are deployed to obtain the model of the soft robot. Then in (Centurelli et al., 2022) the authors designed a closed-loop controller able to deal with different payloads attached to the manipulator end-effector thanks to a deep reinforcement learning method. Recently, this tolerance to external payloads has been achieved with continual learning in (Piqué et al., 2022), where the authors learnt the dynamic models of the robot with different attached weights without encountering the well-known problem of catastrophic forgetting.

All the proposed approaches, have been developed for a specific task, the trajectory following, and are not suitable for ballistic movements, where reaching the realising point with a specified velocity determines the range of the throws. Indeed, for rigid robots, this ability has been implemented for example (Fang et al., 2021) and (Raptopoulos et al., 2020). Both experienced an increased work capacity and efficiency of a robot, especially in a weakly structured logistics sorting scene. While in (Büchler et al., 2022), the authors show the ballistic movement of a muscular robot, in particular, they highlight the issues associated with this fast movement on rigid robot hardware components. Instead in (Zeng et al., 2019), the authors deployed a hybrid controller to grasp an object with an anthropomorphic robot and then toss it at a specific target placed in front of it. The analytical model of the robot is used to make an estimation of the control parameters once the physical problem is solved thanks to some constraints imposed on the movement. As stated above, this approach is difficult to implement with a soft robot since the dynamic model of the robot is not always available or in general accurate. However, lately, this task has been explored also with soft robots. The feasibility of the ballistic task has been demonstrated by (Fischer et al., 2022). Meanwhile, two controllers have been developed, first in simulation and then on an real robot, for throwing an object into a target box. These controllers are described in (Bianchi et al., 2022) and (Bianchi et al., 2023). The simulation-based work utilizes an optimization process that provides the desired landing position and predicts the actuation patterns needed to reach it. In (Bianchi et al., 2023), the authors present a real-time controller that employs a deep reinforcement learning method.

In this work, we present a learning-based inverse dynamic controller approach to performing the tossing task with a soft robot. It is an open-loop controller in which we trained an artificial neural network to learn the relationship between the landing positions and the actuation patterns necessary to reach them. Since it is based on a single neural network, the inverse model of the task, the controller works in real-time because it instantaneously predicts the actuation pattern given a new goal position. We tested this real-time methodology on a modular pneumatic platform with which we tossed objects of different shapes and weights in several directions to explore the robot's capabilities and our controller.

2 MATERIALS AND METHODS

In this section, we will first present an overview of the approach we developed throughout the work. Then, after describing the experimental setup used to test the controller we will pass through the two main phases that brought to its development, i.e., the dataset collection and the training of the neural network on which our controller is based.

2.1 Our Approach

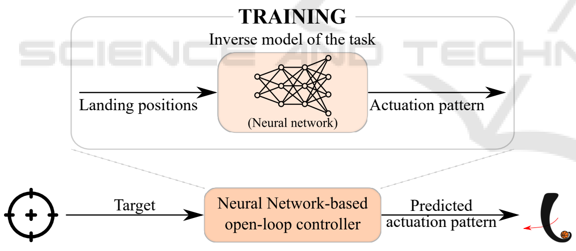
TRAINING
Inverse model of the task
Landing positions
(Neural network)
Actuation pattern
Target
Neural Network-based open-loop controller
Predicted actuation pattern

Figure 1: Neural Network-based open-loop controller.

Our approach is described in Figure 1. By learning the relationship between the landing positions and the actuation patterns to reach them in the training phase, our learning-based inverse dynamic controller is able to predict the actuation pattern according to the desired target position.

We defined the throwing trajectories before gathering the dataset to train the neural network. Inspired by (Braun et al., 2012), we opted for a planar trajectory. To enhance the dynamics of the soft robot we choose to perform a back-swing before moving toward the target goal. We can identify two distinct phases in the robot's movement: 1) the run-up phase; and 2) the forward phase. We established that the object should be released when the robot passes for the lowest position of the trajectory after the back-swing.

Once collected, we used the dataset to train a feed-forward artificial neural network, the inverse model of the task, to approximate the relationship between landing positions and actuation patterns, i.e., the commands that we sent to the robot. Each actuation pattern includes the commands responsible for the two phases of the movement.

Once trained, we used the inverse model of the task to perform the throws into target boxes. In this phase, we assess the effectiveness of the approach that is threatened by the approximation error of the neural network and the exemplifications made during the dataset collection. We conducted tests on our controller using objects of various shapes and weights, as well as different target locations.

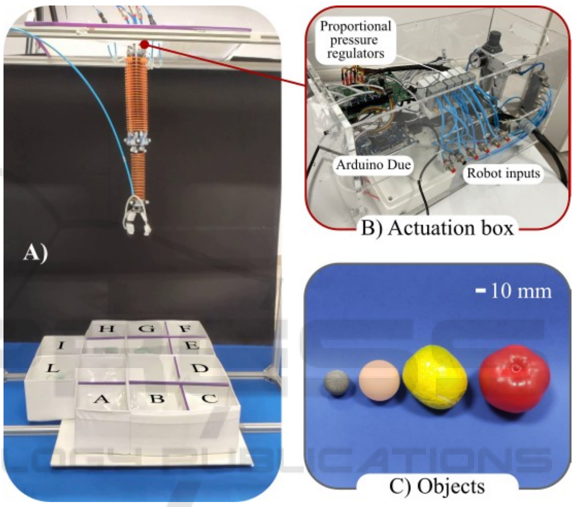

Figure 2: Experimental setup. A) The I-support robot equipped with a pneumatic gripper is positioned over ten designated target boxes used for the experiments. B) The control box used to actuate the robot. C) Various objects, including a toy lemon, a marker, and a ping-pong ball, were thrown as part of the experiments and were also utilized in training the neural networks.

2.2 Experimental Setup

We tested our controller on a modular robot, the I-Support (Manti et al., 2016)(Figure 2A). Each module presents three pairs of pneumatic McKibben-like actuators equally spaced on a circumference and there are several plastic disks to arrange the chambers in the module. To inflate the six chamber ($p_{max} = 1$ bar) we used a custom actuation unit made of several pneumatic pressure regulators controlled with ArduinoDue (Figure 2B).

The robot is mounted on an aluminium frame in a vertical downward position to reduce the effect of the gravitational force and due to weight concerns, the two modules have different sizes. The two modules are connected together, thanks to a plexiglass interface, with an offset of 60 °. The fixed-end of the robot is 1000 mm distant from the ground in which we placed the target boxes(Figure 2A). Each box is geometrically equivalent to a square prism, whose dimensions are $[140 \times 140 \times 100]$ mm.

In the experiments, we toss in the target boxes with different objects (Figure 2C) whose mass and dimensions are shown in Table 1.

Table 1: Tossed objects. Physical characteristics of the object used during the experiments.

Object	**Characteristic dimension [mm]**	**Mass [g]**
Ping-pong ball	36	0.8
Lemon (toy)	62	4.8
Marker	26	3.2
Tomato (toy)	63	10.2

To hold an object during the linear trajectory we equipped our robot with a two-finger gripper. Each finger is made of 3D-printed surfaces joined together with thermoplastic polyurethane (TPU) chambers. We characterised the gripper's opening time. To do that we perform fifty closing-and-opening cycles where we measure the time interval from the instant in which the command is sent and the one in which the distance between the two fingers in ~ 60 mm (approximately the dimension of the biggest object taken into account). We assessed that the opening time of the gripper is equal to $\sim 0.2 \pm 0.01$ s.

To detect the position of the sections of interest we used a vision-based motion capture system (VICON Motion Capture Ltd) thanks to which we were able to track the movement of the object and the robot.

2.3 Dataset collection

The first phase of our approach is the collection of the dataset. We decided to have a linear movement and to divide it into two different phases: 1) Run-up; and 2) Forward, as in (Braun et al., 2012). This division aims to increase the speed of the robot in the overall movement which results in an increased throwing capacity.

To respect the counter-movement condition that we applied to the trajectory we realised a geometrical model of the throwing trajectory. Let us take as an example the model illustrated in Figure 3A. We can identify six bending directions associated with the different chambers of the robot (1, 2, 3 for the proximal module and 4, 5, 6 for the distal one), and let us identify a local reference frame (*xy*). In relation to its working principle, if we inflate chamber 1 of the soft robot it will bend along the negative y-axis. To avoid interference we need to prevent the simultaneous inflation of the chamber placed in the same direction, such as 3 and 4.

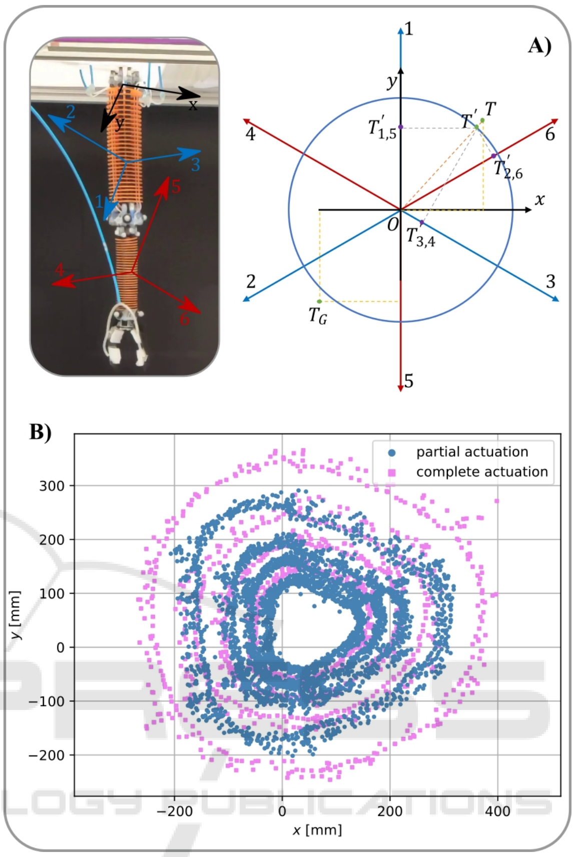

Figure 3: Training datasets. A) On the left, the identification of the major bending direction while on the right, the geometrical model of the throwing trajectory. B) represents the dataset used to train the neural networks for the controller in the two actuation scenarios analysed, respectively the partial and complete. These datasets have been obtained by averaging the dataset collected with the following objects: 1) ping pong ball; 2) vicon marker; 3) lemon (toy).

By knowing the coordinates of the goal point $T_G(x_{T_G}; y_{T_G})$, we can find the throwing direction that enables us to toss an object in that direction, which is represented by the line that passes for this point and the origin of the reference system. Instead, to find an approximative relationship between the actuations, we first took the desired point T_G, and we identified the objective point $T(x_T; y_T)$ by taking the symmetric of the goal point to the origin of the reference system, in other words, $T = (-y_{T_G}; -x_{T_G})$. Then, we projected the objective point on the circle of radius equal to one centred in the centre of the reference system. To obtain the relationships and conditions presented in Table 2, starting from the previously found point $T'(x_{T'}; y_{T'})$, we calculated the projection of it on the six bending axes. The length of the projection is proportional to the pressure we have to send to the actuator.

Table 2: Input patterns generation. To enable the movement of the I-Support robotic arm to respect the countermovement constraint, the actuation inputs (i.e., pressure signals) are adjusted based on the geometric relationship. Specifically, the point T_G, representing the desired goal point, is symmetrically reflected across the origin of the axes to obtain a new point T. This new point is projected onto the circumference of a unit radius (T').

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

We imposed that the first phase has to last $t_{run-up} = 0.50$ s while the overall movement $t_{traj} = 2.00$ s. The object is manually placed between the gripper fingers and then it is released at $t_{eject} = 0.97$ s, i.e., approximately the time instant in which the robot passes through the lowest point of the trajectory after the run-up phase. We collected more than one thousand trajectories by randomly sampling the point P in the xy plane. The overall process lasted two hours and fifteen minutes. For each trajectory, we saved the simulated landing position and the actuation pattern sent to the robot. To obtain the landing position coordinates, starting from the positions and velocities given by the motion tracking system we simulated the throws with the projectile equations.

We repeated this process twice to collect the data in the two actuation scenarios analysed in this work: 1) Partial, in which the distal module, i.e., the one on which is attached the gripper, is passive; 2) Complete, where all the six chambers are combined. In each case we collected the dataset with three different held objects the ping-pong ball, the marker and the lemon (toy) and then we average them to obtain the final dataset. The datasets are averaged to develop a controller that is not object-specific. The weight of the end effector impacts the dynamics of a robot, including its trajectories and controller performance during throwing tasks with various objects. This effect can be amplified when combined with a size-independent opening time. Figure 3B represents the average dataset in the partial and complete actuation scenario on which we trained, as shown in the next section, the neural networks.

2.4 Neural Network Training

With the datasets represented in Figure 3B, with a feed-forward neural network we mapped the landing positions with the actuation pattern associated with them, as in (1).

$$(\underline{\tau_{run-up}}, \underline{\tau_{forward}}) = f(\underline{x_{LP}}) \tag{1}$$

Where the first term is the entire actuation input composed of the run-up phase and the forward one.

The model is represented by an artificial neural network (ANN) with one hidden layer. We performed a model selection based on the average error on the actuation components to choose the best combination of hyperparameters. We have changed the number of units of the hidden layer, their activation function, and the type of input normalisation. Default values have been used for the different hyperparameters collected in Table 3. In addition, to expedite the learning process, the early stopping method has been considered.

In both cases, the input layer presents three neurons, i.e., the coordinates of the landing position, while the output layer depends on the actuation scenario. Indeed, we have six units and twelve units respectively with partial and complete actuation. In both cases, the output layer implements the ReLU activation function because the predictions of the ANN are pressure values. The results of the model selection are reported in Table 3.

3 RESULTS

To test our controllers, the robot performed throws towards ten target boxes of objects different in size and weight Figure 2A. We selected the box positions to test the controller in different directions. We performed our tests in two scenarios: 1) partial actuation, in which just the module attached to the frame is actuated, and 2) complete actuation, where all the chambers of the bi-modular robot can be inflated. In our experiments we decided to perform two types of tests: 1) quantitative ones, in which we tracked, thanks to the motion capture system, the movement of the object, i.e., its trajectory; and 2) qualitative tests (shown in Figure 4), in which we recorded the movement of the overall system to assess if the task was successful or not. We have success if the object fell inside the target box, with or without bouncing on the box borders. In both types of tests, we analysed the ability of the robot to generate actuation patterns to allow the soft robot to toss the object into the desired target. We performed three trials for each combination of parameters (object type, qualitative/quantitative test, target

Table 3: Neural network training. In this table, we collected the different hyperparameters that we changed and keep fixed during the training of the inverse model of the task, in the two distinguished actuation scenarios.

Default values			
learning rate	0.001	Optimiser	Adam
batch's size	16	Loss function	MSE
number of epochs	4000	Training/test partition	0.90

Best combination of the changed parameters		
Hyperparameter	**Partial Actuation**	**Complete actuation**
Number of units	64	128
Activation function	ReLU	tanh
Normalisation	Z-score	Z-score

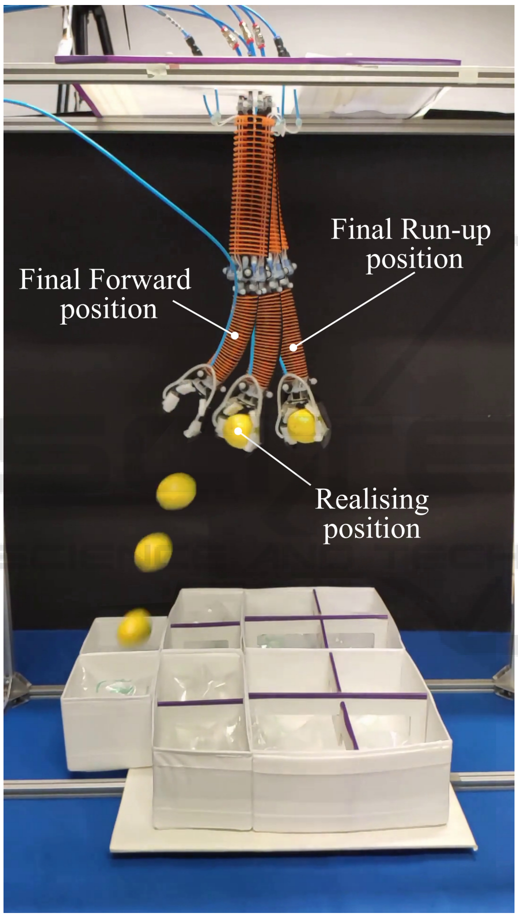

Figure 4: An example of the qualitative experiment. A throw toward the target box "I" with the Lemon (Toy) in the complete actuation scenario.

position), that led to 240 throws for each actuation modality (partial and complete). The results of the qualitative and quantitative tests are summarised respectively in Figure 5 and Figure 6. Here we have an idea of the performance of the controllers in the two actuation scenarios. Figure 5 is obtained by analysing the recorded trials to check if the trials were successful or not (object in/object out). We defined *success rate* as the ratio between the number of successful throws and the total number of throws performed. This index is equal to 55.83 % for the complete actuation modality and 68.33 % for the other.

The boxplots in Figure 6 illustrate the performance of the controllers based on the quantitative tests. These boxplots represent the distribution of errors, specifically the Euclidean distance between the desired target (the centre of the box) and the actual landing position. The tests were conducted across three trials for each combination of object targets.

Based on the results shown in the boxplots, the average error for the partial actuation scenario is 61.34 mm, while for the complete actuation scenario, the average error is 80.87 mm. These values indicate the average discrepancy between the desired and actual landing positions, providing a quantitative measure of the performance of the controllers.

4 DISCUSSION AND CONCLUSION

In this work, we show that despite the lower dynamic and a more challenging control than traditional robotics, a soft robot can perform a task as complex as tossing an object inside different target boxes.

Our results show that the neural network-based controller reaches a performance in successfully tossing objects to target locations, as high as $\sim 56 \div 68$ % on average (depending on the actuation pattern used). These results are highly influenced by the impossibility of the controller to generate appropriate actuation patterns for some targets. Indeed, if we describe the performance of the controller based on the median, which is less sensitive to the outliers than the average, we will obtain for the complete and partial actuation scenario an average error of 65.69 mm and 54.44 mm respectively.

These values are comparable to those found in the state-of-the-art. In their publication, Bianchi et al. (Bianchi et al., 2023) utilized a real-time open-

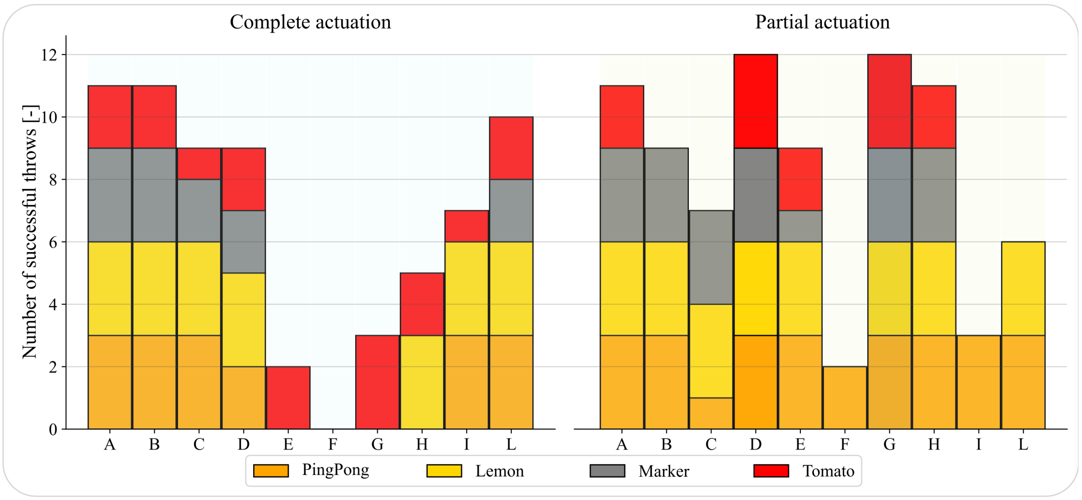

Figure 5: Qualitative results for the neural network-based controllers. The results are shown with respect to the object and each column is proportional to the successful trials. The maximum number of throws for each target is 12, as three attempts are made for each object.

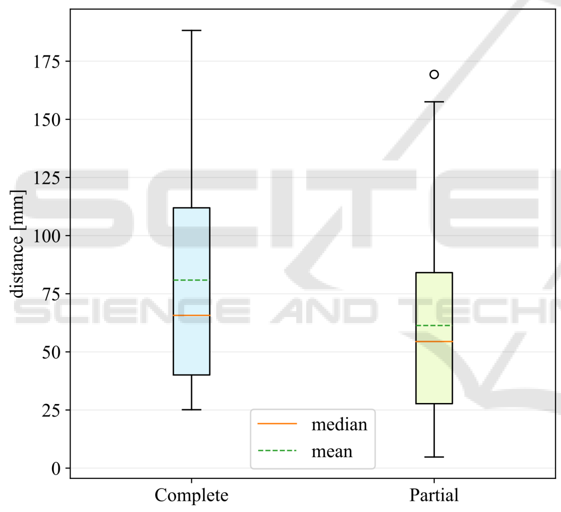

Figure 6: Comparison between the controller in different actuation scenarios based on the distance from the target position, which was reconstructed by recording the falling trajectories of the object during a toss with the Motion Capture System.

loop controller for a bi-modular robot based on a deep reinforcement learning (RL) method. The RL-based controller was tested in both partial and complete actuation scenarios, and the authors achieved a success rate of approximately $\sim 62 \div 63$ %. Compared to the current state of the art, this study presents a controller that exhibits similar real-time characteristics and comparable or slightly better performance in certain instances. Significantly, the inverse dynamic method requires only a fraction of the time needed to train the neural network compared to the training time of the RL agent. However, the method described in (Bianchi et al., 2023) shows the same performance in both actuation modalities.

From Figures 5 and 6, the difference between the performance in the two scenarios is clear. . The neural network in the complete actuation scenario has a poor ability to generate appropriate patterns due to the ill-posed problem. In this case, we are asking the network, given three input values (the desired landing coordinates), to predict the twelve values to perform both the run-up and forward phases. On the contrary, the problem is simpler in the case where the robot has a passive module, as we will have only six values to predict.

However, the performance of the controllers can be improved by acting in several levels of the proposed approach. Firstly, we could improve the collection of the data by trying to pass from a geometrical model of the throwing trajectory (equations collected in Table 2) to a model-free approach or in general a more detailed model. Additionally, in every trial, the object can have different initial conditions since we manually place it which increases the variability of our experimental data.

The gripper has a remarkable influence on the overall experiments because it influences the initial kinematic conditions of the object in its free-fall forward target box. In our experiments, we determined an opening time based on the biggest object that we planned to toss and we consider that in our throwing task specification. However, the objects are considerably different from each other, so the small objects were released before the ideal realising instant. That led to an increased variability of the data and a lower accuracy of the controller.

ACKNOWLEDGMENT

We acknowledge Andrea Centurelli for the help provided during the development of the proposed approach.

This work was partially supported by the European Union's Horizon 2020 FET-Open program under grant agreement no. 863212 (PROBOSCIS project).

REFERENCES

Alessi, C., Falotico, E., and Lucantonio, A. (2023). Ablation study of a dynamic model for a 3d-printed pneumatic soft robotic arm. *IEEE Access*, 11:37840–37853.

Bianchi, D., Antonelli, M., Laschi, C., and Falotico, E. (2022). Open-loop control of a soft arm in throwing tasks. In *19th International Conference on Informatics in Control, Automation and Robotics*, pages 138–145. ISSN: 2184-2809.

Bianchi, D., Antonelli, M. G., Laschi, C., Sabatini, A. M., and Falotico, E. (2023). SofToss: Learning to Throw Objects with a soft robot. *IEEE Robotics and Automation Magazine*.

Braun, D. J., Howard, M., and Vijayakumar, S. (2012). Exploiting variable stiffness in explosive movement tasks. *Robotics: Science and Systems VII*, 7:25–32.

Büchler, D., Calandra, R., and Peters, J. (2022). Learning to Control Highly Accelerated Ballistic Movements on Muscular Robots. *Robotics and Autonomous Systems*, page 104230.

Centurelli, A., Arleo, L., Rizzo, A., Tolu, S., Laschi, C., and Falotico, E. (2022). Closed-loop Dynamic Control of a Soft Manipulator using Deep Reinforcement Learning. *IEEE Robotics and Automation Letters*, pages 1–1. Conference Name: IEEE Robotics and Automation Letters.

Centurelli, A., Rizzo, A., Tolu, S., and Falotico, E. (2021). Open-loop Model-free Dynamic Control of a Soft Manipulator for Tracking Tasks. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 128–133.

Fang, Z., Hou, Y., and Li, J. (2021). A pick-and-throw method for enhancing robotic sorting ability via deep reinforcement learning. In *2021 36th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 479–484.

Fischer, O., Toshimitsu, Y., Kazemipour, A., and Katzschmann, R. K. (2022). Dynamic Task Space Control Enables Soft Manipulators to Perform Real-World Tasks. *Advanced Intelligent Systems*, page 2200024.

Gazzola, M., Dudte, L. H., McCormick, A. G., and Mahadevan, L. (2018). Forward and inverse problems in the mechanics of soft filaments. *Royal Society Open Science*, 5(6):171628. Publisher: Royal Society.

George Thuruthel, T., Falotico, E., Manti, M., Pratesi, A., Cianchetti, M., and Laschi, C. (2017). Learning closed loop kinematic controllers for continuum manipulators in unstructured environments. *Soft robotics*, 4(3):285–296.

Giorelli, M., Renda, F., Ferri, G., and Laschi, C. (2013). A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5033–5039. ISSN: 2153-0866.

Hawkes, E. W., Blumenschein, L. H., Greer, J. D., and Okamura, A. M. (2017). A soft robot that navigates its environment through growth. *Science Robotics*, 2(8):eaan3028. Publisher: American Association for the Advancement of Science.

Holland, D. P., Abah, C., Velasco-Enriquez, M., Herman, M., Bennett, G. J., Vela, E. A., and Walsh, C. J. (2017). The Soft Robotics Toolkit: Strategies for Overcoming Obstacles to the Wide Dissemination of Soft-Robotic Hardware. *IEEE Robotics & Automation Magazine*, 24(1):57–64. Conference Name: IEEE Robotics & Automation Magazine.

Ilievski, F., Mazzeo, A. D., Shepherd, R. F., Chen, X., and Whitesides, G. M. (2011). Soft Robotics for Chemists. *Angewandte Chemie International Edition*, 50(8):1890–1895.

Katzschmann, R. K., DelPreto, J., MacCurdy, R., and Rus, D. (2018). Exploration of underwater life with an acoustically controlled soft robotic fish. *Science Robotics*, 3(16):eaar3449. Publisher: American Association for the Advancement of Science.

Laschi, C., Mazzolai, B., and Cianchetti, M. (2016). Soft robotics: Technologies and systems pushing the boundaries of robot abilities. *Science Robotics*, 1(1):eaah3690. Publisher: American Association for the Advancement of Science.

Laschi, C., Thuruthel, T. G., Lida, F., Merzouki, R., and Falotico, E. (2023). Learning-based control strategies for soft robots: Theory, achievements, and future challenges. *IEEE Control Systems*, 43(3):100 – 113.

Li, S., Vogt, D. M., Rus, D., and Wood, R. J. (2017). Fluid-driven origami-inspired artificial muscles. *Proceedings of the National Academy of Sciences*, 114(50):13132–13137. Publisher: Proceedings of the National Academy of Sciences.

Manti, M., Pratesi, A., Falotico, E., Cianchetti, M., and Laschi, C. (2016). Soft assistive robot for personal care of elderly people. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 833–838. ISSN: 2155-1782.

Piqué, F., Kalidindi, H. T., Fruzzetti, L., Laschi, C., Menciassi, A., and Falotico, E. (2022). Controlling Soft Robotic Arms Using Continual Learning. *IEEE Robotics and Automation Letters*, 7(2):5469–5476. Conference Name: IEEE Robotics and Automation Letters.

Polygerinos, P., Correll, N., Morin, S. A., Mosadegh, B., Onal, C. D., Petersen, K., Cianchetti, M., Tolley, M. T., and Shepherd, R. F. (2017). Soft Robotics: Review of Fluid-Driven Intrinsically Soft Devices; Manufacturing, Sensing, Control, and Applications in Human-Robot Interaction. *Advanced Engineering Materials*, 19(12).

Raptopoulos, F., Koskinopoulou, M., and Maniadakis, M. (2020). Robotic Pick-and-Toss Facilitates Urban Waste Sorting *. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1149–1154. ISSN: 2161-8089.

Runge, G. and Raatz, A. (2017). A framework for the automated design and modelling of soft robotic systems. *CIRP Annals*, 66(1):9–12.

Rus, D. and Tolley, M. (2015). Design, fabrication and control of soft robots. *Nature*, 521:467–75.

Thuruthel, T. G., Falotico, E., Renda, F., Flash, T., and Laschi, C. (2019). Emergence of behavior through morphology: a case study on an octopus inspired manipulator. *Bioinspiration & biomimetics*, 14(3):034001.

Vannucci, L., Cauli, N., Falotico, E., Bernardino, A., and Laschi, C. (2014). Adaptive visual pursuit involving eye-head coordination and prediction of the target motion. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 541–546.

Vannucci, L., Falotico, E., Di Lecce, N., Dario, P., and Laschi, C. (2015). Integrating feedback and predictive control in a bio-inspired model of visual pursuit implemented on a humanoid robot. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9222:256–267.

Wang, J. and Chortos, A. (2022). Control Strategies for Soft Robot Systems. *Advanced Intelligent Systems*, n/a(n/a):2100165.

Webster, R. J. and Jones, B. A. (2010). Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review. *The International Journal of Robotics Research*, 29(13):1661–1683. Publisher: SAGE Publications Ltd STM.

Zeng, A., Song, S., Lee, J., Rodriguez, A., and Funkhouser, T. (2019). TossingBot: Learning to throw arbitrary objects with residual physics. *Proceedings of Robotics: Science and Systems (RSS)*.