2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

DynDLO: Learning-Based Trajectory Planning for Dynamic Robotic Manipulation of Deformable Linear Objects

Daniele Maria Liuni[1*], Alessandro Bartesaghi[1*], Andrea Monguzzi[2], Alessandra Miuccio[3]
Andrea Maria Zanchettin[1], Paolo Rocco[1]

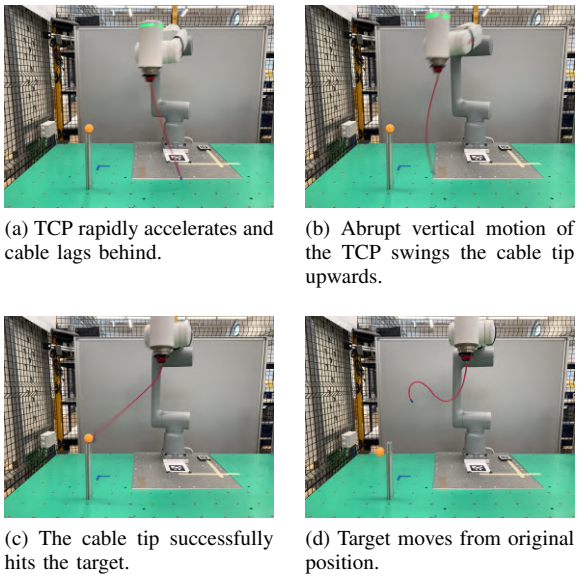
***Abstract*— The automatic manipulation of Deformable Linear Objects (DLOs) remains currently a challenge in robotics. Previous research on robotic DLOs manipulation has primarily addressed quasi-static DLO manipulation at low speeds, leaving the potential of dynamic DLO manipulation largely unexplored. This paper introduces DynDLO, a goal-conditioned, 6-axis robot-independent Reinforcement Learning sandbox for training agents on a variety of DLO dynamic manipulation tasks. In DynDLO, a DLO attached to the robot Tool Center Point (TCP) is simulated in the MuJoCo environment. By employing a B-Spline based trajectory generation function, the agent is capable of learning single and multiple step trajectories for the TCP, which succeed in various DLO dynamic manipulation problems. Specifically, we propose tailored design strategies for the reward function according to the classification of tasks into implicit or explicit DLO shape control tasks. Experiments on four representative tasks demonstrate that DynDLO is capable of generating dynamic manipulation policies that transfer successfully from simulation to the real world, achieving high success rates without requiring real-world training.**

Link to the video: https://youtu.be/-W3tKXyenO4
Link to GitHub page[1]

I. Introduction

In recent years, robotic research has directed its attention to the manipulation of deformable objects and, in particular, to Deformable Linear Objects (DLOs), elements such as cables, wires and hoses. Even though many studies have been conducted on the manipulation of DLOs for different applications such as shape control [1], [2], motion planning for obstacle avoidance [3], [4], and environmental contact exploitation [5], most of them rely on the assumption of quasi-static manipulation. In these works, task execution is constrained to low velocities, ensuring that dynamic effects of the DLO (such as oscillations and whipping) remain negligible. The DLO dynamic manipulation (i.e. involving fast motions) remains an under-explored problem, where learning-based approaches offer significant advantages over classical model-based methods, which are often hindered by model complexity and high computational cost [6]. Among all the possible learning methodologies, this work focuses on Reinforcement Learning (RL), which is preferred over Imitation Learning due to its ability to obtain optimal policies for tasks that are challenging even for humans.

More in detail, this work introduces DynDLO, a RL sandbox designed to train agents in planning Cartesian-space trajectories for a variety of dynamic manipulation tasks involving a DLO characterized by low stiffness. Fig. 1 shows an example of a RL agent trained with DynDLO executing a dynamical DLO tip control task to hit a target. In DynDLO, the agent is represented by the robot gripper Tool Center Point (TCP), which can hold either one end of the DLO (single-arm configuration) or both ends (dual-arm configuration). This formulation enables the learning of policies that generate Cartesian trajectories in a robot-



(a) TCP rapidly accelerates and cable lags behind.
(b) Abrupt vertical motion of the TCP swings the cable tip upwards.
(c) The cable tip successfully hits the target.
(d) Target moves from original position.

Fig. 1: Robotic arm executing a policy learned in DynDLO for a dynamic Tip Control Task, accurately hitting the target. Snapshots of the learned and executed Cartesian trajectory are shown, each accompanied by a brief description.

[1]The Authors are with Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza Leonardo da Vinci 32, 20133, Milan (Italy), email: {danielemaria.liuni, alessandro.bartesaghi, andreamaria.zanchettin, paolo.rocco}@polimi.it
[2]Andrea Monguzzi is with Politecnico di Milano and Leonardo, Innovation Labs, Robotics, Via Raffaele Pieragostini 80, 16151, Genoa (Italy). e-mail: andrea.monguzzi@leonardo.com. The research presented herein was produced externally to Leonardo, while this author was enrolled at Politecnico di Milano. The contribution does not include/use: data/information/schemes/... nor references to: programs/projects/contracts/... owned by Leonardo or involving it.
[3]Alessandra Miuccio is with Royal Military Academy, RAS Lab, Av. de la Renaissance 30, 1000 Bruxelles (Belgium). e-mail: alessandra.miuccio@mil.be The research presented herein was carried out while this author was employed at Politecnico di Milano.
*authors with equal contribution
[1]https://github.com/MerlinLaboratory/DynDLO_Learning_Based_Trajectory_Planning_for_Dynamic_Robotic_Manipulation_of_DLOs

independent way for any 6-axis manipulator. In particular, as detailed in the following sections, we define specific reward terms and introduce an inverse kinematics strategy to prevent singularities during real-world trajectory execution.

Following the task taxonomy introduced in [7], we differentiate between explicit shape control tasks (e.g., DLO oscillation minimization) and implicit shape control tasks (e.g., tip control, vaulting, hooking). Explicit shape control aims to bring the DLO to a desired shape, while implicit shape control focuses on achieving a target configuration that inherently requires the DLO to deform. Based on this classification, we propose tailored reward design strategies within DynDLO for each task type.

As we show in this work, DynDLO can generate dynamic manipulation policies that transfer successfully from simulation to the real world, without requiring real-world training. Note that training in simulation makes the problem robot-independent, and avoids real-world learning issues, such as potential robot wear and damage during exploration, time-intensive experiments and measurement uncertainties. Moreover, real-world training for DLO dynamic manipulation would require a precise and high-frequency state estimation strategy, which remains an open challenge [8].

Summarising, the main contributions of this paper are the following ones:

1) The introduction of a 6-axis robot-independent RL sandbox capable of training agents for both implicit and explicit DLO shape control tasks;

2) The definition of a B-spline–based parametrization strategy for trajectory generation that leverages position and time increments to provide a simple, low-dimensional action space A, while still enabling the definition of complex trajectories $\mathbf{p}(t) \in C^2$. This formulation supports both single-step and multi-step episode policies.

3) The definition of a strategy for the design of the reward function according to the taxonomy of tasks into implicit or explicit DLO shape control tasks.

The remainder of this work is organized as follows. Section II presents the state of the art regarding DLO dynamic manipulation. In Section III, the DynDLO sandbox is described and its main features are detailed. In Section IV, the main differences between single and multiple steps episode policies are highlighted. Section V presents the obtained results both in simulation and real world. Finally, conclusions are drawn in Section VI, and hints for future works are provided.

II. Related Works

In the context of DLOs dynamic manipulation, there are works that rely on dynamical models of the cable to design control strategies. For example, the DLO model detailed by [18] is exploited in [9] and [10] aiming to damp cable oscillations using a fuzzy controller to perform an optimal lifting action and a Sliding Mode Controller, respectively. Authors of [11], [12] used flatness-based control to make one end of a DLO follow a specified path. However, the considered strategies are complex, and their validation is provided only in simulation.

Nah *et al.* [13], used the DIRECT-L optimization algorithm [19] to learn the joint trajectories for a 4-degrees-of-freedom (DoF) robot which manipulated a whip to strike a target with its tip by leveraging a simplified whip model with 50 DoF.

Recent works on dynamic manipulation of DLOs concentrate instead on model-free and data-driven techniques that allow a faster computation of the control action, enabling online re-planning, but usually require an extensive initial training phase where the control policy has to learn how the manipulated DLO interacts with the environment. For example, Chi *et al.* [14] used a neural network which iteratively adjusted the trajectory for a robot to make a whip hit a target, while in [15], a policy was trained to select the apex point for an arcing motion which allowed to manipulate a DLO to vault dynamically over obstacles and to complete other tasks.

In this setting, approaches for DLO dynamic manipulation based on Reinforcement Learning (RL) are particularly interesting as they allow to train policies in simulation, which can then be deployed to real hardware. For example, Chen *et al.* [16] used a RL agent trained only using the MuJoCo [20] simulation engine to learn the apex point for a Cartesian robot trajectory which, after execution, resulted in a wire to be precisely flung between two obstacles. Real-world experiments demonstrated that the policy trained only in simulation can be directly applied to a real robot without further training. Zhaole *et al.* [17] developed DexDLO, a goal-conditioned RL simulation framework that, together with a pose regularized reward function structure, was capable of training policies to control a Shadow Hand to complete various DLO dexterous manipulation tasks.

Inspired by [16], [17], this paper introduces DynDLO, a 6-axis robot-independent RL sandbox which allows to train agents capable of devising smooth Cartesian reference trajectories. Once executed, these trajectories allow to complete various DLO dynamic manipulation tasks. Within this framework, it is possible to easily switch between implicit and explicit shape control tasks and a trajectory parametrization strategy allowing the generation of complex trajectories with a low dimensional action space. This reduces the learning time and makes DynDLO more flexible than [15], [16].

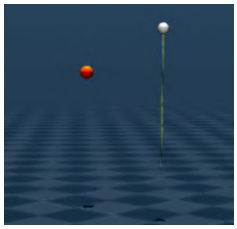
Moreover, DynDLO is able to train agents both in a single step episode fashion, like [15], [16], and in a multiple step one, like [17]. Single step approaches are particularly useful for sim-to-real transfer evaluation as single-step episode agents only need an initial observation of the DLO state to devise a trajectory. Finally, RL trained policies are evaluated in a real experimental setup to assess their sim-to-real transfer performance, whereas [17] only presents arguments on how their approach might help bridge the sim-to-real gap. Table I summarizes the main difference of the proposed methodology involving our framework, DynDLO, with respect to the most relevant works in the literature.

III. DynDLO Framework

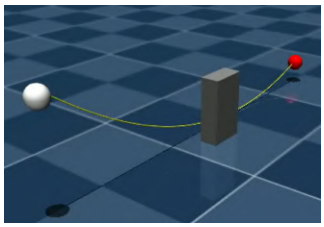
In a RL framework, the agent selects an action $a(t) \in A$, where A is the action space, to bring an *environment* from

Work	Strategy	Innovation	Drawbacks	Validation	Robot
[9]	Fuzzy controller for lifting action	Optimal lifting for damping vibrations	Complex, planar and low damping effect	Simulation	TCP
[10]	Sliding Mode Control	Robust and general strategy for damping	Complex, planar	Simulation	TCP
[11], [12]	Flatness-based control	Cable end point position under control	Complex and not extendable to dual-arm case	Simulation	TCP
[13]	DIRECT-L algorithm	Smooth joint motions capable tonachieve dynamic manipulation	Only single step solutions, high computational cost	Simulation	4 DOF
[14]	NN-Delta Dynamics	Adaptability to environment changes	Single-task oriented and not single-shot	Real World	Sim: TCP(s); Real: Single Arm
[15]	RL in real world	No model, high success in reality	Time consuming learning phase	Real World	Sim: //; Real: Single arm
[16]	RL in simulation	Fast learning and direct transfer	Predefined trajectory shape	Real World	Sim: Single Arm; Real: Single Arm
[17]	RL in simulation	General purpose reward function	Observation redundancy	Simulation	Hand
DynDLO	RL in simulation	6-axis robot-independent general-task sandbox	No robot bulk and joint limits considered	Real World	Sim: TCP(s); Real: Single Arm

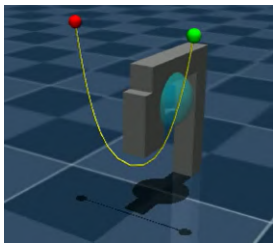
TABLE I: Summary of the comparison of DynDLO with relevant existing methodologies.



(a) Tip Control.



(b) Vaulting.



(c) Hooking.

Fig. 2: TCP(s) and DLO in the DynDLO environments in MuJoCo for single and dual arm tasks.

the state *s(t)* to the state *s(t+1)* and receives from it the observation at the next time step $o(t+1) \in O$, where O is the observation space, and a *reward* $r(t) \in \mathbb{R}$. The objective is to maximize the cumulative reward over time. In this Section, the main elements composing the proposed DynDLO sandbox are defined and accurately described according to the several considered tasks.

A. Environment

Training of RL agents in real-world environments suffers from data inefficiency, slowness and risk of hardware wear. In order to overcome these issues, we create a simulated environment in MuJoCo [20]. As Fig. 2 shows, the environment presents two main components: the gripper TCP and the cable. The robot is not introduced on purpose, in order for the learnt policy to be independent of the type of the 6-axis robot and to reduce the computational cost caused by calculating inverse kinematics for each point of the Cartesian trajectory. Feasibility of the learned trajectory when deployed on the real robot is achieved by imposing specific reward terms and adopting a specific inverse kinematics solver (see Sections III-D and V).

The robot's TCP is represented as a sphere to which the cable is rigidly attached. This simulated gripper has 6 DoFs, and is moved along the three Cartesian directions by using cascaded position-velocity control loops while its orientation is kept fixed. This choice was made in order to keep the dimensionality of A and O small, thereby reducing computational time, and because high performance is achieved by learning only Cartesian position trajectories (see Section V). Nevertheless, the framework can be extended to also control gripper rotation.

The cable is the most challenging component, and accurately modelling its behaviour is crucial to reducing the sim-to-real gap and ensuring real-world applicability. The DLO is modeled as in [16], using the Discrete Elastic Rods (DER) theory. Particularly, the DLO is built as a chain of rigid bodies connected through spherical joints, meaning that this modelling strategy captures bending and torsional strains, which are fundamental for dynamic manipulation, but does not account for axial strain, as its impact on dynamic manipulation is negligible. This model requires selecting an appropriate number of elements to minimize the reality gap while avoiding excessive computation. This number is set on the basis of cable length: for a $50\,cm$ cable it is equal to $n_{bodies} = 20$, while it is $n_{bodies} = 60$ in case of $90\,cm$ long cable.

The scenario described above can be easily extended to the dual arm case by introducing a second gripper holding the other cable end, as shown in Fig. 2.

Four different tasks are used as benchmarks for DynDLO:

I) **Tip Control** (implicit shape control): the manipulator firmly grasps one end of a 50 cm long DLO and has to whip the cable to make its tip hit a spherical target of diameter $D_{target} = 5$ cm placed at a point $\mathbf{P}^*$, as shown in Fig. 2a;

II) **Vaulting** (implicit shape control): one end of the cable is fixed to the manipulator's TCP while the other one is attached to a support. The robot has to make a 90 cm cable vault over a static obstacle (a box of dimensions $4 \times 8 \times 14$ cm) to make the DLO middle section land on the other side, as illustrated in Fig. 2b;

III) **Hooking** (implicit shape control): the DLO is firmly grasped at its extremities by the grippers of a dual-arm manipulator. The goal of this task is to swing a $50\,cm$ cable in order to attach it to a static hook (see Fig. 2c);

IV) **Time and Oscillations Minimization** (explicit shape control): it is an optimization problem which requires to minimize simultaneously the trajectory time and the oscillations of a $50\,cm$ cable after the motion, which arise at high speed due to inertia. It can be seen as an explicit shape control task, where the objective is to have a straight cable at motion end.

B. Observation Space

The structure of the observation space is the same for all the proposed tasks used to benchmark the DynDLO framework and it is reported in Table II.

To further reduce observation space complexity, only the 3D position of the center of mass of a specific body composing the DLO chain, named $\mathbf{P}_{DLO}$, is given as observation, while its velocity is neglected based on what stated in [17].

Observation	Dim	Domain
Gripper Cartesian position, $\mathbf{P}$	$3 \times n_g$	$\mathbb{R}^3$
Gripper Linear velocity, $\dot{\mathbf{P}}$	$3 \times n_g$	$\mathbb{R}^3$
Position of the section of interest in the DLO, $\mathbf{P}_{DLO}$	3	$\mathbb{R}^3$
Target/Obstacle position, $\mathbf{P}^*$	3	$\mathbb{R}^3$

TABLE II: DynDLO observation space: n_g is the number of simulated grippers in the environment.

C. Trajectory Generation and Action Space

The goal of DynDLO is to train RL agents to learn suitable smooth Cartesian trajectories for the robot's end-effector to accomplish various DLO dynamic manipulation tasks. To achieve this, we employ a Trajectory Generation Function which uses Cubic Cardinal B-Spline interpolation to define the path $\mathbf{p}(u) : \mathbb{R} \to \mathbb{R}^3$ as:

$$\mathbf{p}(u) = \begin{bmatrix} x_d(u) \\ y_d(u) \\ z_d(u) \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{l} \alpha_i B^x_{i,3,\mathbf{u}}(u) \\ \sum_{i=0}^{l} \beta_i B^y_{i,3,\mathbf{u}}(u) \\ \sum_{i=0}^{l} \gamma_i B^z_{i,3,\mathbf{u}}(u) \end{bmatrix} \tag{1}$$

where $B_{i,3,\mathbf{u}}$ denotes the i-th B-Spline of degree three derived using the knot vector $\mathbf{u} = \{u_i\}_{i=0}^{n-1} = \{u_0 \leq u_1 \leq ... \leq u_{n-1} \wedge u_i - u_{i-1} = 1\}, n \in \mathbb{N}$. This results in a geometric curve parametrized in the knot variable u. As u takes values between u_0 and u_{n-1}, all the points on the geometric path can be obtained. Therefore, by making u dependent on the time variable t, we are able to construct a timing law for the motion on the path specified by (1). To do so, cubic spline interpolation is used to find the minimum curvature interpolating function for the set of points $\{(t_0, u_0), (t_1, u_1), ..., (t_{n-1}, u_{n-1})\}$. This way, the timing law $u(t) : \mathbb{R}^+ \to \mathbb{R}$ can be built. The desired Cartesian trajectory is then simply computed as $\mathbf{p}(t) = \mathbf{p}(u(t))$. For the trajectory generation function to properly work, we need a way to map agent actions to key points in $\mathbb{R}^3$, which are used to create the geometric path and to the specific time instants at which each of these points need to be reached to create the timing law. Since the initial position of the gripper $\mathbf{P}_0$ is taken as observation at each RL step, which is assumed to start at an arbitrary time instant t_0, the problem reduces to finding $n-1$ points $\mathbf{P}_1, ..., \mathbf{P}_{n-1}$ through which the trajectory passes and the associated reaching time instants $t_1, ..., t_{n-1}$. Then, by exploiting the one-to-one mapping among the n knots $u_0, ..., u_{n-1}$ and the n points $\mathbf{P}_0, .., \mathbf{P}_{n-1}$, the trajectory can be assembled as explained. Since $\mathbf{P}_0$ and t_0 are known, each $\mathbf{P}_i, t_i$ $i = 1, ..., n-1$ can be computed recursively as in (2) and (3) respectively:

$$\mathbf{P}_i = \mathbf{P}_{i-1} + \Delta\mathbf{P}_{i,i-1} \quad i = 1, ..., n-1 \tag{2}$$

$$t_i = t_{i-1} + \Delta t_{i,i-1} \quad i = 1, ..., n-1 \tag{3}$$

where $\Delta\mathbf{P}_{i,i-1}$ and $\Delta t_{i,i-1}$, $i = 1, ...n-1$ are the $n-1$ offsets and time intervals between two consecutive points respectively. Since agent actions are used to compute a trajectory, it is convenient to take the quantities $\Delta\mathbf{P}_{i,i-1}$ and $\Delta t_{i,i-1}$ described in (2) and (3) to define the action space for DynDLO, which is reported in Table III.

Action	Dim	Limits
Longitudinal distance $\Delta x_{i,i-1}$	$n-1$	$[\Delta x_{min}, \Delta x_{max}]$ [m]
Lateral distance $\Delta y_{i,i-1}$	$n-1$	$[\Delta y_{min}, \Delta y_{max}]$ [m]
Vertical distance $\Delta z_{i,i-1}$	$n-1$	$[\Delta z_{min}, \Delta z_{max}]$ [m]
Time interval $\Delta t_{i,i-1}$	$n-1$	$[\Delta t_{min}, \Delta t_{max}]$ [s]

TABLE III: DynDLO action space. Notice that $\Delta\mathbf{P}_{i,i-1} = [\Delta x_{i,i-1}, \Delta y_{i,i-1}, \Delta z_{i,i-1}]^T$.

Notice that this choice of the action space implies that an agent action consists of $3 \times (n-1)$ spatial increments and $(n-1)$ time intervals. Therefore, a single-step agent can only choose a total of $4 \times (n-1)$ values, as only one action is permitted, while a multiple-step episode agent can select a larger number of values during an episode thanks to the re-planning procedure (see Section IV for further details). Taking as actions the offset between each point instead of directly the points themselves yields no advantage, but by choosing $\Delta t_{i,i-1}$ instead of directly t_i we avoid having to perform a sorting operation in the case when $t_{i-1} > t_i$ which would be seen as added complexity to the environment by the RL agent. Moreover, for the oscillation-avoidance task (task IV), the target point corresponds to the final trajectory point. It is added to the trajectory and the time interval dimension is increased by one.

In dual arm cases, the action space is the same since grippers are assumed to move symmetrically.

	Task I	Task II	Task III	Task IV
I_{avoid}	$\|\mathbf{P}\| > outreach \vee \|\dot{\mathbf{P}}\| > v_{max} \vee collision^{\text{i}} \vee \|\mathbf{P}\|_{xy} < \varepsilon_{shoulder}$ [ii]			
I_{dist}	$\|\mathbf{P}^* - \mathbf{P}_{DLO}\| > \varepsilon_{dist}$			//
$I_{success}$	$\|\mathbf{P}^* - \mathbf{P}_{DLO}\| < \varepsilon_{goal}$			$arrived^{\text{iii}} \wedge \|\mathbf{P}^{eq}_{DLO} - \mathbf{P}_{DLO}\|_{xy}$ [iv] $< \varepsilon_{oscill}$ [v]
I_{dyn}	$\bar{\mathbf{z}}_P \geq M$ [vi]			$arrived \wedge \|\mathbf{P}^{eq}_{DLO} - \mathbf{P}_{DLO}\|_{xy} < \varepsilon_{oscill}$
J_{cost}	1			$\|\mathbf{P}^{eq}_{DLO} - \mathbf{P}_{DLO}\|_{xy}$

TABLE IV: Reward terms for each task. Notice that for tasks I-III, $J_{cost} = 1$ as the optimization problem for these tasks is simply reduced to a constraint satisfaction problem.

D. Reward Function Structure

The reward function is crucial in guiding the RL agent during the learning process. We define two reward functions capable of generalizing across multiple DLO dynamic implicit and explicit shape control tasks. This allows the use of the DynDLO sandbox to train policies for new and unforeseen tasks with little to no changes to the underlying structure. The two functions exhibit the same general structure:

$$\mathbf{r}_{tot} = \mathbf{r}_{avoid} + \mathbf{r}_{distance} + \mathbf{r}_{success} + \mathbf{r}_{dyn} \tag{4}$$

[i] $collision$ is equal to 1 if $\mathbf{z}_{P_{DLO}} < 1.25$ cm, 0 otherwise
[ii] $\varepsilon_{shoulder}$ = position threshold to avoid shoulder singularity
[iii] $arrived$ is equal to 1 if the target point is reached, 0 otherwise
[iv] $\|\mathbf{P}^{eq}_{DLO} - \mathbf{P}_{DLO}\|_{xy}$ = x-y distance of $\mathbf{P}_{DLO}$ from equilibrium
[v] ε_{oscill} = oscillation threshold used to speed up minimization
[vi] $\bar{\mathbf{z}}_P$ is the mean gripper height and M is a threshold

Each reward term relates to a different aspect and its definition changes depending on the type of task (implicit or explicit shape control). In the following, the definition of each term for both the shape control tasks is provided:

1) *Avoidance Reward* $\mathbf{r}_{avoid}$:

$$\mathbf{r}_{avoid} = \lambda_1 I_{avoid} \tag{5}$$

where $\lambda_1 < 0$.

This reward penalizes all the conditions which must be absolutely avoided: overcoming the maximum robot outreach $\mathbf{P}_{max}$ and its maximum TCP velocity v_{max}, the cable-floor contact condition and approach the origin of the scene in the x-y plane, where robot's base frame is located (introduced to avoid shoulder singularities). I_{avoid} is 1 when at least one of the conditions is true and zero otherwise. This term is used to derive a control policy which is feasible to transfer on a real manipulator and to avoid unnecessary exploration in case of undesired situations. This term is the same for both the functions;

2) *Distance Reward* $\mathbf{r}_{distance}$:

$$\mathbf{r}_{distance} = \begin{cases} \lambda_2 I_{dist} & \text{if implicit task} \\ \lambda_2 \sum \Delta t_{i,i-1} & \text{if explicit task} \end{cases}$$

with $\lambda_2 < 0$ and $\Delta t_{i,i-1}$ are the time intervals defined in Table III (definition of I_{dist} in Table IV) .

It is a term used to penalize the distance from the desired goal $\mathbf{P}^*$. For implicit control tasks, it is used to guide the agent towards the goal; for explicit control ones, it is used for time minimization;

3) *Success Reward* $\mathbf{r}_{success}$:

$$\mathbf{r}_{success} = \lambda_3 I_{success} \tag{6}$$

where $\lambda_3 > 0$.

This term rewards the agent with a bonus if the task is completed successfully. $I_{success}$ is 1 whenever the intended dynamic manipulation action is successfully completed and zero otherwise;

4) *Cable dynamics Reward* $\mathbf{r}_{dyn}$:

$$\mathbf{r}_{dyn} = \lambda_4 J_{cost} I_{dyn} \tag{7}$$

with $\lambda_4 < 0$ (J_{cost} and I_{dyn} specified in Table IV).

This term is related to the dynamics effects of the DLO. For task IV, it is proportional to oscillation amplitude, while for the other tasks it is used to enforce the agent to exploit cable dynamics.

In the end, it is essential to specify that, for implicit shape control tasks, all terms are evaluated only once at the episode's end, while they are evaluated at each step for the explicit shape control one, except for $r_{distance}$. In this way, terms cumulate over time and yield a reward gradient that assists the RL algorithm (see Section III-E).

Table IV summarizes the definition of each reward term parameter for the four considered tasks.

Properly tuning the weights λ_i is crucial for the agent to learn the desired task correctly. As a rule of thumb, one should choose $\lambda_3 \gg |\lambda_2|, |\lambda_4|$ so that if task success is achieved, the agent receives a total reward $\mathbf{r}_{tot} > 0$. In task IV, attention has to be paid in tuning λ_2 and λ_4 properly so that time minimization and oscillation suppression reward terms have the same weight during learning.

E. RL Algorithm selection

Depending on the type of task (implicit or explicit shape control), a different RL algorithm must be adopted. For explicit shape control tasks (e.g. tasks I, II, III), we use the Proximal Policy Optimization (PPO) algorithm implemented in StableBaselines3 [21] with default settings to train agents. In implicit shape control tasks, where it is required to optimize the way the goal is reached, the best result will be obtained by adopting a learning algorithm that is able to efficiently explore the environment to find the global optimum. For this reason, the Soft Actor-Critic (SAC) algorithm from StableBaselines3 [21] with default settings is selected for Task IV.

IV. Single and Multiple step Strategies

One of the main advantages of DynDLO is the possibility to select between single-step and multi-step episode strategy. Both of them present strong points and weaknesses.

A. Single step episode strategy

In this strategy, the agent provides the trajectory at the start of the episode, and it is nevermore called: it performs only one action and receives the cumulated reward at the episode end. The agent practically acts as a planner and the trajectory is performed in an open loop fashion, as in [16]. The strength of this method lies in the faster training with respect to the multiple step one and in its simplicity, since it requires only the DLO initial configuration and target/obstacle position. The main drawback lies in the open loop nature of the strategy, which prevents any recovery of the error given by the gap between the simulated and the real cable and makes the error accumulate over time. Improvements can be obtained by correctly identifying cable parameters that are crucial for dynamic manipulation in order to reduce the sim-to-real gap.

B. Multiple step episode strategy

The multiple step episode strategy calls the agent at a given frequency in order to re-plan the trajectory on the basis of the actual observations $o(t+1) \in O$. The final performed trajectory will be a sequence of different trajectory segments elaborated at different times according to the observed states of $\mathbf{P}_{DLO}$. The great advantage is given by the possibility of receiving feedback from the real cable and using it to adjust the TCP motion, making the solution more robust with respect to model uncertainties. However, observing in real-world scenarios $\mathbf{P}_{DLO}$ during high-speed manipulations is challenging. Strategies based on Deep Neural Networks for fast DLO state estimation, building on the work of [22], could be developed, but they remain an open challenge. Due to this problem, the multi-step episode agent has not been tested in real-world in this work.

V. Results

For all tasks, training is performed using a PC with an Intel(R) Core(TM) i7-9850H 2.60GHz CPU with 64 GB of RAM and a NVIDIA QUADRO RTX300 GPU. To speed up computations, GPU acceleration is used simulating eight environments in parallel. Considering the trade-off between computational effort and need of well-shaped trajectories to achieve the tasks, the number of intermediate trajectory points used by the Trajectory Generation Function is selected to be 3 and only planar movements of the manipulator are considered, in the same way as [14] and [16]. For the sake of reproducibility, parameters and thresholds of the reward function for each task are reported in Table V.

In the following, simulation results from single-step episode policies are shown for all the described tasks. For the purpose of showing the potentiality of the multi-step episode strategy, the outcome from a multiple-step episode scenario is also presented. Then, learnt policies for tasks I-II-IV are deployed in real world and achieved performances using a real manipulator are detailed. Policy for task III has not been transferred on a real dual arm robot, but considerations similar to those reported in V-B are valid also for this case.

	Task I	Task II	Task III	Task I Multi-step	Task IV
λ_1	-5	-10	-10	-50	-50
λ_2	-1	-1	-1	-1	-3
λ_3	10	10	10	100	1.5
λ_4	-1	-2	-2	-1	-2.5
ε_{goal}	$0.01\,m$				//
M	$0.75\,m$	$0.3\,m$	$0.3\,m$	$0.75\,m$	//
ε_{oscill}	//				$0.05\,m$
$outreach$	$0.9\,m$				
v_{max}	$1.8\,m/s$				
$\varepsilon_{shoulder}$	$0.25\,m$				

TABLE V: Reward function coefficients.

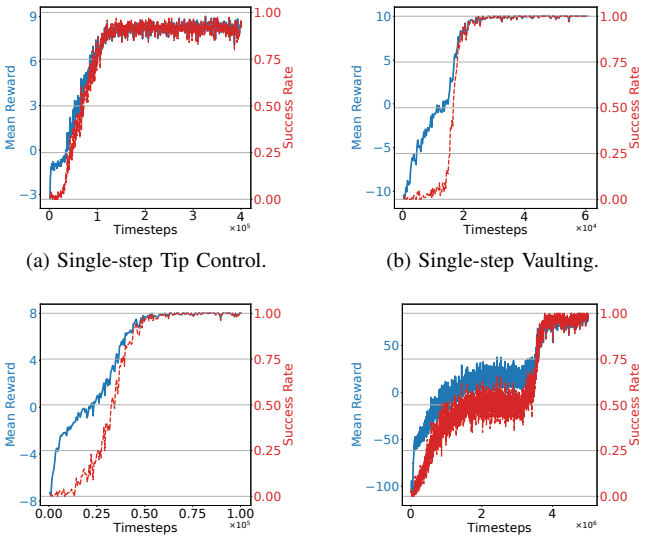
A. Simulation results

In the single step episode case, RL agents for tasks I and III are trained for $4{\cdot}10^5$ and 10^5 timesteps respectively, while for task II the process is limited to $6{\cdot}10^4$ timesteps. Based on the algorithm's working principle SAC requires much more learning time-steps with respect to PPO. As a consequence, RL agent for task IV in single step is trained for $1.5 \cdot 10^6$ timesteps.

For task I (both single and multi step episode), the target position in the environment is randomized in a range of $20\,cm$ (ensuring also that distance from TCP is sufficiently high), while in other tasks all positions are kept constant.

The training results are shown in Fig. 3.

Tasks I-III, Single-step: as Fig. 3a, 3b, 3c show, the DynDLO sandbox proves capable of training agents to devise trajectories that, once executed, are capable of dynamically manipulating DLOs to complete the implicit shape control tasks. All training procedures exhibit an initial plateau accompanied by negative rewards. This is attributed to constraints on end-effector velocity and position being enforced using the avoidance reward described in (5), requiring agents



(a) Single-step Tip Control. (b) Single-step Vaulting.

(c) Single-step Hooking. (d) Multi-step Tip Control.

Fig. 3: Success rate and mean reward per episode for agents trained for tasks I-III.

to spend training time to understand what the robot limits are. After this phase, success rates and rewards increase almost exponentially to the maximum values. The capability to correctly train agents even on multiple-step episode environments for implicit shape control is then tested by training an agent for task I for $5 \cdot 10^6$ time-steps, as detailed in the following.

Task I Multi-step: the learnt policy not only achieves a success rate close to 100% in the proposed task but also allows the benchmark of the trajectory re-planning mechanism, which proves capable of modifying the gripper trajectory without introducing significant discontinuities. By comparing Fig. 3a with Fig. 3d, we can see that the multiple step episode agent achieves a slightly better success rate in the proposed task thanks to its re-planning capabilities. Furthermore, one may notice that the multiple step episode agent for task I presents two plateaux during the training phase. This is due to the re-planning mechanism which intervenes to modify the TCP trajectory every $0.04\,s$. However, in some cases, this does not allow the cable tip to hit targets placed further away from the initial gripper position, as the policy settles on a local maximum (first plateau), which corresponds to an agent capable of hitting only targets which are sufficiently near. Nevertheless, by continuing training, the agent is capable of escaping this local maximum and converging to an optimal policy. This allows for larger manipulator motions, with the possibility of finding an optimal policy avoiding local maxima.

Furthermore, comparing Fig. 3a and 3d with Fig. 3b and 3c, one may notice that the former present a larger variance both in the mean reward per episode and success rate during training. This is because, both in the case of multiple and single step episode Tip Control, the position of the target with respect to the robot TCP is randomized in the simulated

scene. This increases the complexity of the RL environment, leading to noisier training metrics, but it also makes the agent robust with respect to uncertainties in target position and capable of devising trajectories to hit targets placed in different locations.

Task IV Single-step: explicit shape control agents are trained in two different situations: in the first one, the target lies above the starting point; in the other, below it. The second scenario has been introduced in order to show the ability of RL to deal with more complex challenges, since downward movements cause the transformation of potential energy into kinetic energy, making the vibration control problem more troublesome.

The results are extracted from a deterministic policy (the agent performs always the same action), since even very small variations in actions can have severe consequence on performance (over 50 tests, the oscillation amplitude exhibits a mean and standard deviation of $6.01\,cm$ and $3.43\,cm$, respectively). Results show that minimization is achieved successfully: in the first case, the target, that is $0.55\,m$ far from the TCP, is reached in $t^{up}_{traj} = 1.35\,s$ with a maximum residual oscillation amplitude of $oscill^{up}_{sim} = 1.34\,cm$, while in the second one, the trajectory time is $t^{down}_{traj} = 1.468\,s$ (TCP-target distance equal to $0.412\,m$) and the maximum recorded oscillation amplitude is $oscill^{down}_{sim} = 1.00\,cm$.

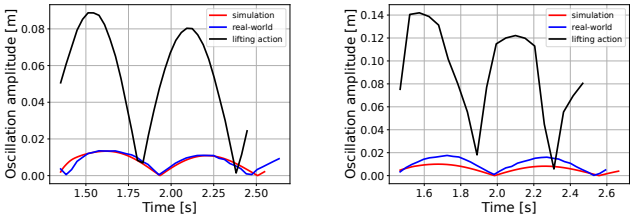
Single-step episode policies trained using the simulation environment for tasks I, II and IV are tested on a real experimental setup to assess the sim-to-real transfer capabilities of DynDLO and to compare their performance with other methods.

B. Real-World experiments

The trained agents in DynDLO define reference trajectories in the operational space of the manipulator. The corresponding joint trajectory references are computed through kinematic inversion using the TRAC-IK algorithm [23] with solver of type "Manipulation1" and seed equal to the solution obtained at the previous step in order to avoid wrist singularities and configuration changes.

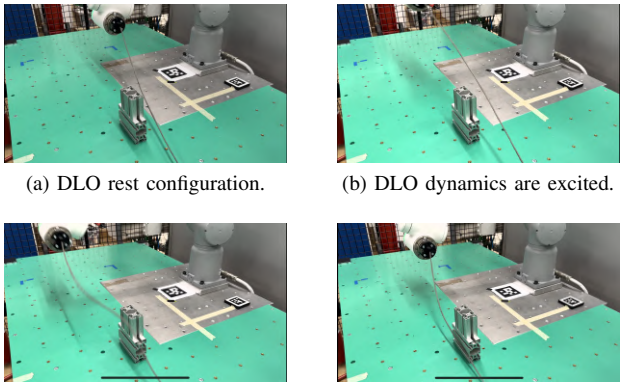
The resulting joint trajectories are provided as input to an ABB GoFa CRB $15000 - 5/0.95$ robot grasping a cable with length $l = 50 \backslash 90\,cm$ (depending on the task), mass $m = 10\,g$ and Young's modulus $E = 2 \cdot 10^6\,Pa$, obtained with a simple tensile test assuming elastic behaviour.

Task I Single-step: the policy performance is assessed by conducting ten experiments where the target position varies inside the randomization range used for training in simulation. Results show that the learnt policy is able to generate smooth trajectories which can be easily tracked by the robot's end-effector (Fig. 1) resulting in task success in $9/10 - 90\%$ of the cases. These results highlight how agents trained using DynDLO can be effectively deployed to real manipulators without the need for re-training. Moreover, with respect to [14], which needs to create a huge dataset and has to make attempts every time the goal is shifted, our agent trained with RL is able to immediately adjust the trajectory depending on the goal position, also attaining a higher success rate than [16].



(a) Target up case (b) Target down case

Fig. 4: Residual oscillations for task IV (single step episode).



(a) DLO rest configuration. (b) DLO dynamics are excited.

(c) Motion abruptly ends. (d) Succesful vaulting.

Fig. 5: Experiments on the Vaulting task demonstrate that the correct dynamic action is learnt, even though joint limits prevents an adequate sim-to-real transfer.

Task II Single-step: performed experiments show that some trajectories from the RL agent cannot be followed by the real robot with the obstacle considered in the DynDLO environment, resulting in task failure. This flaw arises because the provided limits at the end-effector do not guarantee that limits on joint velocities and accelerations are also satisfied. Nevertheless, by reducing the obstacle's height for the real-world experiments, we can appreciate that the agent is still capable of correctly performing a dynamic vaulting action as shown in Fig. 5. However, the single step episode agent is only able to achieve a $5/10 - 50\%$ success rate in the experiments conducted on the simplified setup as compared to one obtained in simulation (Fig. 3b), which is close to 100%. This sim-to-real gap arises due to a mismatch in the properties of the two cables adopted in the experiments (E is estimated considering the $50\,cm$ long cable) and due to limitation of the cable model to accurately represent this more complicated configuration.

Task IV Single-step: the agent performance in the real scenario is evaluated through the difference between the expected residual oscillations and the actual ones, which are shown in Fig. 4 (red and blue lines, respectively). Since oscillations in real world experiments exhibit a maximum amplitude of $oscill^{up}_{real} = 1.417 \pm 0.0732\,cm$ and $oscill^{down}_{real} = 1.8087 \pm 0.0684\,cm$ (over 10 tests), it is

possible to conclude that agents trained only in simulation can be directly transferred on the real robot. The sim-to-real gap is still present, particularly in the target-down scenario, but its impact on the final results is minimal, with a maximum deviation of less than $1.0\,cm$.

Due to the lack of a baseline for comparison (as the other methods proposed in the literature are tested only in simulation, see Table I), the agent performance is compared with a geometrically defined lifting action: a linear path bringing the end effector $20\,cm$ below the goal (or at $z = 55\,cm$ if the goal is too close to the floor) and then a fast upward movement, with a trajectory time equal to the one learned in DynDLO. Results show that RL agents outperform the geometric agent in both the situations (Figure 4, blue and black lines).

VI. Conclusions

In this paper, we presented DynDLO, a RL sandbox that allows training single and multiple step episode agents to perform various DLO dynamic manipulation tasks. The proposed framework allows RL agents to generate smooth trajectories for DLO dynamic manipulations, effectively achieving success across multiple tasks. Experiments conducted with four different benchmark tasks show that DynDLO can be used to generate policies which can be directly deployed to real manipulators without the need for re-training, contributing to bridging the sim-to-real gap. Future improvements will focus on the evaluation of the robot's bulk and accelerations at the joint level to enforce possible constraints on these quantities, with the goal of avoiding the issues observed during the real-world deployment of the policy for Task II.

Moreover, randomizing cable parameters inside a limited range can improve performances by making RL agents robust with respect to small estimation errors. Additionally, a suitable vision-based DLO tracking algorithm capable of tracking fast cable motions needs to be developed. This would enable the possibility of an effective implementation of multiple-step agents with real robots, contributing to bridging the sim-to-real gap.

References

[1] K. Almaghout and A. Klimchik, "Planar shape control of deformable linear objects," *IFAC-PapersOnLine*, vol. 55, pp. 2469–2474, 10 2022.

[2] A. Monguzzi, T. Dotti, L. Fattorelli, A. M. Zanchettin, and P. Rocco, "Optimal model-based path planning for the robotic manipulation of deformable linear objects," *Robotics and Computer-Integrated Manufacturing*, vol. 92, p. 102891, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584524001789

[3] D. McConachie, A. Dobson, and M. Ruan, "Manipulating deformable objects by interleaving prediction, planning, and control," *The International Journal of Robotics Research*, vol. 39, p. 027836492091829, 06 2020.

[4] M. Yu, K. Lv, C. Wang, M. Tomizuka, and X. Li, "A coarse-to-fine framework for dual-arm manipulation of deformable linear objects with whole-body obstacle avoidance," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 10 153–10 159.

[5] A. Monguzzi, Y. Karayiannidis, P. Rocco, and A. M. Zanchettin, "Force-based semantic representation and estimation of feature points for robotic cable manipulation with environmental contacts," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 16 139–16 145.

[6] R. Laezza, M. Shetab-Bushehri, G. A. Waltersson, E. Özgür, Y. Mezouar, and Y. Karayiannidis, "Offline goal-conditioned reinforcement learning for shape control of deformable linear objects," 2024. [Online]. Available: https://arxiv.org/abs/2403.10290

[7] R. Laezza, R. Gieselmann, F. T. Pokorny, and Y. Karayiannidis, "Reform: A robot learning sandbox for deformable linear object manipulation," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 4717–4723.

[8] F. Gu, Y. Zhou, Z. Wang, S. Jiang, and B. He, "A survey on robotic manipulation of deformable objects: Recent advances, open challenges and new frontiers," 2023. [Online]. Available: https://arxiv.org/abs/2312.10419

[9] F. Ding, J. Huang, T. Matsuno, and T. Fukuda, "Skill-based vibration suppression in manipulation of deformable linear objects," in *2011 International Symposium on Micro-NanoMechatronics and Human Science*, 2011, pp. 421–426.

[10] F. Ding, J. Huang, Y. Wang, T. Matsuno, and T. Fukuda, "Vibration damping in manipulation of deformable linear objects using sliding mode control," *Advanced Robotics*, vol. 28, no. 3, pp. 157–172, 2014.

[11] T. Knüppel and F. Woittennek, "Flatness based control design for a nonlinear heavy chain model," *IFAC Proceedings Volumes*, vol. 43, no. 14, pp. 701–706, 2010.

[12] T. Knüppel and F. Woittennek, "Control design for quasi-linear hyperbolic systems with an application to the heavy rope," *Automatic Control, IEEE Transactions on*, vol. 60, pp. 5–18, 01 2015.

[13] M. Nah, A. Krotov, M. Russo, D. Sternad, and N. Hogan, "Learning to manipulate a whip with simple primitive actions - a simulation study," *iScience*, vol. 26, p. 107395, 07 2023.

[14] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, "Iterative residual policy: For goal-conditioned dynamic manipulation of deformable objects," *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 389–404, 2024. [Online]. Available: https://doi.org/10.1177/02783649231201201

[15] H. Zhang, J. Ichnowski, D. Seita, J. Wang, H. Huang, and K. Goldberg, "Robots of the lost arc: Self-supervised learning to dynamically manipulate fixed-endpoint cables," in *IEEE Int. Conf. On Robotics and Automation (ICRA)*, 05 2021, pp. 4560–4567.

[16] Q. J. Chen, T. Bretl, N. Vuong, and Q.-C. Pham, "Dynamic manipulation of a deformable linear object: Simulation and learning," 2023. [Online]. Available: https://arxiv.org/abs/2310.00911

[17] S. Zhaole, J. Zhu, and R. B. Fisher, "Dexdlo: Learning goal-conditioned dexterous policy for dynamic manipulation of deformable linear objects," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 16 009–16 015.

[18] J. Huang, P. Di, T. Fukuda, and T. Matsuno, "Dynamic modeling and simulation of manipulating deformable linear objects," in *2008 IEEE International Conference on Mechatronics and Automation*, 2008, pp. 858–863.

[19] J. Gablonsky and C. Kelley, "A locally-biased form of the direct algorithm," *Journal of Global Optimization*, vol. 21, pp. 27–37, 01 2001.

[20] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[21] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[22] A. Caporali, P. Kicki, K. Galassi, R. Zanella, K. Walas, and G. Palli, "Deformable linear objects manipulation with online model parameters estimation," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–8, 03 2024.

[23] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 928–935.