Received July 21, 2021, accepted September 28, 2021, date of publication October 5, 2021, date of current version October 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3118209

Robot Learning-Based Pipeline for Autonomous Reshaping of a Deformable Linear Object in Cluttered Backgrounds

RICCARDO ZANELLA AND GIANLUCA PALLI, (Senior Member, IEEE)
DEI Department, University of Bologna, 40136 Bologna, Italy

Corresponding author: Riccardo Zanella (riccardo.zanella2@unibo.it)

This work was supported in part by the European Union's Horizon 2020 Research and Innovation Program as part of RIA Project Robotic tEchnologies for the Manipulation of cOmplex DeformablE Linear objects (REMODEL) under Grant 870133.

ABSTRACT In this work, the robotic manipulation of a highly Deformable Linear Object (DLO) is addressed by means of a sequence of pick-and-drop primitives driven by visual data. A decision making process learns the optimal grasping location exploiting deep Q-learning and finds the best releasing point from a path representation of the DLO shape. The system effectively combines a state-of-the-art algorithm for semantic segmentation specifically designed for DLOs with deep reinforcement learning. Experimental results show that our system is capable to manipulate a DLO into a variety of different shapes in few steps. The intermediate steps of deformation that lead the object from its initial configuration to the target one are also provided and analyzed.

INDEX TERMS Deformable linear object, manipulation, robot vision, robot learning.

I. INTRODUCTION

Deformable and non-rigid objects are extensively manipulated in our everyday life. Paper, cloths, wires, food, are only few examples. Thus, deformable object manipulation is an essential skill for robot to enter the human living and working environments. For instance, robots could become more involved in forestry operations [1] or healthcare activities for the elderly and disabled [2]. Also many industrial applications require robots able to manipulate non-rigid objects. Food industry, for example, could boost the production [3], farming industries could use robots to manipulate plants to lessen physical burden on workers [4] and manufacturing industry can minimize labor cost [5]–[7]. Despite the numerous applications and the effort made by the robotics community [8], effective and reliable methods for deformable object manipulation remain exceptionally difficult to construct.

Earlier works on deformable object manipulation have sought open-loop strategies, which are ineffective since the material can shift in unpredictable ways [9]. Successive works attempted to develop various model-based strategies for controlling the object shape through robot manipulation [10], [11]. This is a common and effective approach with rigid objects, but it results weak with non-rigid objects. Indeed, there is no obvious mapping from an observation of the object to a compact representation in which planning can be performed.

Deep Reinforcement Learning (DRL) is becoming more and more popular in robotic manipulation [12]–[17]. We are actually witnessing a run for the best DRL algorithm (in terms of flexibility and efficiency), that would enable the robot to perform any kind of manipulation, without engineering but only through its personal interacting experience with the environment [18]. However even the state-of-the-art solutions based on DRL algorithms produce results [15]–[17] quite far from those achievable with classical engineering methods.

The challenge in these works is the development of an algorithm which could learn the joint torque trajectories for a generic task directly from the input raw images by means of a rewarding system. This process demands to the agent to intrinsically learn operations like inverse kinematics, trajectory planning, visual feature extraction, object detection and semantic segmentation. All problems extensively studied and efficiently solved in literature.

Anyway, in order to discard the requisite of a model, one of the major challenge when interacting with deformable

The associate editor coordinating the review of this manuscript and approving it for publication was Kin Kee Chow.

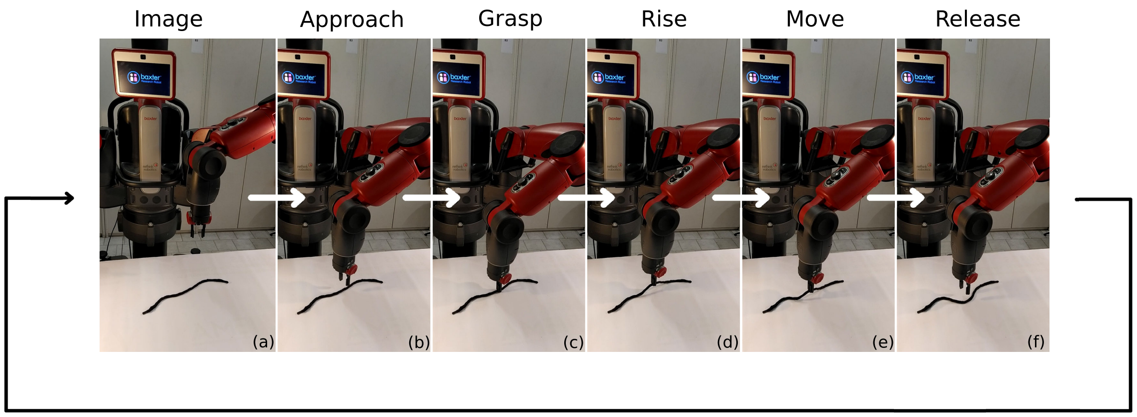

FIGURE 1. Pick-and-drop trajectories performed by the robot during every iteration of the proposed algorithm. It starts acquiring an image of the table with the hand camera (a). The decision process selects a grasping point based on this image, and computes the corresponding releasing point. In the second step the right arm moves toward the grasping point and stops 0.05*m* over the table (b). Then it grasps the DLO in the decided point (c) and returns to the approaching point (d). In the last two steps approaches the releasing point (e) and opens the gripper (f). Finally it returns in the initial configuration ready to start over, by taking a new image (a).

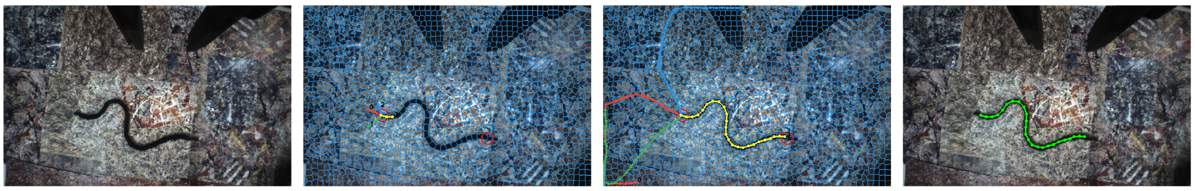

FIGURE 2. Image segmentation algorithm for DLOs. The first step consists in segmenting the input image into adjacent sub-regions (superpixels) and creating an adjacency graph. From the extremity of the DLO, an arbitrary number of walks are started, by moving into adjacent superpixels. Each walk moves forward along the adjacency graph by choosing the best next superpixels until it reaches the other extremity (this walk is masked as 'closed'). As a set of random walks are started, Ariadne keeps only the most likely one, among those marked as 'close'.

objects, reinforcement learning seems a very reasonable and very attractive approach [14], [19]. In fact, the optimization skills and the flexibility of DRL are essential to overcome the complex behaviour of deformable objects. However, to the state-of-the-art of DRL, a worth solution would be lighting the learning load by integrating the DRL algorithms with other non-learning-based tools and engineering consideration, in order to make the most of their capabilities.

In this work, we build a smart integration between efficient engineered solutions and DRL algorithms. In particular we propose a wise use of DRL algorithms in the few tasks in which the process needs to predict the optimal interaction with the DLO. While we prefer to employ a stable inverse kinematics (IK) solver and a trajectory planner to perform the robot motion. Moreover, we lighten the information extraction from visual data with a state-of-the-art vision technique specifically designed for DLOs [20]. The presented work is motivated by the lack of effective application solutions for DLO manipulation in tasks like untangle, spread and routing a wire in assembly processes [7], [21]. Thus, our study wants to move a step forward into these challenging tasks, proposing a solution able to control the shape of a DLO in a clutter environment using vision feedback.

In line with our work, also other authors adopted similar approaches. Boularias *et al.* [22] explores the use of DRL combined with well-known techniques for image segmentation, for manipulating unknown objects. They propose a pipeline that first segments images into separated objects, predicts pushing and grasping actions, extracts hand-tuned features for each action, then executes the action with highest expected reward. In [23] and [24], to make training tractable on a real robot, they simplified the action space to a set of end-effector-driven motion primitives. They formulate the task as a pixel-wise labeling problem: where each image pixel – and image orientation – corresponds to a specific robot motion primitive executed on the 3D location of that pixel in the scene. Similarly to these works, we turn action prediction into a classification problem by discretizing the action space and we define specific robot motion primitive (grasping and releasing).

The main contributions of our work are: (1) a novel robot learning-based system for autonomous deformation of a rope from/to a general shape using visual feedback capable to work with any cluttered background; (2) a study on DLOs deformation through a re-positioning sequence, in particular we investigated different strategies to decide the grasp/release locations and their relations.

The remainder of this paper is structured as follows: section II, reports an overview of previous works in this field; section III presents the experimental setup; section IV provides relevant background on reinforcement learning and Deep Q-Network; section V describes the proposed method in details; finally, in section VI, we examine the experiments and make some piratical considerations.

II. RELATED WORKS

The problem of DLOs manipulation has been studied before, with particular attention to tying knots. For instance, Yamakawa *et al.* [25] proposed a trajectory planning approach where a knot can be tied with a single robot arm at high speed. Mayer *et al.* [26] examined the use of recurrent neural networks to learn the knot tying trajectories. Learning from Demonstration (LfD) was proposed by Lee *et al.* [27] to learn a function that maps a pairs of correspondence points, while minimizing a bending cost.

The insertion of a DLO in a hole is another widely investigated task, due to all the useful applications that it would have in assembly operations [6], [7]. Inaba and Inoue [28] developed an hand-eye system to insert a rope into a hole using stereo vision for computing the relative position between rope tip and hole. In [29] they presented a method to insert string through tight workspace openings online using an approximate Jacobian to estimate the motion of the string. In [30] the insertion of a DLO into a hole is performed by analyzing the feedback coming from a tactile sensor by means of a recurrent neural network which estimate the force acting on the wire itself.

Few works attempt to address the shape control of a DLO using a robot. Rambow *et al.* [31] used a two-arm robot to mount a deformable tube in a desired configuration based on a single teleoperated demonstration. Nair *et al.* [23] developed a learning-based system where a robot takes as input a sequence of images showing small deformations of a rope from an initial to the goal configuration, performed by a human demonstrator, and outputs a sequence of actions that would lead the rope to the target shape, imitating the demonstrator deformations sequence. In [23] a Baxter robot has been configured to collect interaction data with the rope for 500 hours, used later to learn an inverse dynamics model which is finally employed to imitate the human demonstration. Similarly, also Sundaresan *et al.* [32] proposed an approach using imitation learning to arrange the configuration of a rope. They also show that the proposed solution can be used for a knotting task from human demonstration and assuming to start always from the same configuration containing a single loop. To brake symmetry and enable consistent correspondence mapping with target shape in [32] and [33] added, respectively, a ball and a blue tape. Moreover, in [33] they also tied one end of the rope to a clamp attached to the table. In this work, instead, we use a perfectly symmetric rope, with both the extremity free and identical. Another recent work on the same topic is [34], where they estimate a state-space representation of the rope and learn a dynamics model with an LSTM network and solve the rope manipulation with MPC. The weakest point of this solution is the assumption of having a strong color contrast between the rope and the table for a correct state estimation.

Differently from the over mentioned works, we addresses the problem of autonomous deformation of a rope from/to a general shape by training a reinforcement learning agent from scratch on a real robot, without: (1) the necessity of demonstrate the intermediate deformation steps in test time; (2) adding easily distinguishable object to brake the rope symmetry; (3) fixing any extremity to the table; (4) making any restrictions on the background color. In the sequence of Figure 4 we used a white background to make images clearer and to facilitate readers in the vision of the rope. However, as explained in subsection V-B, the system is designed to work on heterogeneous and confusing backgrounds.

III. EXPERIMENTAL SETUP

For the experiments described in the paper, we employ a Rethink's Baxter robot, which has a wrist-mounted gripper with two degrees of freedom (one rotational and one for closing/opening the two fingers). An RGB camera integrated with the robot hand provides visual data, with a resolution of 960×600 px.

The setup is illustrated in Figure 1. Also in this case, a white background is used to make images clearer and to facilitate readers in the vision of the rope. However, it is worth to remark that, as explained in subsection V-B, the system is designed to work on heterogeneous and confusing backgrounds, see e.g Figure 2.

A perfectly symmetric DLO (i.e. a rope), lies free on a table, at a known height z_*, in front of the robot. We define a fixed camera pose over the table to acquire the input RGB image. The interaction of the robot with the rope is limited to two simple motion primitives consisting of grasping the rope at location (u_1, v_1) and releasing it at location (u_2, v_2), where u_1, v_1, u_2, v_2 are pixel coordinates in the input RGB image. Since both the table height and hand-camera pose are known with respect to the robot base frame, we can estimate the grasping (x_1, y_1, z_*) and releasing (x_2, y_2, z_*) coordinates in the base frame.

As shown in Figure 1, during the grasping the robot first approaches the point (x_1, y_1, z_*) from the top, with and offset of $z' = 0.05$ m along the vertical z-axis and the gripper open. It moves down with a linear trajectory in the Cartesian space along z to z_*, then it close the gripper's fingers before rising back to $z_* + z'$. The motion sequence for dropping the rope is the same, with the intuitive difference that it starts with the gripper close, and opens it after the descent to z_*. In both the motion primitives, the motion planning is automatically executed with the native Baxter's IK solver.

IV. PRELIMINARIES ON DRL

We formulate the grasping task as a Markov decision process defined by $(\mathcal{S}, \mathcal{A}, p, r)$. Where state space $\mathcal{S}$ and action space $\mathcal{A}$, that represent respectively all possible combination of current and target shape and all possible grasping point in the scene, are assumed to be discrete. In subsection V-B and subsection V-D we illustrate the discretization strategy and we define the environment's state, while in subsection V-E we define the agent's actions. The unknown state transition probability $p(s_{t+1}|s_t, a_t)$ represents the probability density of the next state s_{t+1} given the current state s_t and current action a_t. For each state s_t at time t of the environment (i.e. the DLO), the agent (i.e. the robot) chooses and executes an action a_t according to the policy $\pi(a_t|s_t)$, which implies the transition of the environment to a new state s_{t+1} and the formulation of a reward r_t as defined in subsection V-D. Under this formulation, the goal is to find an optimal policy π^* that maximizes the expected sum of future rewards $\sum_{t=i}^{+\infty} \mathbb{E}_{(s_t,a_t)\sim p_\pi}[r_t]$, where we use ρ_π to denote the state or state-action marginals of the trajectory distribution induced by a policy $\pi(a_t|s_t)$.

In this work, we investigate the use of deep Q-learning, that is a Q-learning where a deep neural network is used to approximate the Q-value function $Q_\pi(s_t, a_t) = \sum_{t_i=t}^{T} \mathbb{E}_{\pi_\theta}[r_t|s_t, a_t]$, which measures the expected reward of taking action a_t in state s_t at time t. The network that approximates Q-value function is called Deep Q-Network (DQN) [35] and the training data are processed by using stochastic gradient updates. In Q-learning, a greedy policy $\pi(a_t|s_t)$ is trained to choose optimal actions by maximizing the action-value function $Q_\pi(s_t, a_t)$. Formally our learning objective is to iteratively minimize the temporal difference error δ_t of $Q_\pi(s_t, a_t)$ to a fixed target value y_t,

$$\begin{aligned} \delta_t &= |Q_\pi(s_t, a_t) - y_t| \\ y_t &= r_t + \gamma\, Q_\pi\left(s_{t+1}, \underset{a' \in \mathcal{A}}{\text{argmax}}\, Q_\pi\left(s_{t+1}, a'\right)\right) \end{aligned}$$

where $\gamma \in \mathbb{R}_+$ is called the discount rate.

V. METHOD

A. OVERVIEW

In this section, we describe our method to reshape a DLO using a single arm robot. The proposed method relies on a DQN-based decision process that leverages on an effective visual representation of the DLO shape. Current and target shapes are modeled using both a *Key Points Path* and a *Spatial Grid Matrix*, detailed in subsection V-B. The interaction with the DLO, and its reshaping process, take place through a sequence of grasping and releasing operations. The decision process, detailed in subsection V-C, learns to predict the best grasping point from the input image while the corresponding releasing point is computed by projection. A sample sequence of steps that leads the DLO to the target shape is shown in Figure 4. Since the proposed method relies on a reinforcement learning algorithm, in subsection V-D and subsection V-E, we formally define states, actions and rewards, while in subsection V-F some considerations about the training and how we speed it up when starting from scratch are made.

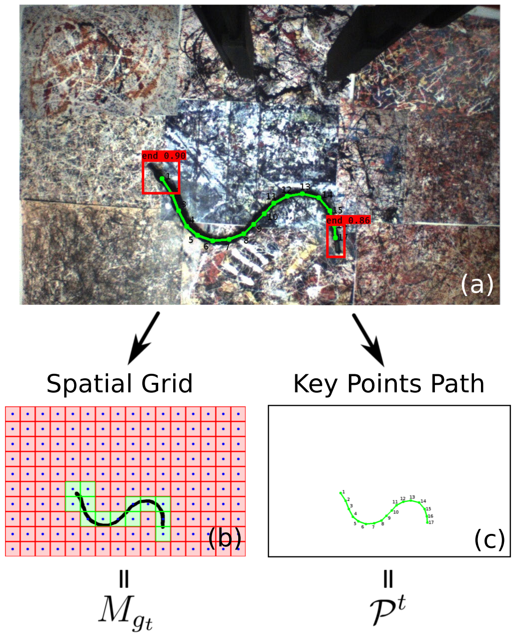

FIGURE 3. The input raw image is processed by ariadne (a). Since it needs to be initialized with the DLO extremities, YOLO object detector is employed for the purpose. Ariadne produces a binary mask and a list of image points that describes a walk along the DLO. From the binary mask we create the spatial grid (b) and define the matrix M_{g_t}, while from the points path (c) we obtain the list of points $\mathcal{P}^t$.

B. SHAPE REPRESENTATION

In order to effectively exploit its decision-making skills, the DRL agent has been integrated into a framework that lightens the learning load, as will be detailed in subsection V-C. This process is based on two representations of the DLO, both shown in Figure 3, processed from the visual input. The first representation, consists of a sorted sequence of key points belonging to the DLO. This representation allows us to effectively identify the releasing point on the target shape as a projection of the grasping point (taken from the current shape). In this way the agent needs to learn only the grasping point. In the second representation, a dimensionality reduction of the visual data is performed by mapping the segmentation mask into a spatial grid matrix. This matrix will later compose the state of the environment that the agent uses to predict the best action to perform.

Both the representations relays on an algorithm called Ariadne [20], able to perform simultaneously instance segmentation and b-spline modeling of DLOs. The basic idea of Ariadne is to detect the DLOs as suitable walks over the Region Adjacency Graph built on a super-pixel over-segmentation of the source image. In Figure 2 is visible an example of segmentation on a cluttered background.

1) KEY POINTS PATH

Ariadne segments the image into adjacent sub-regions (superpixels) then finds a walk that connects the two extremities of the DLO. This walk is essentially a sorted list of superpixels, that can be represented by their centroids, hence it can be converted into a sorted list of image points $\mathcal{P} = [p_1, \dots, p_n]$. Each walk need to be initialized with seed superpixels located at the DLOs' extremities. Purposely, we deployed YOLO v2 [36], an object detection tool based on convolutional neural networks. We fine-tuned the YOLO v2 model, pretrained on ImageNet, on a dataset that we created with the black rope used in the experiments. To create this dataset we developed an automated labeling tool based on video sequences that we allows us to easily gather massive amounts of training images in the field with minimal human intervention [37]. The tool is based on the idea that restricted camera movements (i.e. lift and rotate) leads to a controlled rigid transformation $\mathbf{A}$ between the two consecutive images I_i, I_{i+1} such that $I_{i+1} = \mathbf{A}I_i$. The same rigid transformation A can be applied to each bounding box (BB) $\breve{b}_i$ present in the image I_i so as to obtain a new set of BB such that $\breve{b}_{i+1} = \mathbf{A}\breve{b}_i$. This procedure can be repeated for each consecutive pair of images in the video sequence, it is therefore clear how the sole human intervention is to create the BB labels in the first frame I_0.

2) SPATIAL GRID MODEL

A uniform space partitioning is performed on a binary image mask $I_t^{\text{mask}} \subseteq [0, 1]^{h \times w}$ obtained as segmentation of the DLO from the input RGB image $I_t \subseteq [0, 255]^{3 \times h \times w}$. This partitioning consists of a set with size $n_{\text{rows}} \times n_{\text{cols}}$ of rectangular regions of pixels $\{\Psi_{i,j} \in \mathbb{R}^{\psi_h \times \psi_w}\}_{i \in n_{\text{rows}}, j \in n_{\text{cols}}}$ (image windows) with constant size $\psi_h \times \psi_w = \frac{h}{n_{\text{rows}}} \times \frac{w}{n_{\text{cols}}}$. Each region is mapped into a scalar value $g_t^{i,j} = \Omega_{[0,1]}\left(\frac{1}{\psi_h \psi_w} \sum_{u,v \in \Psi_{i,j}} I_t^{\text{mask}}[u, v], g^{\text{Th}}\right)$, that is the average of all the region-pixels binarized through the function $\Omega_{[0,1]}(x, x^{\text{Th}})$, which gets 1 only when $x \geq x^{\text{Th}}$ and 0 otherwise. From these values we define the spatial grid matrix at time t as $M_{g_t} = [g_t^{i,j}]_{i \in n_{\text{rows}}, j \in n_{\text{cols}}} \in [0, 1]^{n_{\text{rows}} \times n_{\text{cols}}}$, where every cell (i, j) and every region $\Psi_{i,j}$ have a bijective correspondence. To simplify position calculations, each region is represented by its center point.

C. DECISION PROCESS

The goal is to reshape a DLO by means of a sequence of grasp and release operations. To achieve this we employ the decision-making process schematically outlined in Figure 5. This process aims to determine the optimal grasping and releasing points, respectively $p_{\text{grasp}} \in \mathbb{R}^2$ and $p_{\text{release}} \in \mathbb{R}^2$, in order to maximize the visual overlap between the current and the target shapes, using as input data the image of the current scene.

A straightforward approach that we initially explored is to train an agent for learning jointly the two optimal locations p_{grasp} and p_{release} from the observation of the current scene s_t. However, the releasing location is strongly dependent on the grasping point, but the over mentioned approach does not take into account this conditional nature of the two operations.

To address this problem, we could combine two agents in a cascade, where the first predicts the grasping point and the second the releasing point. In other words, instead of learning jointly the two locations with an unique policy $\pi([p_{\text{grasp}}, p_{\text{release}}]|s_t)$, we define two policies: one that learns the grasping point from the current state $\pi_{\text{grasp}}(p_{\text{grasp}}|s_t)$; while the other one learns the releasing point from both the state and the predicted grasping location $\pi_{\text{release}}(p_{\text{release}}|s_t, p_{\text{grasp}})$. Nevertheless, training this policy is inefficient. In fact, the two operations would require two dedicated rewards, but we can only generate one reward after the releasing which is proportional to the visual overlap between the current and the target shapes. Clearly, in this setup, the decision process does not have the possibility to understand if an high (or low) reward is due to π_{grasp} or π_{release}.

Ultimately, to overcome also this action reward assignment issue, we propose to only learn the grasping point, while the releasing point is derived from the *key points path* representation of current and target shapes presented in subsection V-B. In fact, given the target shape path $\mathcal{P}^* = [p_1^*, \dots, p_m^*]$ and the current shape path $\mathcal{P}^t = [p_1^t, \dots, p_n^t]$, we can easily project a point form one path to the other. In particular we can project the grasping point p_k^t, taken from $\mathcal{P}^t$, into a releasing point p_s^* belonging to $\mathcal{P}^*$, where $s = \left\lfloor k\,\frac{m}{n} + \frac{1}{2} \right\rfloor$.

Having established that we can find the placing location with this projection strategy, one might wonder if we can choose also the picking point simply form the representations, without learning it. The most trivial solution would be grasping every time a random point from those that are not overlapped with the target shape. This is clearly very ineffective, since it neglects the DLO property of interconnection among the key points. Moreover, if we grasp only the free points, i.e. those that are not-overlapped, we cannot ensure to really reshape the rope, since the algorithm would simply aim to clean all the free points moving them to the target location. Hence, a trivial solution such as winding the rope in a small region that completely overlaps just a portion the target would conclude erroneously the task if there no free point is left. On the other hand, if we grasp also those already overlapped, we risk to make many pointless re-positioning actions. Another trivial approach would be following the order in the path, but also in this case we are not taking into account the interlinked nature of the object. In fact every time we place a point we might erroneously move those that we placed earlier.

As already stated, in the proposed solution we develop a decision process based on a DQN agent that learns the optimal grasping cell (action) in a grid that combine the spatial information of both the target and the current shapes (state). As shown in Figure 5 the agent is wrapped into a structure that defines the agent's state by extracting the useful features from the input image and derives the grasping and

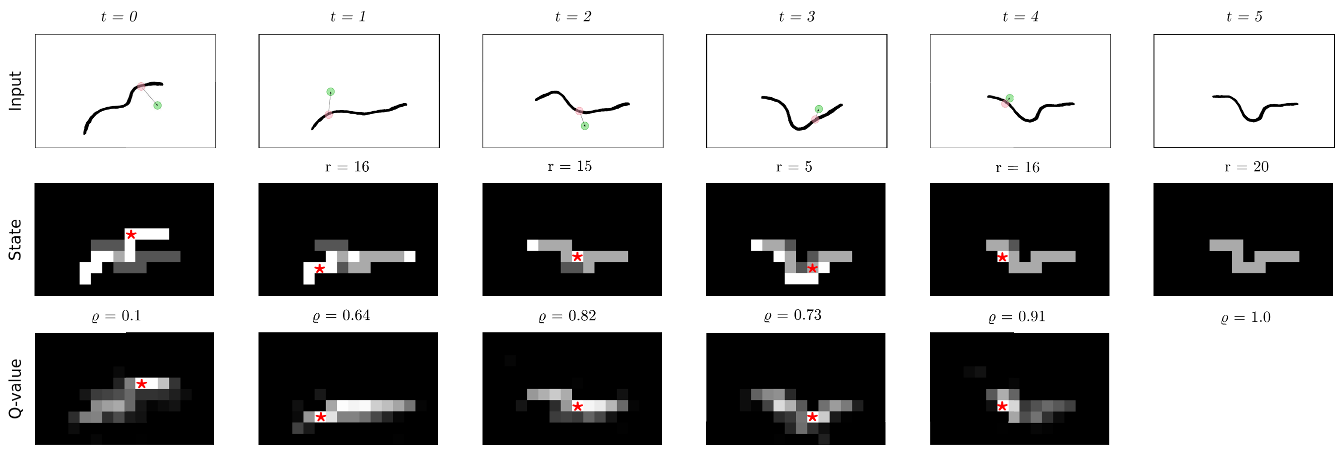

FIGURE 4. **Example of grasping-and-releasing sequence for reshaping the DLO. From the target and the current input visual data (first row) we define the state s_t (second row) and predict the Q-value $\phi_Q(s_t)$ (third row). We use a visual representation of the state where a cell can be: black if part of the background ($s_t^{i,j} = 0$); white if part of the current shape only ($s_t^{i,j} = 2$); light-grey if part of the overlapped region ($s_t^{i,j} = 3$); dark-grey if part of the target shape only ($s_t^{i,j} = 1$). In each step t, we obtain the action a_t as the coordinates to the highest value of $\phi_Q(s_t)$ (red star). On the input images we draw the grasping (red circle) and the releasing (green circle) points correspondent to the predicted action a_t. For each transaction we also compute the reward $r(a_t, s_t, s_{t+1})$, as a function of the overlap score $\varrho(s_t)$ (see Equation 3).**

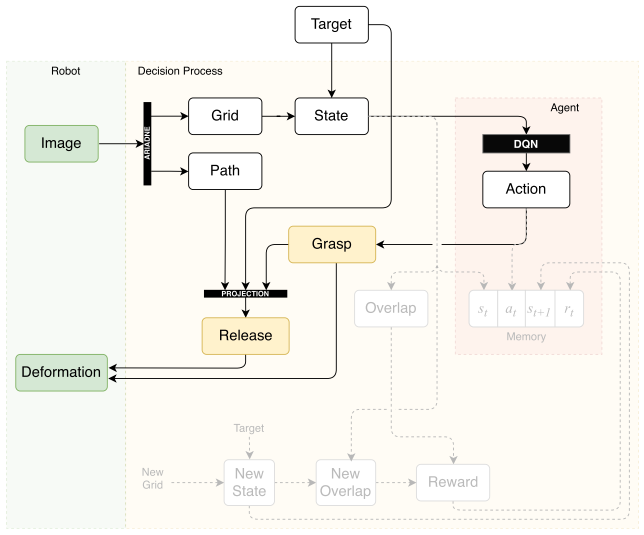

FIGURE 5. **Scheme representing the proposed method. We highlight in green the *robot side*, which includes the image acquired by the hand camera and the deformation (grasp and releasing operations) executed on the DLO. The decision making process is highlighted in yellow and the agent in red. The scheme shows also the agent's memory update, with dashed lines and grey boxes. In particular, the bottom part of the scheme reports the *new state* and *new overlap* that are obtained from the same scheme in the successive time step, from the new image acquired after the deformation.**

releasing point from the agent's action. The task starts by providing a goal that can be either a key points path and a spatial grid or a raw image of the rope in a target shape. In each iteration the system acquires a new RGB image of the scene. Then, the visual segmentation algorithm, creates the binary mask and the key points path for the current shape. The mask is reduced to the correspondent spatial grid matrix, which is combined with the target's one, as defined in subsection V-D, to obtain the state. The agent predicts the best action for the current state, i.e. it provides the optimal grasping cell of the spatial grid, as detailed in subsection V-E. This action needs to be mapped into a grasping point with respect to robot frame $\{B\}$, so first we find the point in the input image as center of the region of pixels corresponding to the grasping cell, and then, with the knowledge of the camera pose, we transform it with respect to $\{B\}$. The releasing point, as explained previously in this section, is obtained from the key points path and the grasping point. While the angle is simply estimated with a line fit algorithm from the image window contained in a the corresponding cell. Note that this estimation is affected by an ambiguity of π between current and target shapes. This would imply an undesired twist when releasing the rope. To have a consistent angle between the two shapes we can use the sorting information of the key points in the path. By consistently defining the two extremities on target and current shapes, the ambiguity is automatically solved. Obviously, arises a new problem on defining the extremities, since the DLO is perfectly symmetric. Let A and B be the end points of the current shape and A^* and B^* those of the target one. Thus, we define A^* as the end point of the target closer to A, which is instead arbitrarily assigned, and B^* the other one.

Once the robot has performed the deformation as explained in section III, a new iteration starts. In the successive iteration, the reward, that is function of the overlap score, and the new state are computed and sent to the agent, which records the transaction state, action, new state and reward for the learning. The task ends when the overlap score reaches a given threshold.

D. ENVIRONMENT

We model each state s_t as a linear combination of the spatial grid matrix of the scene at time t, M_{g_t}, and the

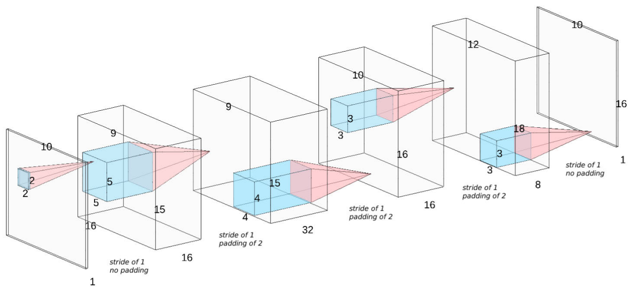

FIGURE 6. **The DQN is a CNN with five convolutional layers, where input and output have the same size 10 × 16.**

one of the target shape, M_{g^*},

$$s_t = 2M_{g_t} + M_{g^*}. \tag{1}$$

In this way the state is a matrix $s_t \in [0, 3]^{n_{\text{rows}} \times n_{\text{cols}}}$ where each element $s_t^{i,j}$ corresponds to the cell (i, j) of the spatial grid built on the scene. Note that it can be rewritten as

$$s_t^{i,j} = \begin{cases} 0 & \text{if } \Psi_{i,j} \text{ is part of the background} \\ 1 & \text{if } \Psi_{i,j} \text{ is part of the target shape only} \\ 2 & \text{if } \Psi_{i,j} \text{ is part of the current shape only} \\ 3 & \text{if } \Psi_{i,j} \text{ is an overlapped region,} \end{cases} \tag{2}$$

where the overlapped regions are set of image pixels belonging to both the target and the current shape.

E. AGENT

This work uses an implementation of deep Q-learning, where the DQN $\phi_Q(s_t)$ that approximate the Q-function $Q_\pi(s_t, a_t)$ is a convolutional neural network (CNN) schematically represented in Figure 6. Since both state and action space are quite simple by construction, simple network architectures can be used as well. The default architecture consists of five convolutional layers interleaved with nonlinear activation functions (ReLU) [33] and spatial batch normalization [38]. As already said, the input and the output of the DQN have the same size, that is the size of the spatial grid, $n_{\text{rows}} \times n_{\text{cols}}$.

1) ACTIONS

The agent predicts a vector action $a_t = [i\ j]^\top$, where $i \in \mathbb{N}^{n_{\text{rows}}}$ and $j \in \mathbb{N}^{n_{\text{cols}}}$ are the coordinates of a target region in the spatial grid where to perform the grasping. These coordinates are easily inferred from the DQN's output $\phi_Q(\cdot) \in \mathbb{R}^{n_{\text{rows}} \times n_{\text{cols}}}$. In fact, the matrix ϕ_Q has the same size of the spatial grid matrix M_{g_t}, thus we have a one-to-one correspondence between the elements. This implies that we can take $\phi_Q^{i,j}(s_t)$, the value in coordinate i, j of $\phi_Q(\cdot)$, as the approximated Q-value $Q_\pi(s_t, a_t)$ of the action $a_t = [i\ j]^\top$, or in other words, $\phi_Q^{i,j}(s_t)$ can be considered as the expected future reward of grasping the DLO in the region (i, j). Hence, the action that maximizes the Q-function is the couple of indices corresponding to the region with the highest Q-value across the spatial grid matrix: $\text{argmax}_{a'} Q_\pi(s_t, a') = \text{argmax}_{(i,j)} \phi_Q^{i,j}(s_t)$.

2) REWARD SHAPING

In our decision process we use a shaped reward. In fact, shaped reward functions compared to sparse reward functions, require more design effort as they incorporate knowledge of the problem into the reward structure, but in general they require less time to train, or at least they should speed-up the training in a complex setup.

The reward scheme we designed is very simple. First of all let us consider the state as written in Equation 2. We can easily assert that only the regions belonging to the current DLO shape are worth considering for grasping, which means that we can assign a reward $r(a_t, s_t, s_{t+1}) = 0$ to all the actions $a_t = [i\,j]^\top$ that leads the robot to the regions corresponding to the value $s_t^{i,j} \in \{0, 1\}$ or equivalently $g_t^{i,j} = 0$ (void grasping).

Let us consider now a valid action, $a_t \in \{[i\ j]^\top : s_t^{i,j} \in \{2, 3\}\}$. Our goal is to maximize the number of overlapped regions, ideally the algorithm should converge to a state in which current and target shapes are completely overlapped in the spatial grid model, that is $s_t : s_t^{i,j} \in \{0, 3\}, \forall i, j$.

We define an overlap score $\varrho(s_t) = \frac{n_{s_t=3}}{n_{s_t \neq 0}}$ at time t as the number of overlapped regions $n_{s_t=3}$ over the number of all regions that are either part of the current or the target shape $n_{s_t \neq 0}$. Hence, assuming that $\varrho(s_{t+1}) - \varrho(s_t) > 0$, the reward that we assign to a valid action is directly proportional to the increment in the overlap score

$$r(a_t, s_t, s_{t+1}) = k \left[1 + \frac{\varrho(s_{t+1}) - \varrho(s_t)}{1 - \varrho(s_t)} \right], \tag{3}$$

where $k \in \mathbb{R}$ is a gain that we set to $k = 10$.

Moreover, to penalize the actions that cause an overlap loss, $\varrho(s_{t+1}) - \varrho(s_t) \leq 0$, we assign a constant reward $r(a_t, s_t, s_{t+1}) = \frac{k}{2}$, greater than zero (since the action is still valid) but always smaller than Equation 3.

F. TRAINING AND TEST

We train the DQN using Adam optimization with fixed learning rates of 10^{-4}. Our models are implemented in PyTorch and trained with an NVIDIA GeForce GTX 1080 Ti on an Intel Core i7-7700K CPU clocked at 4.20GHz. We train with prioritized experience replay [39] using stochastic rank-based prioritization, approximated with a power-law distribution. Our exploration strategy is ϵ-greedy, with ϵ initialized at 0.7 then annealed over training to 0.1. Our future discount γ is constant at 0.5. The experience replay uses batches of size 132.

At the beginning of the training the DQN has random values and the agents can only take random actions in order to explore the environment. To speed this process up, human expertise can be used as agent's prior knowledge or heuristic. Hence, in the first phase of the training a human demonstrator provides a sequence of pick points on the rope toward the target shape, while the agent only collects data (i.e. state, action, reward and new state). Ideally, once the process is over, the agent has learnt a raw but satisfactory policy. Thus, in the second phase of the training, the agent can acts

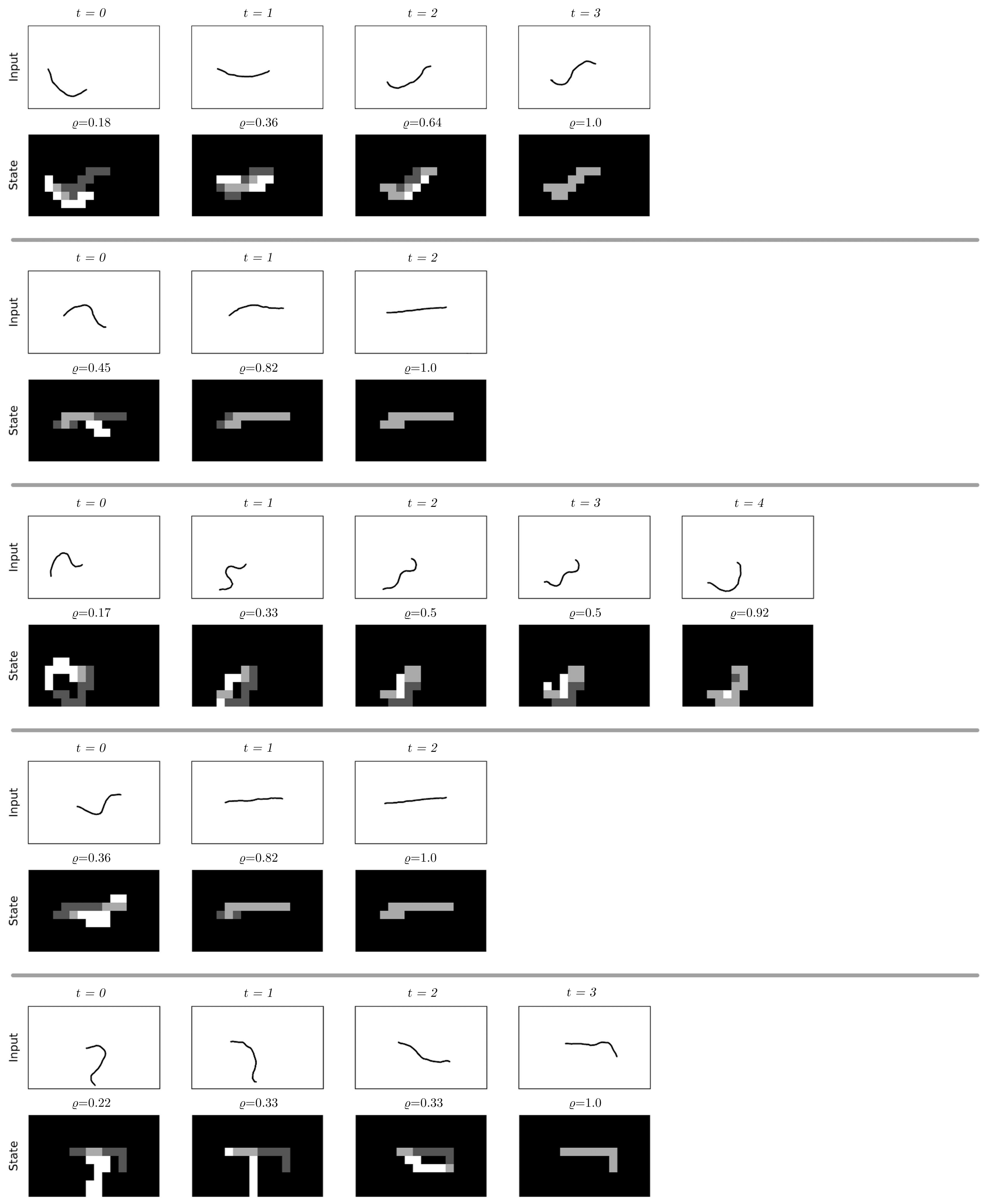

FIGURE 7. First set of 5 experiments that shows the DLO deformation steps performed by the robot using the proposed method. The images are binarized for visual clearance. The final shape corresponds to an overlap score greater than 90% ($\varrho(s_t) >= 0.9$). The state cells are: black if $s_t^{i,j} = 0$; white if $s_t^{i,j} = 2$; light-grey if $s_t^{i,j} = 3$; dark-grey if $s_t^{i,j} = 1$.

autonomously on the system and collects more self-generated data. Differently to other works like [23] or [32], the human demonstrations are used only to initialize the agent's experience and no longer needed in test time.

The demonstration phase is useful for gathering a large amount of meaningful data, possibly that cover a wide set of different scenarios. Hence, the demonstrator should to prevent the system to fall in some irrecoverable state (highly

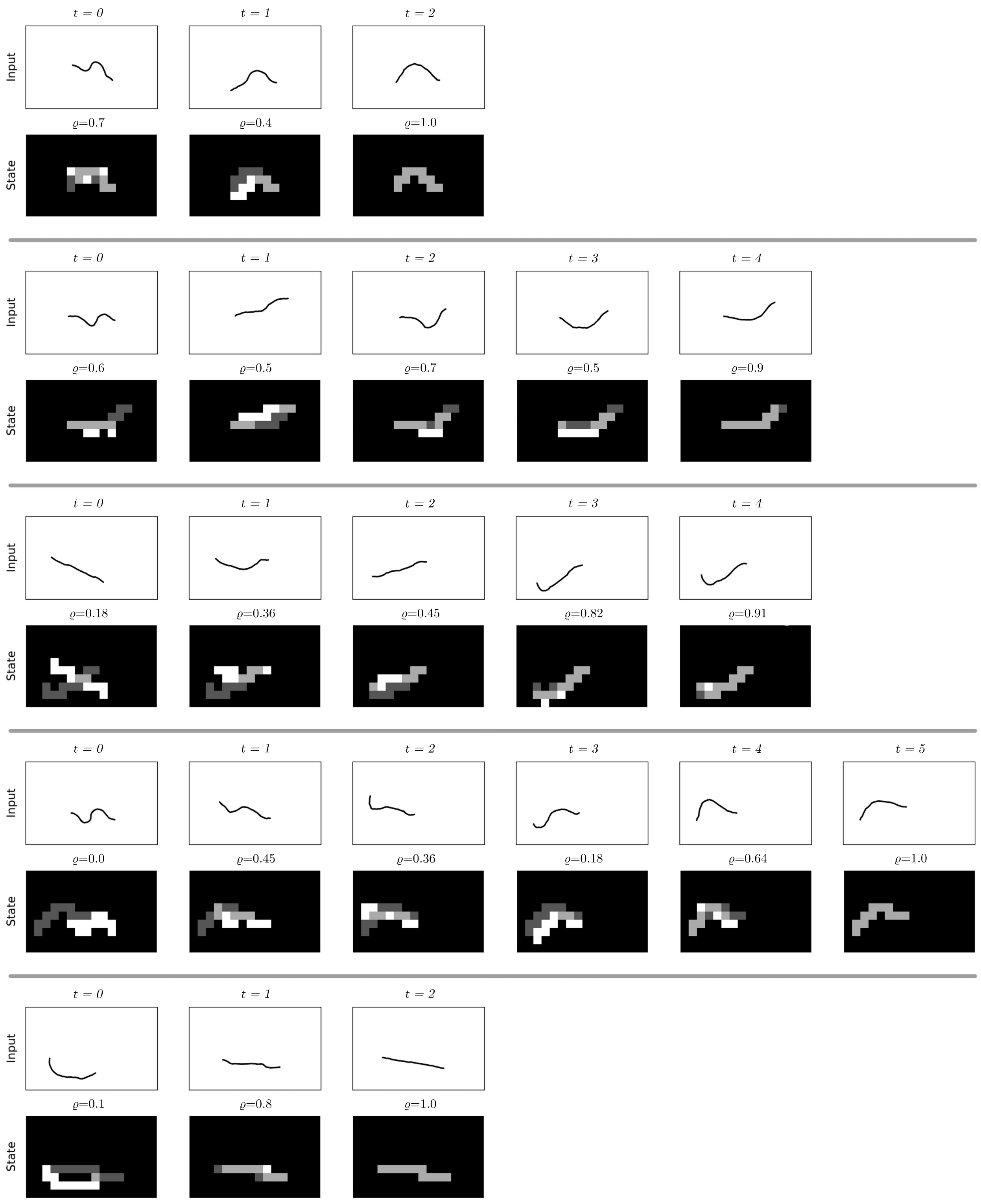

FIGURE 8. Second set of 5 experiments that shows the DLO deformation steps performed by the robot using the proposed method. The images are binarized for visual clearance. The final shape corresponds to an overlap score greater than 90% ($\varrho(s_t) >= 0.9$). The state cells are: black if $s_t^{i,j} = 0$; white if $s_t^{i,j} = 2$; light-grey if $s_t^{i,j} = 3$; dark-grey if $s_t^{i,j} = 1$.

tangled DLO) or in a loop of similar transitions, that could cause an over-fit in the DQN training. For this reason, in the first phase, we change target shape as soon as we reach an overlap score of $\varrho^{\text{Th}} = 0.5$, since over this threshold we are performing small adjustments and the state would remain similar in a long sequence of transitions. This fine learning can be done in the second phase of autonomous exploration, when the agent has already some raw experience on the task. Following this principle we gradually increase the overlap score threshold ϱ^{Th} up to 0.8 every 50 transitions with step $\Delta\varrho = 0.1$.

We observed that, the agent first learns to find the non-empty regions taking into account that all the regions are linked because part of the same DLO and some of them are already correctly aligned with the target. In order to avoid over-fitting the agent on a particular shape, we collected 30 target shapes and change among them every $n = 15$ transitions or every time the overlap score reaches the given threshold.

VI. EVALUATION

In this section we evaluate the proposed method on our experimental setup. The spatial grid considered for the DLO shape representation has size $n_{\mathrm{cols}} \times n_{\mathrm{rows}} = 16 \times 10$. We collected 200 transactions by demonstration and other 300 during the autonomous exploration phase. We evaluate the performance by counting the number of steps required to reach an overlap score greater than 90% ($\varrho(s_t) > 0.9$). By running the experiment on 30 different scenarios, we estimate a success rate of 76.7% (23/30 tests) in achieving the goal with less than 12 steps and 86.7% (26/30 tests) with less than 18 steps. In 4/30 tests we assumed a failure due to an undesired tangling. In Figure 7 and Figure 8 the 10 experiments are reported, showing the intermediate deformation steps performed by the robot and the agent's state. In this figure, the images have been binarized to improve readability. It is worth noticing that the system learns to stretch the DLO in only 2 steps by simply adjusting the two extremities.

The experimental data reported in Figure 4 show an example of correct learning, where the agent predicts as optimal grasping locations those that are not aligned with the reference shape. In particular, this behaviour is clearly visible in the first two steps and in the last one. Note also that the estimate Q-values are zero in the cells that are empty or occupied by the target shape only (not suitable for grasping). Moreover, while the cells not aligned with the reference are frequently preferred to those already aligned, these are not excluded, as happens in the 4th step of Figure 4.

VII. CONCLUSION

In this work we studied the robotic manipulation of a deformable linear object lying on a table, i.e. a rope, using visual data. The proposed method relays on a decision making process that learns the optimal grasping location from the input visual data, by means of a DQN agent, and finds the best releasing point from a path representation of the rope shape. Also other solutions are examined and discarded for inefficiency or inadequacy. Differently from other studies in that field, the proposed technique only needs very limited human intervention during the initial training phase, while the system is able to learn autonomously how to deal with generic scenarios thereafter.

Experimental results of reshaping tests are provided, showing the intermediate steps of deformation that lead the rope from its initial configuration to the target and we examined the output of the DQN in each step of a sample experiment. This results show that our system is capable to manipulate ropes into a variety of different shapes in few steps.

Since our technique only assumes a Q-learning algorithm with CNNs, we believe it can be easily improved by applying state-of-the art algorithms, e.g. HER [40] or including some awareness of the sequential deformation by integrating recurrent neural networks.

REFERENCES

[1] M. Bergerman, J. Billingsley, J. Reid, and E. van Henten, *Robotics in Agriculture and Forestry*. Springer, 2016, pp. 1463–1492.

[2] T. L. Mitzner, T. L. Chen, C. C. Kemp, and W. A. Rogers, "Identifying the potential for robotics to assist older adults in different living environments," *Int. J. Social Robot.*, vol. 6, no. 2, pp. 213–227, Apr. 2014.

[3] P. Y. Chua, T. Ilschner, and D. G. Caldwell, "Robotic manipulation of food products—A review," *Ind. Robot, Int. J.*, vol. 30, no. 4, pp. 345–354, Aug. 2003.

[4] H. G. Tanner, K. J. Kyriakopoulos, and N. I. Krikelis, "Advanced agricultural robots: Kinematics and dynamics of multiple mobile manipulators handling non-rigid material," *Comput. Electron. Agricult.*, vol. 31, no. 1, pp. 91–105, Mar. 2001.

[5] X. Jiang, K.-M. Koo, K. Kikuchi, A. Konno, and M. Uchiyama, "Robotized assembly of a wire harness in a car production line," *Adv. Robot.*, vol. 25, nos. 3–4, pp. 473–489, Jan. 2011.

[6] D. De Gregorio, R. Zanella, G. Palli, S. Pirozzi, and C. Melchiorri, "Integration of robotic vision and tactile sensing for wire-terminal insertion tasks," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 585–598, Apr. 2019.

[7] G. Palli, S. Pirozzi, M. Indovini, D. De Gregorio, R. Zanella, and C. Melchiorri, "Automatized switchgear wiring: An outline of the wires experiment results," in *Advances in Robotics Research: From Lab to Market*. Cham, Switzerland: Springer, 2020, pp. 107–123.

[8] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey," *Int. J. Robot. Res.*, vol. 37, no. 7, pp. 688–716, Jun. 2018.

[9] J. E. Hopcroft, J. K. Kearney, and D. B. Krafft, "A case study of flexible object manipulation," *Int. J. Robot. Res.*, vol. 10, no. 1, pp. 41–50, Feb. 1991.

[10] H. Nakagaki, K. Kitagaki, and H. Tsukune, "Study of insertion task of a flexible beam into a hole," *J. Robot. Soc. Jpn.*, vol. 14, no. 3, pp. 398–405, 1996.

[11] M. Moll and L. E. Kavraki, "Path planning for deformable linear objects," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 625–636, Aug. 2006.

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

[13] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation," *Robot. Auton. Syst.*, vol. 112, pp. 72–83, Feb. 2019.

[14] A. X. Lee, A. Gupta, H. Lu, S. Levine, and P. Abbeel, "Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to deformable object manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 5265–5272.

[15] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable deep reinforcement learning for robotic manipulation," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2018, pp. 6244–6251.

[16] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," 2019, *arXiv:1904.07854*. [Online]. Available: http://arxiv.org/abs/1904.07854

[17] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[18] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, ''Multi-goal reinforcement learning: Challenging robotics environments and request for research,'' 2018, *arXiv:1802.09464*. [Online]. Available: http://arxiv.org/abs/1802.09464

[19] J. Matas, S. James, and A. J. Davison, ''Sim-to-real reinforcement learning for deformable object manipulation,'' 2018, *arXiv:1806.07851*. [Online]. Available: http://arxiv.org/abs/1806.07851

[20] D. De Gregorio, G. Palli, and L. Di Stefano, ''Let's take a walk on superpixels graphs: Deformable linear objects segmentation and model estimation,'' in *Proc. Asian Conf. Comput. Vis.* Cham, Switzerland: Springer, 2018, pp. 662–677.

[21] K.-M. Koo, X. Jiang, K. Kikuchi, A. Konno, and M. Uchiyama, ''Development of a robot car wiring system,'' in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, Jul. 2008, pp. 862–867.

[22] A. Boularias, J. A. Bagnell, and A. Stentz, ''Learning to manipulate unknown objects in clutter by reinforcement,'' in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 1–7.

[23] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, ''Combining self-supervised learning and imitation for vision-based rope manipulation,'' in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 2146–2153.

[24] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, ''Learning synergies between pushing and grasping with self-supervised deep reinforcement learning,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4238–4245.

[25] Y. Yamakawa, A. Namiki, and M. Ishikawa, ''Motion planning for dynamic knotting of a flexible rope with a high-speed robot arm,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 49–54.

[26] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber, ''A system for robotic heart surgery that learns to tie knots using recurrent neural networks,'' *Adv. Robot.*, vol. 22, nos. 13–14, pp. 1521–1537, 2008.

[27] A. X. Lee, S. H. Huang, D. Hadfield-Menell, E. Tzeng, and P. Abbeel, ''Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 4402–4407.

[28] M. Inaba and H. Inoue, ''Hand eye coordination in rope handling,'' *J. Robot. Soc. Jpn.*, vol. 3, no. 6, pp. 538–547, 1985.

[29] W. Wang, D. Berenson, and D. Balkcom, ''An online method for tight-tolerance insertion tasks for string and rope,'' in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 2488–2495.

[30] R. Zanella, D. De Gregorio, S. Pirozzi, and G. Palli, ''DLO-in-hole for assembly tasks with tactile feedback and LSTM networks,'' in *Proc. 6th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Apr. 2019, pp. 285–290.

[31] M. Rambow, T. Schauß, M. Buss, and S. Hirche, ''Autonomous manipulation of deformable objects based on teleoperated demonstrations,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 2809–2814.

[32] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg, ''Learning rope manipulation policies using dense object descriptors trained on synthetic depth data,'' 2020, *arXiv:2003.01835*. [Online]. Available: http://arxiv.org/abs/2003.01835

[33] V. Nair and G. E. Hinton, ''Rectified linear units improve restricted Boltzmann machines,'' in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[34] M. Yan, Y. Zhu, N. Jin, and J. Bohg, ''Self-supervised learning of state estimation for manipulating deformable linear objects,'' *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2372–2379, Apr. 2020.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, ''Playing atari with deep reinforcement learning,'' 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602

[36] J. Redmon and A. Farhadi, ''YOLO9000: Better, faster, stronger,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7263–7271.

[37] D. De Gregorio, R. Zanella, G. Palli, and L. Di Stefano, ''Effective deployment of CNNs for 3DoF pose estimation and grasping in industrial settings,'' in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 7419–7426.

[38] S. Ioffe and C. Szegedy, ''Batch normalization: Accelerating deep network training by reducing internal covariate shift,'' 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[39] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, ''Prioritized experience replay,'' 2015, *arXiv:1511.05952*. [Online]. Available: http://arxiv.org/abs/1511.05952

[40] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, ''Hindsight experience replay,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5048–5058.

RICCARDO ZANELLA received the M.Sc. degree in automation engineering from the University of Padua, in 2016, and the Ph.D. degree in biomedical, electrical and system engineering from the University of Bologna, in 2021. He is currently a Research Fellow with the University of Bologna. His research interests include robot learning and deformable object manipulation.

GIANLUCA PALLI (Senior Member, IEEE) received the Laurea and Ph.D. degrees in automation engineering from the University of Bologna, Bologna, Italy, in 2003 and 2007, respectively. He is currently an Associate Professor with the University of Bologna. He has authored or coauthored more than 100 scientific articles presented at conferences or published in journals. His research interests include design and control of manipulation devices, mobile manipulation, manipulation of deformable objects, and soft robotics.