Visual servoing for dual arm shaping of soft objects in 3D

Célia Saghour, Mathieu Celerier, Philippe Fraisse, Andrea Cherubini

Visual servoing for dual arm shaping of soft objects

Célia Saghour, Mathieu Célérier, Philippe Fraisse and Andrea Cherubini
LIRMM, Université de Montpellier, CNRS Montpellier, France.

***Abstract*— We propose a real-time vision-based dual arm controller for manipulating soft objects in three dimensions. Our controller relies on a data-based approach to learn how the object deforms, without any prior knowledge of the shape nor material. We control the two arms with a cooperative task representation, which makes them move together in an absolute frame and/or relatively to one another. This allows object deformation in the relative frame and/or object motion in the absolute frame. Through a short initialization phase for extracting visual features with a Principal Component Analysis on the object contour points, we estimate an interaction matrix. This matrix relates the visual features to the robot control inputs for the cooperative task. At each step of the deformation, the process is reiterated with the last data and the interaction matrix is updated along a sliding window. Experiments on a few different objects show the accuracy of the method in real-time.**

I. INTRODUCTION

Despite the presence of non-rigid objects everywhere in our daily life and the increasing need for automating their manipulation, to date there is no generally applicable and easily implementable method to shape such objects. The high number of Degrees of Freedom (DOF) of soft objects makes it difficult to control their deformation during manipulation.

Manipulating objects such as clothes [1], wires [2] or tying suture on biological surfaces [3], [4] constitute important and open challenges in robotics and related areas. Folding clothes [5] or picking fruits and/or vegetables [6] are simple tasks for humans, which prove difficult for robots. Such tasks often involve two important aspects: the use of two coordinated arms, and the handling of non-rigid objects.

The works [7] and [8] have developed uncalibrated visual servoing, a method where the model mapping visual data to robot control is estimated online. Yet, these works focus only on rigid objects. In humanoid robotics, visual servoing for eye-to-hand coordination is largely used for tasks such as reaching and grasping [11] or manipulating rigid objects [12] sometimes in interaction with a human [10]. More recently, researchers have designed visual shape servoing controllers, relying on model estimation [13], neural networks [14] or reinforcement learning [15] to compute the command. Although these methods present good results, the first one requires a prior knowledge of the object (shape, material parameters) which limits its range of application, while the others require huge datasets and training time. It should also be noted that most of those methods rely only on single arm manipulation.

*This work was supported by European Union Horizon 2020 Research and Innovation Programme as part of the project SOPHIA under grant agreement No 871237.

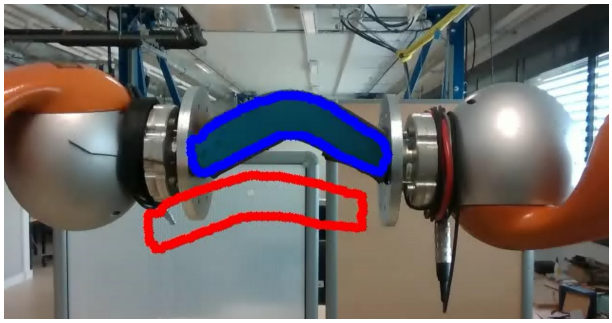

Fig. 1: The goal of our work is to control a dual arm robot so that the initial contour (blue) reaches the target contour (red) in three dimensions.

In this paper, we propose a real-time, dual-arm vision-based shape servoing controller. Our method does not require any prior knowledge of the object, apart from its color, needed for visual extraction. It applies to soft objects of different shapes and materials, with very short initialization time. In addition, our task representation allows a cooperative control of the two arms, to command 3D deformations and/or 3D motions of the tracked part of the object.

II. RELATED WORK

Few works present research on soft object manipulation using a dual arm robot; most works on multi-robots manipulation of soft objects are compiled in survey [16].

In recent years, research for soft object manipulation has mainly focused on learning methods, and none of them, to the best of the authors' knowledge, consider cooperative task representation. The authors of [17] use Deep Reinforcement Learning for clothing manipulation. To alleviate the need for heavy data-sets imposed by regular reinforcement learning methods, the authors propose two different algorithms that allow the learning process to be more efficient. However, samples (80 to 300 in their experiments) need to be generated by a human operator for prior policy initialization. Similarly, in [18], the authors use both Fast Point Feature Histogram and Principal Component Analysis (PCA) to obtain a feature vector, and Deep Neural Networks (DNN) to learn the deformation function between the object and the end-effectors. The network is trained online, but the DNN model is initialized with 100 data beforehand. The need for numerous training data is something we aim to remedy.

Some works focus on specific types of objects, such as Deformable Linear Objects (DLO). Paper [19] presents a model-free servoing method for shaping deformable wires.

The approach tracks the deformations, to extract a visual feature based on a geometric B-spline model, and to estimate iteratively the deformation Jacobian for control. In [20], Coherent Point Drift is used for state estimation, relating the point cloud of the DLO between two iterations. The authors however use learning by demonstration for task and trajectory planning, needing teaching by human operators. While these methods show good results with two end-effectors and in 3D, the use of B-spline modeling can only be used to represent one-dimensional shapes for one, while the second requires demonstrations for control.

In [21], the authors use contour moments as a state representation of the manipulated object. Their method is applicable for elastic, composite and rigid objects, but is limited to the 2D manipulation space. Similarly, the authors of [22] obtain a good encoding of the object shape, via PCA. Their method shows very promising results for moving and deforming 2D contours with a single arm moving in the plane (3 DOF). It can also be applied on both rigid and soft objects, allowing a large range of applications.

We build on top of that work, going farther and performing deformations in $\mathbb{SE}(3)$ with two *cooperating* robot arms, bearing in mind applications such as "shape and place". Our contributions are the following:

- While [22] considered 3 DOF planar motion (two translations and one rotation) of a single arm, we increase the robot operational space dimension to 12 DOF (6 per arm), hence extending the range of applications.
- We apply the cooperative task representation [**?**] to control the deformation and/or the pose of the object separately. We explore how both tasks affect the global deformation and the reaching of the target.
- We validate our controller in 3 dimensions (including orientation) in a series of experiments, with different objects and targets.

III. PROBLEM STATEMENT

We consider a dual-arm robot with both end-effectors holding an object, rigid or deformable. The dimension of the robot operational space is 12 (i.e., each end-effector is free to move in $\mathbb{SE}(3)$). The robot observes the object with a fixed RGB-D camera. The goal is to modify the position and/or shape of the object, to see it in a given fashion (a target image), shown in Fig. 1.

In some cases, the task may consist only in giving the object the desired shape, without caring about its position and orientation. For such a task the robot is *redundant*: it only requires 6 DOF, and can use the other 6 for other purposes. Yet, doing this is not trivial if each end-effector is modeled and controlled separately, as in [19]. A solution is to use the cooperative task representation introduced in [**?**] and outlined in Fig.2, to describe the task as the combination of an absolute task and/or a relative task. The absolute task frame $\mathcal{F}_{ctrl}^{abs}$ is attached to one of the arms chosen arbitrarily (the left one in our case) by a virtual link (dashed orange in the figure) and it is described in the absolute reference frame $\mathcal{F}_{ref}^{abs}$, which is fixed in the world. The relative task describes the pose of one end-effector $\mathcal{F}_{ctrl}^{rel}$ in the frame of the other one $\mathcal{F}_{ref}^{rel}$. This representation allows to consider as relative task the object deformation, and as absolute task, the object's pose. Then, for operations which solely imply deformation or shaping, only the relative task matters, "freeing" the absolute task's 6 DOF. On the contrary, for rigid object manipulation consisting only in translating or orienting, the relative task's 6 DOF are "freed". More formally, two alternatives are possible ($k = 6$ or $k = 12$ DOF), as one can either control only one of the poses or both the relative and absolute poses:

$$\mathbf{r} = \mathbf{r}^{\mathbf{ref}} \in \mathbb{R}^6 \quad \text{or} \quad \mathbf{r} = \begin{bmatrix} \mathbf{r}^{\mathbf{abs}} \\ \mathbf{r}^{\mathbf{rel}} \end{bmatrix} \in \mathbb{R}^{12}. \tag{1}$$

with $\mathbf{ref} = \{\mathbf{abs}, \mathbf{rel}\}$. To represent orientations, we use angle-axis vectors $\mathbf{q}_r = [x_r \; y_r \; z_r]$, so $\mathbf{r}^{\mathbf{ref}}$ is defined as:

$$\mathbf{r}^{\mathbf{ref}} = \left[x^{\mathbf{ref}} \; y^{\mathbf{ref}} \; z^{\mathbf{ref}} \; x_r^{\mathbf{ref}} \; y_r^{\mathbf{ref}} \; z_r^{\mathbf{ref}}\right]^\top \in \mathbb{R}^6. \tag{2}$$

Another important aspect is the perception of the object. In our work, the robot has no knowledge of the object's material or characteristics, except for its color (needed for visual extraction) and current shape. At each iteration, to represent the object shape, we use the contour seen by the RGB-D camera. We extract the target contour (to be reached) from one of few previously saved images of the same object, held by the robot in various positions/shapes.

It is worth discussing the limits of the contour representation. Using 3D contours as representation will make it impossible to apply any deformation out of the image plane or manage self-occlusions. Furthermore, the target contour is reachable, e.g., it should not require changing from convex to concave bending. We show a case of unreachable target in our experiments video, linked in Section V.

We denote the visible contour, composed of the 3D metric coordinates of p points, $\mathbf{c} \in \mathbb{R}^{3p}$. Working with a task of such high dimension (300 if $p = 100$ points) makes

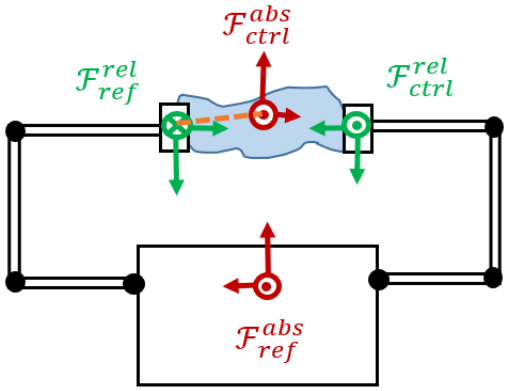

Fig. 2: Schema of the task representation. The relative task (green) describes the motion of $\mathcal{F}_{ctrl}^{rel}$ in relation to the relative frame $\mathcal{F}_{ref}^{rel}$, i.e. the right end-effector in relation to the left one. The absolute task (red) describes the motion of $\mathcal{F}_{ctrl}^{abs}$, in relation to the absolute frame $\mathcal{F}_{ref}^{abs}$. The dashed line is the virtual link between the end-effector and $\mathcal{F}_{ctrl}^{abs}$.

it complex to control the largely lower number of DOF ($k = \{6, 12\}$) of our robot. To solve this underactuated problem, we encode the contour $\mathbf{c}$ into a smaller feature vector $\mathbf{s} \in \mathbb{R}^k$, with k the dimension of the required robot pose. Furthermore, we consider that the mapping between feature variation $\delta\mathbf{s}$ and robot pose variation $\delta\mathbf{r}$ is linear, through what we refer to as the interaction matrix $\mathbf{L} \in \mathbb{R}^{k\times k}$:

$$\delta\mathbf{s} = \mathbf{L}\delta\mathbf{r} \tag{3}$$

In the rest of the paper, we assume the following:

- The object is already grasped by the two robot end-effectors.
- The tracked part of the object stays entirely visible throughout the manipulation, and can be represented by closed contours $\mathbf{c}$.
- The object target shape and (when applicable) pose are reachable, i.e. the tracked part of the object can physically be deformed to (and placed at) the target.

Our framework (outlined in Fig. 3) operates as follows.

First, we control the robot end-effectors (initially in open loop), to collect a sequence of images, with the corresponding extracted contours $\mathbf{c}$ and robot poses $\mathbf{r}$. These are stored in matrices $\mathbf{C}$ and $\mathbf{\Delta R}$, respectively. Then, we perform a PCA (Principal Component Analysis) on the sequence of contours $\mathbf{C}$, to encode the current contour $\mathbf{c}$ into a smaller k-dimensional feature vector $\mathbf{s}$, via projection operator $\mathbf{U}$. This reduces the task shape dimension to the number of required DOF. We also use these sequences of object contours and corresponding robot poses, to estimate the inverse of the interaction matrix $\mathbf{L}$, which maps feature variations to robot pose variations according to (3)). Since linear mapping (3) is only valid locally, we must limit the motions to small displacements and cannot drive the manipulators to the final target right away. Therefore, at every iteration i, we compute a local target $\mathbf{c}_\mathbf{i}^*$, via a linear interpolation between current and target contours ($\mathbf{c}$ and $\mathbf{c}^*$, respectively). We encode this $\mathbf{c}_\mathbf{i}^*$ into a feature vector as well, denoted $\mathbf{s}_\mathbf{i}^*$. We use the inverse of the interaction matrix $\mathbf{L}^{-1}$ to compute the robot pose variations $\delta\mathbf{r}$, required to drive the object to the local target, $\mathbf{s}_\mathbf{i}^*$. We then send $\delta\mathbf{r}$ to our dual-arm controller [24], which relies on hierarchical inverse kinematics, to compute the robot joint commands. The whole process is repeated, by continuously updating data matrices $\mathbf{C}$ and $\mathbf{\Delta R}$ at each iteration i, until the object reaches the target shape and/or position. While previous works made use of these different techniques (i.e. cooperative task representations, image Jacobian estimation), our work presents a promising approach to control deformable objects in $\mathbb{SE}(3)$, without considering specific actions or objects (e.g., cables or clothes).

IV. OUR METHOD

In this Section, we detail each of the modules which compose our framework (again, refer to Fig. 3).

A. Image processing for object contour extraction

We use an Intel Realsense D435 camera, which looks at the robot end-effectors and at the object from a fixed (in the world) position. Both the RGB and depth images are used to obtain a sampled, ordered contour of the object in 3D at each iteration. The different steps of the contour extraction are presented in Fig. 4.

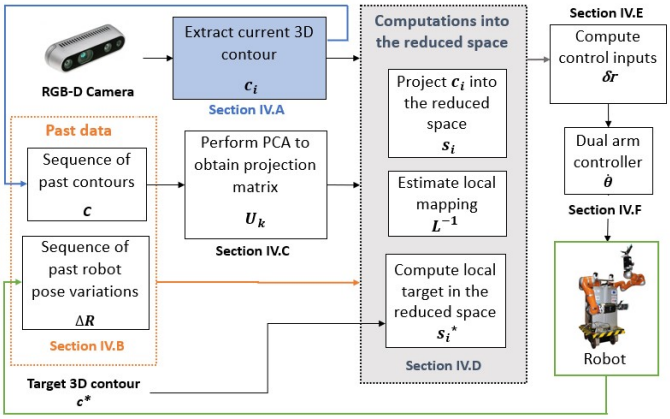

Fig. 3: Overview of our framework. The robot poses and object contours are stored in matrices $\mathbf{\Delta R}$ and $\mathbf{C}$, respectively. At each iteration, PCA yields the projection matrix $\mathbf{U}$, for extracting the feature vector $\mathbf{s}$ from contour $\mathbf{c}$, and the local target $\mathbf{s}_\mathbf{i}^*$ from contour $\mathbf{c}_\mathbf{i}^*$, obtained by linear interpolation between $\mathbf{c}$ and $\mathbf{c}^*$. We also project the contour variation matrix $\mathbf{C}$ to estimate the inverse interaction matrix $\mathbf{L}^{-1}$. We use $\mathbf{L}^{-1}$ to compute the robot desired pose variations $\delta\mathbf{r}$ needed to drive $\mathbf{s}$ to $\mathbf{s}_\mathbf{i}^*$. These $\delta\mathbf{r}$ are finally sent to the dual-arm controller, which computes the robot joint commands $\dot{\boldsymbol{\theta}}$.

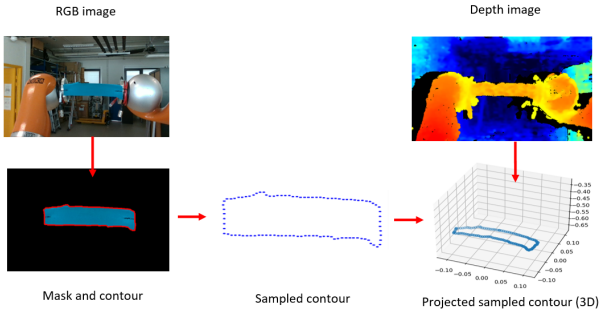

Fig. 4: Steps of the image processing for object contour extraction (here, a rectangular sponge). The 2D RGB image of the camera is used to extract a mask of the object, which in turn lets us obtain its contour. This contour is sampled into a constant number of pixels, which are consistently spaced and ordered. These pixels are then projected as 3D points using their depth and the camera intrinsic parameters.

To simplify the image processing algorithm – which is not the main scope of our work – we have only considered blue objects or blue parts of the objects. After thresholding in the HSV space, we obtain the points defining the contour of the object. The contour is then sampled into a given number of points p; it is essential for the contour points to be uniformly spaced and, more importantly, always ordered the same way from one contour to the next. Namely, the indexes of the points in $\mathbf{c}$ must correspond in $\mathbf{c}_i$ and $\mathbf{c}_{i+1}$, for the PCA

to extract the accurate contour variations. We chose to order the contours in a clockwise direction starting from the left end-effector. Once the contour is sampled and ordered, it is projected in 3D metric coordinates using the pixel depth and the camera intrinsic parameters.

The output of this module, at each iteration i, is vector:

$$\mathbf{c}_i = [x_i^1, \cdots, x_i^p, y_i^1, \cdots, y_i^p, z_i^1, \cdots, z_i^p]^\top \in \mathbb{R}^{3p}. \quad (4)$$

B. Generating a sequence of contours and robot poses

The next step consists in generating a sequence of contours and corresponding robot poses. These are respectively the contour variation matrix $\mathbf{C}$ and the robot pose variation matrix $\mathbf{\Delta R}$, which are needed for two purposes: to extract the contour's principal components, and to estimate the inverse interaction matrix. Matrices $\mathbf{C}$ and $\mathbf{\Delta R}$ are built on a sliding window containing the most recent data: at each iteration, the oldest data in $\mathbf{C}$ and $\mathbf{\Delta R}$ are removed and replaced by the current data.

To build matrices $\mathbf{C}$ and $\mathbf{\Delta R}$, we collect the last $M+1$ samples of contours $\mathbf{c}$ and robot poses $\delta\mathbf{r}$ (so as to get M variations) while the robot moves. While after $M+1$ iterations ($i > M+1$) the robot autonomously moves using the designed control law, for the first $M+1$ iterations we let it move in open-loop. We do this by executing a series of small motions to deform the object, exciting one by one each considered DOF as shown in Fig. 5. The magnitude of these motions is chosen arbitrarily so as to sufficiently vary the contour; we set ± 0.02 m for the translations and $\pm\frac{\pi}{6}$ rad for the rotations. The relative motions should be reduced for stiffer or more fragile objects, to avoid breaking them. We decided to move along each robot DOF in both directions ($\pm$) just once yielding $M = 2k$. M should not be too small, to ensure that $\mathbf{L}^{-1}$ is not under-determined, and not too large, so that the model estimation is local and initialized quickly.

To build matrix $\mathbf{C}$, we stack all the $\mathbf{c_i}$:

$$\mathbf{C} = [\mathbf{c}_0, \cdots, \mathbf{c}_M] \in \mathbb{R}^{3p \times M+1}. \quad (5)$$

Building $\mathbf{\Delta R}$ is a bit more cumbersome. We must obtain the robot pose variation $\delta\mathbf{r}_i$ between consecutive iterations $i-1$ and i. For the translations, we simply subtract the components. For the orientations, we transform rotation vectors $\mathbf{q}_{r_i}^{\mathbf{ref}}$ and $\mathbf{q}_{r_{i-1}}^{\mathbf{ref}}$ to quaternions, apply the quaternion difference and then transform back to the rotation vector representation.

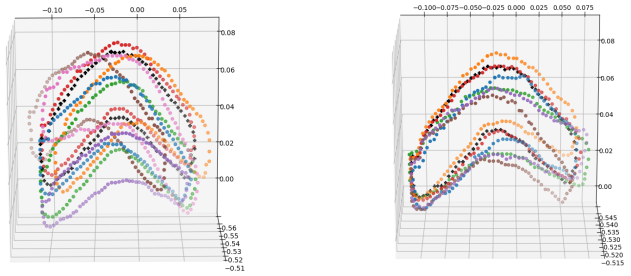

Fig. 5: Initialization motions of the object (here, a rectangular sponge) for the absolute (left) and relative (right) task. Here, each of the $k = 12$ DOF are stimulated: 3 translations and 3 rotations around the initial position (black) for each task.

Finally we stack all $\delta\mathbf{r}_i$, to compose robot pose variation matrix:

$$\mathbf{\Delta R} = \begin{bmatrix} \delta\mathbf{r}_1 & \dots & \delta\mathbf{r}_M \end{bmatrix} \in \mathbb{R}^{k \times M}. \quad (6)$$

C. Principal Component Analysis

Even for a small dataset and small number of contour samples, the dimension of matrix $\mathbf{C}$ is too high compared to the robot DOF (for $M = 24$ and $p = 100$ points: $\mathbf{C} \in \mathbb{R}^{300 \times 25}$). Hence, we perform a PCA on $\mathbf{C}$, to reduce its dimension. First, we shift each column of $\mathbf{C}$ by the mean of all columns, $\bar{\mathbf{c}} = \frac{\sum \mathbf{c}_j}{M+1}, j = 0, ..., M$, to obtain:

$$\mathbf{C}_m = [\mathbf{c}_0 - \bar{\mathbf{c}}, \cdots, \mathbf{c}_M - \bar{\mathbf{c}}] \in \mathbb{R}^{3p \times M+1}. \quad (7)$$

Then, we compute $\mathbf{Q}$, the covariance matrix of $\mathbf{C}_m$. We obtain the eigenvector matrix $\mathbf{U} \in \mathbb{R}^{3p \times 3p}$ by performing a Singular Value Decomposition (SVD) on $\mathbf{Q}$. We select the first k columns of $\mathbf{U}$ to be able to control k DOF of our system (no redundant visual data, [23]).

They define the projection matrix $\mathbf{U}_k \in \mathbb{R}^{3p \times k}$. These columns correspond to the k principal components (with highest variances) in the dataset, and therefore determine the directions of highest variability in the data.

At each iteration i, the reduced feature vector is then:

$$\mathbf{s}_i = \mathbf{U}_k^\top (\mathbf{c}_i - \bar{\mathbf{c}}) \in \mathbb{R}^k. \quad (8)$$

D. Estimation of the Inverse Interaction Matrix

The next step consists in estimating the inverse interaction matrix $\mathbf{L}^{-1}$ needed for control. Recall that the interaction matrix is the linear mapping between feature variation $\delta\mathbf{s}$ and robot pose variation $\delta\mathbf{r}$ (see (3)). This matrix is unknown for a non-rigid object, and it should be inverted to control the robot pose.

To this end, we project the contour matrix $\mathbf{C}$ into the reduced space as follows:

$$\mathbf{S} = \mathbf{U}_k^\top \mathbf{C}_m \in \mathbb{R}^{k \times M+1}, \quad (9)$$

and then derive the features variation over the M-dimensional window:

$$\mathbf{\Delta S} = \begin{bmatrix} \mathbf{S}_1 - \mathbf{S}_0 & \dots & \mathbf{S}_M - \mathbf{S}_{M-1} \end{bmatrix} \in \mathbb{R}^{k \times M}. \quad (10)$$

Finally, the inverse interaction matrix is given by:

$$\mathbf{L}^{-1} = \mathbf{\Delta R} \mathbf{\Delta S}^+ \in \mathbb{R}^{k \times k} \quad (11)$$

With $\mathbf{\Delta S}^+$ the pseudo-inverse of $\mathbf{\Delta S}$.

E. Controlling the robot pose

Linear approximation (3) is only valid locally. Therefore, a control law based on $\mathbf{L}^{-1}$ cannot guarantee convergence if the initial and final contours are too far. To solve this issue, we design a local target contour $\mathbf{c}_i^*$ at each iteration, via linear interpolation:

$$\mathbf{c}_i^* = \frac{\mathbf{c}^* - \mathbf{c}_i}{n} \quad (12)$$

with n big enough to ensure small displacements. Correspondingly, we can project $\mathbf{c}_i^*$ into the feature vector space to obtain the local target feature vector:

$$\mathbf{s}_i^* = \mathbf{U}_k^\top (\mathbf{c}_i^* - \bar{\mathbf{c}}) \in \mathbb{R}^k \tag{13}$$

The desired robot pose variations are then computed via:

$$\delta \mathbf{r}_i = \mathbf{\Lambda} \mathbf{L}_i^{-1} (\mathbf{s}_i^* - \mathbf{s}_i) \in \mathbb{R}^k \tag{14}$$

with $\mathbf{\Lambda} \in \mathbb{R}^{k \times k}$ a diagonal matrix of control gains. This feedback controller guarantees asymptotic convergence of $\mathbf{s}_i$ to $\mathbf{s}_i^*$, in the ideal case that $\mathbf{L}^{-1}$ is perfectly estimated, as proved using the Lyapunov criterion in [22].

F. Controlling the robot joints

To map the desired robot pose variations $\delta \mathbf{r}$ to the robot joint velocities $\dot{\theta}$, we use the cooperative task representation, outlined in Sect. III, and recalled here (for further details, refer to [24], [25]). We consider $\delta \mathbf{r} \in \mathbb{R}^{12}$ as the variation of $\mathbf{r}$ defined in (1). From $\mathbf{J_{lef}}$ and $\mathbf{J_{rig}}$ (the Jacobian matrices of the left and right arm in $\mathcal{F}_{ref}^{abs}$), we derive the cooperative task Jacobian matrices:

$$\begin{aligned} \mathbf{J_{abs}} &= \begin{bmatrix} \frac{1}{2}\mathbf{J_{lef}} & \frac{1}{2}\mathbf{J_{rig}} \end{bmatrix} \\ \mathbf{J_{rel}} &= \begin{bmatrix} -\mathbf{\Psi \Omega J_{lef}} & \frac{1}{2}\mathbf{\Omega J_{rig}} \end{bmatrix}, \end{aligned} \tag{15}$$

where

$$\mathbf{\Psi} = \begin{bmatrix} \mathbf{I} & -\mathbf{\Upsilon_{lef}} \\ 0 & \mathbf{I} \end{bmatrix}, \quad \mathbf{\Omega} = \begin{bmatrix} \mathbf{\Phi_{lef}} & 0 \\ 0 & \mathbf{\Phi_{lef}} \end{bmatrix}, \tag{16}$$

with $\mathbf{\Upsilon_{lef}}$ the skew-symmetric matrix of the position of the left arm in the relative task frame, and $\mathbf{\Phi_{lef}}$ the rotation matrix of the left arm in $\mathcal{F}_{ref}^{abs}$.

If both the relative and absolute tasks have to be satisfied at the same time ($k = 12$), a relevant choice is to prioritize the relative task, to avoid undesired internal stress which may damage the object and/or the robot. Taking into account joint (position, velocity and acceleration) limits, the highest priority task is solved through:

$$\begin{aligned} \dot{\boldsymbol{\theta}}_1 \in \min_{\dot{\boldsymbol{\theta}}} & ||\mathbf{J_{rel}} \dot{\boldsymbol{\theta}} - \delta \mathbf{r^{rel}}||_2 \\ & \text{subject to: joint limits.} \end{aligned} \tag{17}$$

The obtained solution vector $\dot{\boldsymbol{\theta}}_1$ provides a null-space condition to the second task, so the final joint velocities to be sent to the robot are:

$$\begin{aligned} \dot{\boldsymbol{\theta}} \in \min_{\dot{\boldsymbol{\theta}}} & ||\mathbf{J_{abs}} \dot{\boldsymbol{\theta}} - \delta \mathbf{r^{abs}}||_2 \\ & \text{subject to: joint limits,} \\ & \mathbf{J_{rel}} \dot{\boldsymbol{\theta}} = \mathbf{J_{rel}} \dot{\boldsymbol{\theta}}_1. \end{aligned} \tag{18}$$

We add the relative task solution as a constraint to the absolute task to avoid interference. Using this formulation, the absolute task error is minimized as long as the resulting joint velocity vector provides the best solution for (17).

If only the relative task has to be satisfied to deform the object, and we "free" the object pose, $k = 6$ and we can simply apply (17) and set $\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\theta}}_1$.

V. RESULTS

To validate the method, we conducted many experiments, during which we tested different initial and target shapes, as well as different objects. A video of experiments and the source code are accessible at `https://gite.lirmm.fr/csaghour/rkcl-bazar-flex-app.git`.

At the beginning of each experiment, $M = 24$ predefined motions are executed. We set the number of contour points to $p = 100$ and the number of intermediate targets to $n = 20$. The control gain matrix was tuned experimentally to $\mathbf{\Lambda} = 0.07\ \mathbf{I}_k$, to obtain motions neither too fast nor too slow.

The experiments using a rectangular sponge demonstrate successful convergence to the different final targets (in red in the figure), involving the deformation, as well as change in the orientation and position of the object (see Fig. 6).The evolution of the cooperative task frames is shown in the third column of Fig. 6.

We define the task error as the difference between the current contour and the final target contour:

$$\epsilon = \|\mathbf{c}^* - \mathbf{c}_i\| \tag{19}$$

This metric decreases until reaching an acceptable threshold (see the last column of Fig. 6).

We also performed experiments with the other objects, soft (a differently shaped sponge, a plastic glove) and rigid (a cardboard box). The method proves to be effective with those as well, as shown in Fig. 6, despite the rough contours for an object of more complex shape like the glove. For this experiment, noise was detected in the images, explaining the wide oscillations of the error observed in Fig. 6. Nevertheless, the controller converged to the final target shape, proving the robustness of the method. Again, tasks like bending, compressing, rotating and translating were successfully carried out, proving the large spectrum of applications of our controller.

Regarding the manipulation of the cardboard box, which we considered as rigid, the experiment consists only in orienting and translating the object. Through this experiment, we show that our framework can be used for manipulating a wider spectrum of objects, including rigid ones. In this case, only $k = 6$ DOF are needed for the absolute task, and the 6 other DOF are freed. These could be used, for example, for obstacle avoidance.

We can, on the contrary, choose to only deform the object and not move its position and orientation in the space. Then, only the relative task is needed, therefore reducing the number of DOF to $k = 6$. Such an experiment is presented in Fig. 7. To achieve this, the target is aligned so as to match with the fixed end-effector during the experiment. This allows us to shape the object with no concern on the orientation and position. The error is then computed between the aligned contour and the current contour.

VI. CONCLUSIONS

This paper presents a complete method for dual-arm shaping of soft objects in 3D and in real time. We use a

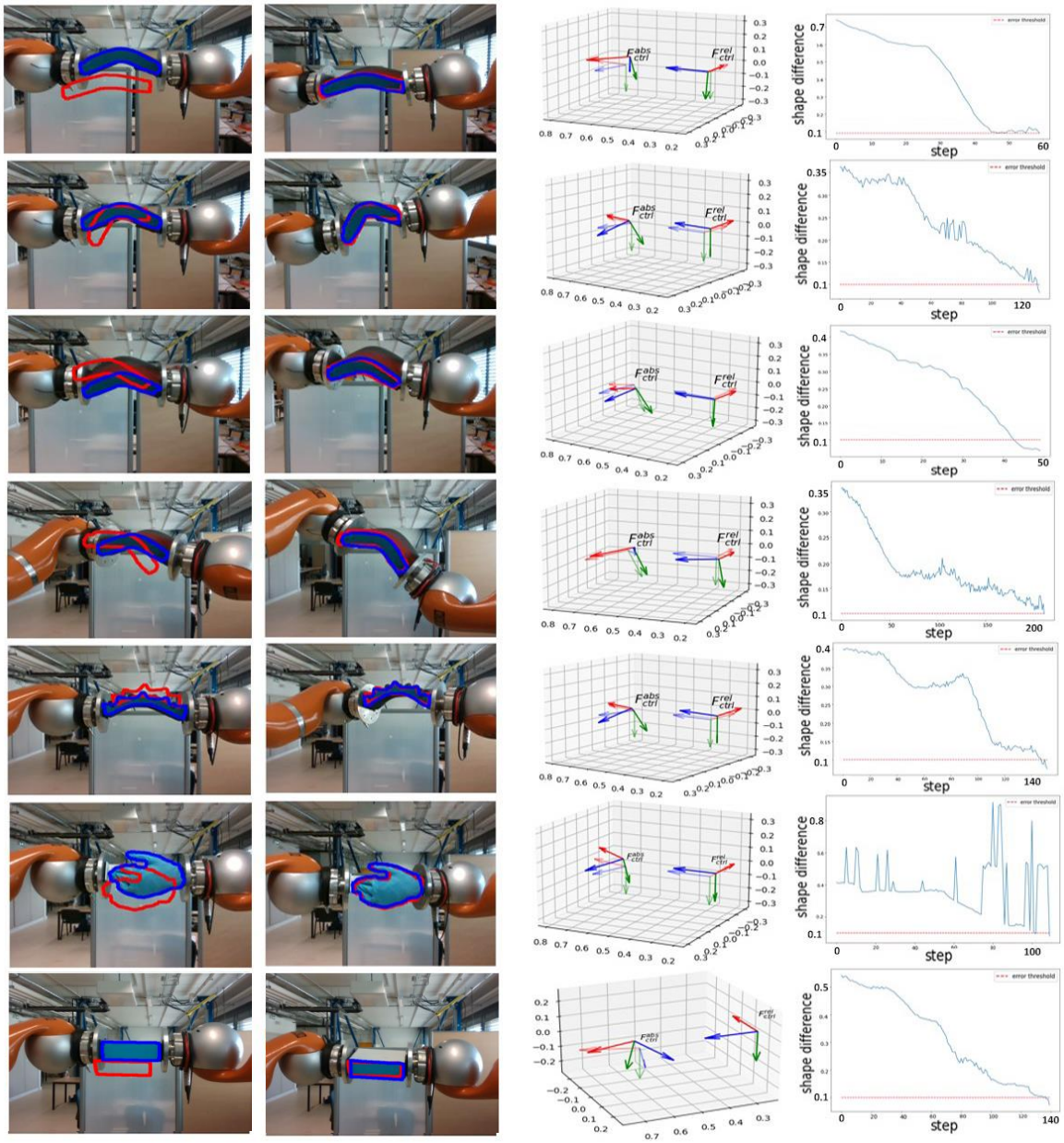

Fig. 6: The two first columns are the robot camera view of seven experiments. Starting from different initial configurations (in blue on the first column), we reach the final targets (in red). The third column shows the initial (transparent) and final poses of the absolute frame F_{ctrl}^{abs} and relative frame F_{ctrl}^{rel}. The last column shows the evolution of the shape difference defined in (19) according to the iteration step. The experiments include, from top to bottom: four different shaping of a rectangular sponge, then shaping of different objects: a crown-shaped sponge, a plastic glove, and a cardboard box. For the cardboard box, the relative frame F_{ctrl}^{rel} is freed, so only the absolute task F_{ctrl}^{abs} evolves. Noise in the background during the experiment with the glove causes the shape difference to oscillate greatly, but the error still decreases.

few initializing motions to both execute a PCA to reduce the dimensions of the visual data and to compute an interaction matrix, and iteratively obtain robot control inputs to reach a final target shape. The framework is able to handle objects of different geometry and materials, soft and rigid alike without any prior knowledge, although we experimented on a limited variety of objects. Nevertheless, the 3D target shapes could be successfully reached in these experiments.

From a theoretical viewpoint, our work raised questions on the representations, while considering the numerous DOFs needed to shape and move the objects. We believe that using a relative task to describe and control exclusively deformations is a promising approach. Our work addresses different challenges regarding 3D motion representation, while considering the limitations of the objects' geometrical representation. In an industrial context, we aim at proposing

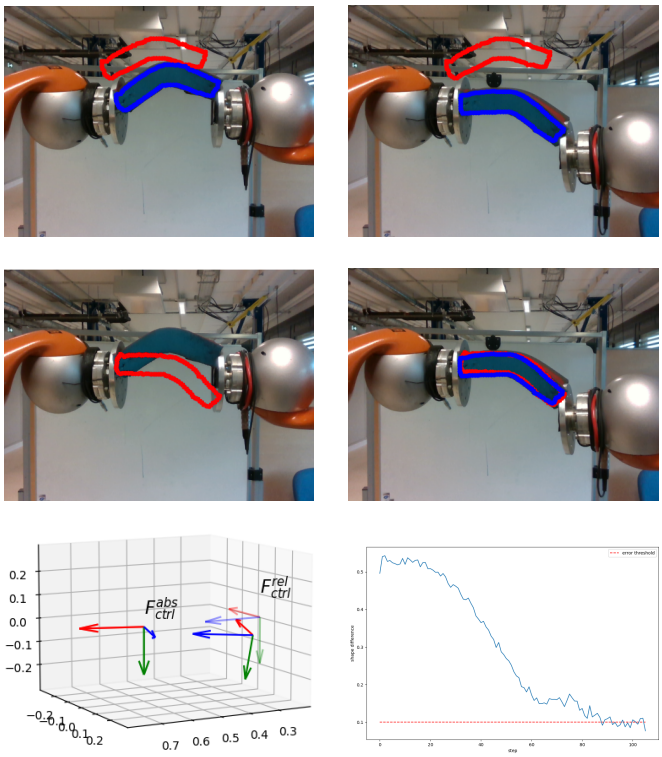

Fig. 7: Experiment with the rectangular sponge, reaching the target configuration from the initial position (in blue, left) while only controlling the relative task. The target (in red, top row) is aligned with the relative reference frame F_{ref}^{rel} (the left end-effector) so that the aligned target (in red, bottom row) can be achieved with only the relative task being controlled. On the bottom row are shown the evolution of the cooperative task frames (left) and of the error (right).

solutions for tasks such as "deform and place".

One of the main aspects to improve is vision. Contours are not well suited to describe volumetric objects and limit the deformations to those of the visible surface. Yet, a 3D volumetric sampling might be challenging to achieve, since the success of our method largely depends on the consistency of the points and on their order from one iteration to the other, which is not assured with simple point clouds. Occlusions could also be difficult to handle.

As future work, the addition of an adaptive deformation model would be interesting, including material parameters estimation. Such a hybrid controller would be able, for instance, to avoid non recoverable deformation of the object and to ensure target reachability.

REFERENCES

[1] Schlechter, A., and Dominik H., "Manipulating deformable linear objects: Manipulation skill for active damping of oscillations." IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2002.

[2] X. Li, X. Su, Y. Gao and Y. Liu, "Vision-Based Robotic Grasping and Manipulation of USB Wires," IEEE Int. Conf. on Robotics and Automation (ICRA), 2018.

[3] Shademan, A., Decker, R.S., Opfermann, J.D., Leonard, S., Krieger, A. and Kim, P.C. "Supervised autonomous robotic soft tissue surgery." Science translational medicine: 8.337, 2016.

[4] Zhong, F., Wang, Y., Wang, Z., and Liu, Y. H., "Dual-arm robotic needle insertion with active tissue deformation for autonomous suturing." IEEE Robotics and Automation Letters: 2669-2676, 2019.

[5] Garcia-Camacho, I., Lippi, M., Welle, M.C., Yin, H., Antonova, R., Varava, A., Borras, J., Torras, C., Marino, A., Alenya, G. and Kragic, D., "Benchmarking bimanual cloth manipulation." IEEE Robotics and Automation Letters 5.2: 1111-1118, 2020.

[6] SepLveda, D., Fernández, R., Navas, E., Armada, M., and González-De-Santos, P. Robotic aubergine harvesting using dual-arm manipulation. IEEE Access, 8: 121889-121904, 2020.

[7] Jagersand, M., Fuentes, O. and Nelson, R., "Experimental evaluation of uncalibrated visual servoing for precision manipulation". IEEE Int. Conf. on Robotics and Automation: Vol. 4, 2874-2880, 1997.

[8] Shademan, A., Farahmand, A. and Jagersand, M., "Robust Jacobian estimation for uncalibrated visual servoing," IEEE Int. Conf. on Robotics and Automation: 5564-5569, 2010.

[9] Piepmeier, J.A., McMurray, G.V. and Lipkin, H., "Uncalibrated dynamic visual servoing". IEEE Trans. on Robotics and Automation,: 143-147, 2004.

[10] Agravante, D. J., Cherubini, A., Bussy, A., Gergondet, P. and Kheddar, A., "Collaborative human-humanoid carrying using vision and haptic sensing," IEEE International Conference on Robotics and Automation (ICRA): 607-612, 2014.

[11] Vahrenkamp, N., Böge, C., Welke, K., Asfour, T., Walter, J. and Dillmann, R., "Visual servoing for dual arm motions on a humanoid robot," 2009 9th IEEE-RAS International Conference on Humanoid Robots: 208-214, 2009.

[12] Claudio, G., Spindler, F. and Chaumette, F., "Vision-based manipulation with the humanoid robot Romeo," 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids): 286-293, 2016.

[13] Jin, S., Wang, C., Tomizuka, M., "Robust deformation model approximation for robotic cable manipulation". IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS): 6586-6593, 2019.

[14] Nguyen, A., Do, T.T., Reid, I., Caldwell, D.G. and Tsagarakis, N.G., "V2cnet: A deep learning framework to translate videos to commands for robotic manipulation." arXiv preprint arXiv:1903.10869, 2019.

[15] Wu, Y., Yan, W., Kurutach, T., Pinto, L., and Abbeel, P., "Learning to manipulate deformable objects without demonstrations." arXiv preprint arXiv:1910.13439, 2019.

[16] Herguedas, R., López-Nicolás, G., Aragüés, R. and Sagüés, C. Survey on multi-robot manipulation of deformable objects. In 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA): 977-984, 2019.

[17] Tsurumine, Y., Cui, Y., Uchibe, E., Matsubara, T. Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation. Robotics and Autonomous Systems, 112, 72-83, 2019.

[18] Hu, Z., Han, T., Sun, P., Pan, J., and Manocha, D., "3-D deformable object manipulation using deep neural networks." IEEE Robotics and Automation Letters 4.4: 4255-4261, 2019.

[19] Lagneau, R., Alexandre K., and Maud M., "Automatic shape control of deformable wires based on model-free visual servoing." IEEE Robotics and Automation Letters 5.4: 5252-5259, 2020.

[20] Tang, T., Wang, C., and Tomizuka, M. A framework for manipulating deformable linear objects by coherent point drift. IEEE Robotics and Automation Letters, 3(4): 3426-3433, 2018.

[21] Qi, J., Ma, G., Zhu, J., Zhou, P., Lyu, Y., Zhang, H. and Navarro-Alarcon, D. Contour moments based manipulation of composite rigid-deformable objects with finite time model estimation and shape/position control. IEEE/ASME Transactions on Mechatronics, 27(5): 2985-2996, 2021.

[22] Zhu, J., Navarro-Alarcon, D., Passama, R., and Cherubini, A., "Vision-based manipulation of deformable and rigid objects using subspace projections of 2D contours." Robotics and Autonomous Systems, 142: 103798, 2021.

[23] Chaumette, F., Dombre, E. and Khalil, W., "Visual servoing. In Robot Manipulators: Modeling, Performance Analysis and Control", Chap. 6: 279-336, ISTE, 2007.

[24] Tarbouriech, S., "Dual-Arm control strategy in industrial environments", chapters 2-3. PhD, Université Montpellier, 2019.

[25] Jamisola Jr, R. S., Roberts, R. G., "A more compact expression of relative jacobian based on individual manipulator jacobians". Robotics and Autonomous Systems: 158-164, 2015.