Dual-Arm Shaping of Soft Objects in 3D Based on Visual Servoing and Online FEM Simulations

Célia Saghour, David Navarro-Alarcon, Philippe Fraisse, Andrea Cherubini

Dual-Arm Shaping of Soft Objects in 3D Based on Visual Servoing and Online FEM Simulations

Célia Saghour[1], David Navarro-Alarcón[2], Philippe Fraisse[1] and Andrea Cherubini[3]

Abstract

In this work, we propose a vision-based and Finite Element Method (FEM) based controller to automate the 3D shaping of soft objects with dual-arm robots. Our controller relies on a data-based approach to learn how the robot's actions result in object deformations, while also running FEM-based simulations to infer the shape of the whole body. These model-based simulations are used to generate initial shape data, allowing to extract visual features through a Principal Component Analysis and thus estimate the interaction matrix of the object-robot system. In contrast with most existing shape servoing controllers, our new model-based approach continuously predicts the object deformations produced by the robot, which are then compared to the visually observed deformation feedback. This iterative process enables to correct the deformed mesh model before updating the interaction matrix. To validate this new control methodology, we present a detailed experimental study with a dual-arm robot and different soft objects, which showcases the performance of our automatic shaping framework.

Keywords

Shape control; Dual-arm manipulation; Soft objects; Finite Element Method; Jacobian estimation

Introduction

Cables, organic matter like tissues or vegetables, polymer foams, fabrics, even metal parts are all deformable. Manipulating such type of materials with robots typically involves two important aspects: (i) the use of active motions that produce deformations onto the object, and (ii) the use of visual feedback to track and control the object's shape (Zhu et al. 2022). Traditionally, the shape servoing problem has been tackled in two dimensions, i.e., by only controlling the object's contour as perceived by a monocular camera. The main complications have been the difficulty to continuously monitor the changes in the complete surface of an object, which is typically occluded during its manipulation (Qi et al. 2024).

To address this issue, in this paper we propose a new framework that enables to actively control the object's morphology in three dimensions (3D), see Fig. 1. For that, we derive an efficient control method that exploits RGB-D vision and a mechanical model of the object to guide the shaping actions in 3D. The model is constructed based on geometric and physical parameters, and is used to predict the behavior of the object under robotic manipulation. The model is continuously updated based on deviations between the predicted object behavior and the visually measured deformations. This new method builds on our earlier work (Saghour et al. 2023), by now taking into account the whole volume of the manipulated body, and not only its partial visible surface (as done in our previous result).

The advantages of our proposed model-based shape servoing controller compared to traditional manipulation approaches can be summarized as follows:

- It provides a consistent 3D representation of the whole object that allows to implement effective Jacobian estimation algorithms that compute the relations between robot motions and deformations, even for occluded parts of the object;
- It provides a way to virtually (hence, safely) collect the shape-pose data that is needed to construct latent feature representations, without the risk of damaging the object during exploratory testing motions;
- It provides the user with an intuitive interface that allows to specify the target shape of the task as well

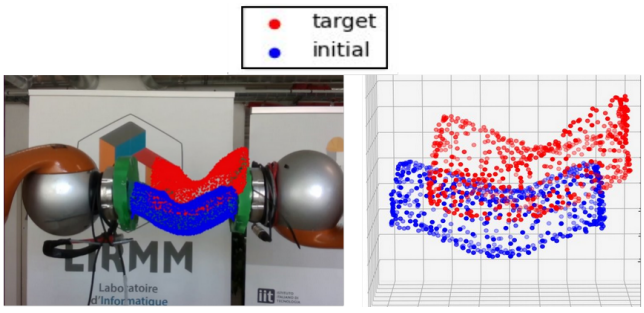


Figure 1. The goal of this work is to control a dual arm robot so that the current mesh (blue) reaches the target mesh (red) in 3D.

[1]LIRMM, Université de Montpellier, CNRS Montpellier, France.
[2]The Hong Kong Polytechnic University, Kowloon, Hong Kong.
[3]Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, 1, rue de la Noe, 44321 Nantes, France.

Corresponding author:
Célia Saghour, Université de Montpellier, CNRS Montpellier, LIRMM, 161 Rue Ada, 34095 Montpellier, France.
Email: celia.saghour@lirmm.fr

as to visualize the planned and predicted behavior of the system beforehand.

Related Work

While some works favor physics-based modeling for accuracy in the prediction of the behavior of soft objects, other rely on data to track and control the deformation, without prior knowledge of the object behavior. Both methods have their benefits and drawbacks. Data-based methods require training for every new object. On the other hand, model assumptions and parameters may be inaccurate for describing deformation.

Various researchers have addressed the development of data-based approaches for soft object manipulation. For example, in (Shin et al. 2019) the authors propose two predictive control algorithms (reinforcement learning and learning from demonstration) to manipulate tissues with surgical robots. In (Lee et al. 2022), the robot controller learns to manipulate DLO through self-supervision, via Fully-Convolutional Neural Networks. The method in (Tsurumine et al. 2019) relies on Deep Reinforcement Learning to make a humanoid robot manipulate clothes. The authors of (Hu et al. 2019) use a Deep Neural Network to map the end-effectors motion to the object deformation. In (Seita et al. 2021), reinforcement learning is used to learn to manipulate 1D, 2D, and 3D deformable objects into goal configurations.

Model-based methods typically construct geometric and/or physical models of the object to predict its deformation in response to external actions, either in the form of applied forces or input positions. The authors of (Ruan et al. 2018) rely on a geometric model and on gradient descent to predict the shape of deformable objects, given the robot end-effectors motion. In (Aghajanzadeh et al. 2022b) and (Aghajanzadeh et al. 2022a), an ARAP (As Rigid As Possible) model characterizes the behavior of deformable linear objects (DLO) in 2D. Some other works use FEM to describe the behavior of the manipulated objects; For example, in (Ficuciello et al. 2018) an offline visual and force algorithm is used to estimate the FEM parameters, and then invert the FEM to compute the necessary robot motions to shape the object. The authors of (Zimmermann et al. 2021) propose a FEM simulation-based open-loop trajectory planner for dual arm manipulation of soft objects. Other models like the Cosserat theory (Azad et al. 2023), or a mass-spring model can also be used (Andronas et al. 2021) to model and control the deformation of a soft object. In (Yu et al. 2023), a simplified energy-based model (Discret Elastic Rod) is used in addition to a Jacobian mapping between the end-effectors and the DLO centerline. In (Makiyeh et al. 2023), a mass-spring model is used along Fourier-based features to control 3D deformations of surfaces.

Other works address the manipulation of non-rigid objects through visual servoing techniques (Chaumette 2007). In contrast with data-driven approaches, adaptive visual servoing relies on the local estimation of the mapping between the robot inputs and the computed visual features. The main challenge with non-rigid objects is to encode the object shape with image features, which should be continually tracked during manipulation. In (Navarro-Alarcon et al. 2016), an adaptive deformation model for elastic materials is computed from feature points of interest. The authors of (Lagneau et al. 2020) shape wires in 3 dimensions, by extracting visual features from a B-spline model. The visual servoing framework presented in (Shetab-Bushehri et al. 2022) relies on a 3D lattice representation, linked to the object by geometrical constraints. The authors of (Zhu et al. 2020) encode the object via Principal Component Analysis (PCA) on image contours the object. Similarly, (Qi et al. 2022) uses contour moments as a state representation of the manipulated object. In (Zhou et al. 2024), the authors use a topological latent state obtained through an auto-encoder as a state representation, for collaborative manipulation of DLO.

Our aim in this work is to develop a shape servoing framework that unites the benefits of data-driven methods (that learns the shape-motion mapping and the visual features on-the-fly) with those of model-based methods (that provides a consistent geometric representation of the whole body and a good approximation of the shape behavior when subjected to external disturbances).

This new approach enables to control 3D shapes with little sensor feedback (only a partial view of object observable with one single vision sensor), and does not require expressing any explicit force constraint on the model, since the constraints are entirely *geometric*. This feature makes our method intuitive, and the estimation of the inverse Jacobian relying only on past data makes it adaptable to the behavior of the object without the need of material parameters nor of precise modeling of the object behavior.

We also show that we can exploit the tools offered by Simulation Open Framework Architecture (SOFA)* (Faure et al. 2012) in different steps of the framework (to generate data without using the robot, select a target, monitor the stress, visually correct the model or even reconstruct meshes) to reduce the need for real life manipulation, preferring simplified and intuitive use of simulations instead.

Overview of our Framework

Our goal is to generate a sequence of commands for a dual arm robot to deform and drive the current mesh nodes $\mathbf{m}$ to the target mesh nodes $\mathbf{m}^*$. The target mesh can be chosen in the simulation, e.g., via keyboard commands.

We assume the following *hypotheses*:

- The object is already grasped by the robot, and the positions and orientation of both end-effectors, denoted $\mathbf{r}_l$ and $\mathbf{r}_r$ are known.
- The effect of gravity can be neglected (because the object is rigid, or light). Hence, given to the end-effectors poses $(\mathbf{r}_l, \mathbf{r}_r)$, precise knowledge of the material parameters is not necessary to simulate the object behavior (see Sec. Material Parameters Effect on Simulations).
- Prior to manipulation, the mesh of the grasped object is known a priori, or it can be reconstructed.

*https://www.sofa-framework.org/

- The color of the object can be segmented from that of the end-effectors.
- The camera pose in the robot frame is known.
- The object is at least partly in the camera field of view throughout manipulation.

In contrast with our previous work (Saghour et al. 2023), here we use simulations to track the shape of the object online. The object material parameters are not necessary for manipulation, since the constraints are considered *geometric*; yet, they can be used to monitor the deformation. The control framework is shown in Fig.2. We use tetrahedral mesh nodes $\mathbf{m}$ to represent the shape of the manipulated object, defined as follows:

$$\mathbf{m} = \begin{bmatrix} m_{1,x} & m_{1,y} & m_{1,z} \\ \vdots & \vdots & \vdots \\ m_{n,x} & m_{n,y} & m_{n,z} \end{bmatrix} \in \mathbb{R}^{n\times 3}. \quad (1)$$

In addition, we use visual mesh nodes $\mathbf{m}^v$ from triangular, surface elements to represent the visible part of the object.

At each iteration i, we deform the object mesh in a simulator, according to $(\mathbf{r}_l, \mathbf{r}_r)$. The simulation runs a few steps until a state of quasi-equilibrium is reached. This configuration yields the new mesh nodes. During manipulation, the point cloud of the new state of the object $\mathbf{pc}_{i+1}$ is retrieved through visual processing and used to correct the meshes in the simulation. To first initialize the data matrices, we collect a sequence of shapes (mesh nodes) with corresponding robot poses, all in simulation. These are stored in matrices $\mathbf{M}$ and $\Delta\mathbf{R}$, respectively. These matrices are then updated at each step of the manipulation with the latest shape (from simulation) and robot pose variation (from the robot).

Then, we conduct PCA on $\mathbf{M}$, to obtain the projection matrix $\mathbf{U}_k$, encoding the features in the reduced space of k highest variance eigen values. Both the current and intermediary target mesh nodes $\check{\mathbf{m}}_i$ and $\check{\mathbf{m}}_i^*$ are projected to encode the shape into a smaller number of k features, respectively $\mathbf{s}_i$ and $\mathbf{s}_i^*$. We consider that the mapping between feature variation $\delta\mathbf{s}$ and robot pose variation $\delta\mathbf{r}$ is **locally linear**, via interaction matrix $\mathbf{L} \in \mathbb{R}^{k\times k}$:

$$\delta\mathbf{s} = \mathbf{L}\delta\mathbf{r} \in \mathbb{R}^k. \quad (2)$$

We estimate the inverse interaction matrix $\mathbf{L}^{-1}$, mapping robot pose variation to the object's shape variation, and use it to compute the control input $\delta\mathbf{r}$ driving the current shape to the intermediary target. The dual-arm controller computes the necessary robot joint commands, here denoted as $\dot{\theta}$. This process is repeated until the target shape is reached.

Simulating Deformations

We use SOFA to run online FEM simulations of the deformation of the object being manipulated. In the simulation scene, the *deformation model* is the tetrahedral mesh of the object in the robot frame, $\mathcal{M}$. The visual mesh $\mathcal{M}^v$ is imported in SOFA as a *visual model*. The tetrahedral mesh DOF are connected to the visual mesh nodes through SOFA's mapping functions, so that the deformation propagates to the visual model. Accurate mechanical models are complex to define, as they depend on the material properties and are computationally expensive. Thus, instead of trying to estimate a precise mechanical model, we use linear elasticity as a simple approximation of the mechanical behavior of the manipulated objects and the robot end-effectors' poses to run simple and efficient simulations.

As assumed above, if the object is light or stiff enough, the effect of gravity on its shape is negligible compared to the deformation effect produced by the position constraints imposed by the robot manipulators. Then the material parameters (mass, Young modulus and Poisson's ratio) will not have a significant impact on the simulation (see Sec. Material Parameters Effect on Simulations), and we can set realistic default variables. Yet, we can use the knowledge of the parameters to monitor the internal stress, and to assure that the object is being manipulated in the elastic domain and does not undergo irreversible deformations. Then, the mass of the object can be obtained through the force sensors on the end-effectors and the Young modulus and Poisson's ratio included in the simulator FEM model.

Setting-up the Object Mesh

In the following, we denoted a *rigid particle* by $\mathbf{pr}$, which can be moved in the SOFA scene. Specifically, $\mathbf{pr}_l$ and $\mathbf{pr}_r$ are the rigid particles corresponding to the left and right end-effectors, respectively (see Fig. 3). A rigid particle is defined by a position and an orientation as:

$$\mathbf{pr} = [x\ y\ z\ \mathbf{q}], \quad (3)$$

with $\mathbf{q}$ the quaternion representing the end-effector orientation. We define the contact between the end-effectors of the robot and the object by a set of mesh nodes, called *holding nodes*, and representing the contact between the mesh and the end-effectors. These nodes are contained in each of the bounding boxes built around the position of $\mathbf{pr}_l$ and $\mathbf{pr}_r$ and denoted $\mathbf{H}_l$ and $\mathbf{H}_r$ respectively (see Fig.3). Since the object is considered firmly gripped, the mapping is supposed rigid: when the particles are displaced, the attached holding nodes are transformed accordingly, and the rest of the mesh deforms following the FEM. To simulate the deformation caused by the real end-effectors during manipulation, we relate $\mathbf{pr}_l$ and $\mathbf{pr}_r$ to the position and orientation of the actual robot end-effectors $\mathbf{r}_l$ and $\mathbf{r}_r$, or in general:

$$\mathbf{r} = [x\ y\ z\ \boldsymbol{\rho}], \quad (4)$$

with $\boldsymbol{\rho}$ the angle-axis vector representing end-effector orientation. The whole dual-arm robot pose $\mathbf{r}$ is then defined as:

$$\mathbf{r} = [\mathbf{r}_l\ \mathbf{r}_r]. \quad (5)$$

The simulation setup presented in this section can be expanded to any objects for which the meshes are known or acquired beforehand. Some methods that we developed to reconstruct the meshes of DLO are presented in Sec. Simulations for DLO Mesh Reconstruction.

Material Parameters Effect on Simulations

For small displacements (as those applied by our controller), we have seen empirically that the material parameters (both

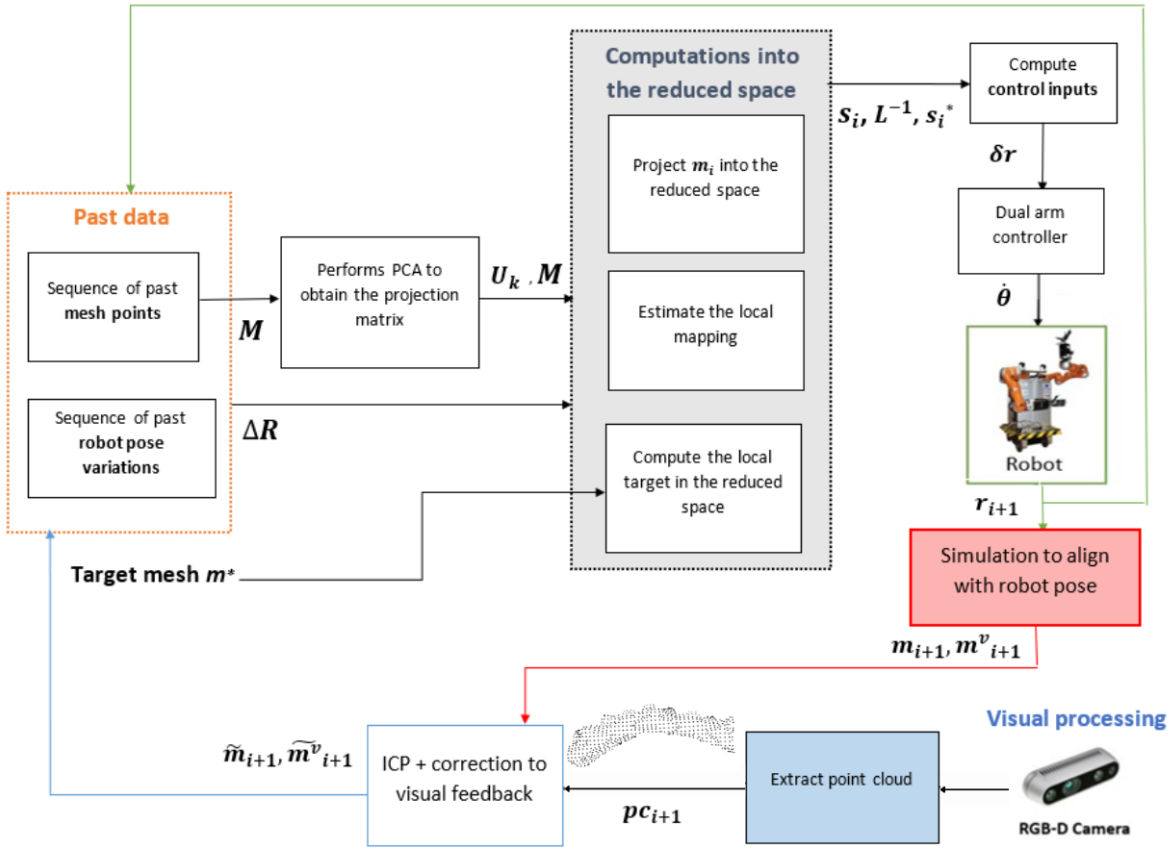


Figure 2. Overview of our framework at each iteration i. Given a sequence of past mesh nodes $\mathbf{M}$ and corresponding robot poses $\Delta\mathbf{R}$, we apply PCA on $\mathbf{M}$, to obtain projection matrix $\mathbf{U}_k$. This is then used to reduce the dimension of $\mathbf{M}$, to estimate $\mathbf{L}^{-1}$, linearly mapping the features variation to the robot pose variation. With $\mathbf{L}^{-1}$, we compute the robot input $\delta\mathbf{r}$ driving current shape $\mathbf{s}_i$ to intermediary target $\mathbf{s}_i^*$ in the reduced space. The robot input is sent to the dual-arm robot controller to obtain the joint command $\dot{\theta}$. Once the command is achieved, the new robot pose $\mathbf{r}_{i+1}$ is sent to the simulator, and the simulation runs until quasi equilibrium. The resulting visual mesh nodes $\mathbf{m}^v_{i+1}$ are compared to the point cloud of the object $\mathbf{pc}_{i+1}$ obtained via visual processing, through Iterative Closest Point (ICP). Correction steps are run until the resulting error is lower than a threshold. Finally, the past data matrices are updated with the new mesh nodes (from the simulation), and the latest robot pose variation (from the robot). The whole process is repeated at iteration $i+1$.

Young's Modulus E and Poisson's ratio ν) given as input to the simulator have a small impact on the mesh deformation. Since the effect of gravity can be neglected (because the object is rigid, or light), the constraints are solely *geometric*. In Fig. 4, we show meshes resulting from simulations, during which a displacement constraint is applied, with different parameters E and ν.

In this example, the displacement is applied on the right end-effector, with a span of 0.02 m for the translation and angles of $\frac{\pi}{10}$rad for the rotation on each axis. We find that for a Young Modulus going from $5e^3$ Pa to $5e^7$ Pa, the different simulations result in a maximum error (norm L2) between the deformed meshes of $5.3e{-}5$m. This confirms that the Young's Modulus does not have a significant impact on the simulated deformation.

Regarding the Poisson's ratio, we try out values from 0.1 to 0.45, which can correspond to materials such as foams, polymers, or metals. The maximum error (norm L2) between the deformed meshes after application of a same displacement constraint is $e_{max} = 0.07$m, corresponding to a ASE (average sample error, $ASE = e_{max}/3n$ with n the number of nodes) of $4.22e^{-5}$m.

The impact of the Poisson's ratio is more visible than that of the Young's Modulus, but it is still quite small when related to the whole shape. These simulations show that for such objects (ones on which gravity can be neglected), the exact knowledge of the Poisson's ratio is not necessary to simulate an approximate behavior of the object, since it does not have *significant* effects on the shape when small displacements are applied.

Initialization

We use simulations to construct the initial sequence of mesh nodes $\mathbf{M}$, and the corresponding robot pose variations $\Delta\mathbf{R}$ for a first estimation of the mapping $\mathbf{L}^{-1}$, as well as for features extraction. The matrix $\mathbf{M}$ is composed of $D+1$ different samples of shapes (deformed meshes), so as to have D variations. To obtain them, we generate random samples of $\mathbf{r}$, with constraints on the end-effectors distance, to avoid stretching the object too much.

Examples of generated shapes are shown in Fig. 5. The $D+1$ resulting meshes $\check{\mathbf{m}}_i \in \mathbb{R}^{3n}$ are stored in matrix $\mathbf{M}$:

$$\mathbf{M} = [\check{\mathbf{m}}_0 \cdots \check{\mathbf{m}}_D] \in \mathbb{R}^{3n(D+1)}. \quad (6)$$

The corresponding poses of the rigid particles $[\mathbf{pr}_l, \mathbf{pr}_r]$ yield the robot pose variation between iterations $i-1$ and i as:

$$\delta\mathbf{pr} = \left[x_i - x_{i-1}\ y_i - y_{i-1}\ z_i - z_{i-1}\mathbf{q}_i \cdot \mathbf{q}_{i-1}^{-1}\right], \quad (7)$$

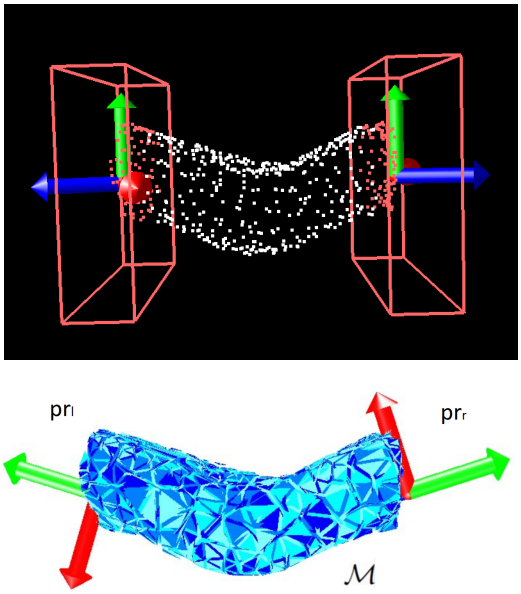


Figure 3. Simulation setup. The nodes of the mesh are shown in white. The frames $\mathbf{pr}_l$ and $\mathbf{pr}_r$ represent the the rigid particles related to $\mathbf{r}_l$ and $\mathbf{r}_r$ respectively, acting on the mesh when transformed accordingly to the displacements of the end-effectors. The red lines represent the bounding boxes around the positions of $\mathbf{pr}_l$ and $\mathbf{pr}_r$. The holding nodes $\mathbf{H}_l$ and $\mathbf{H}_r$ are the red nodes in left and right boxes respectively. The mesh $\mathcal{M}$ is shown in blue on the last row.

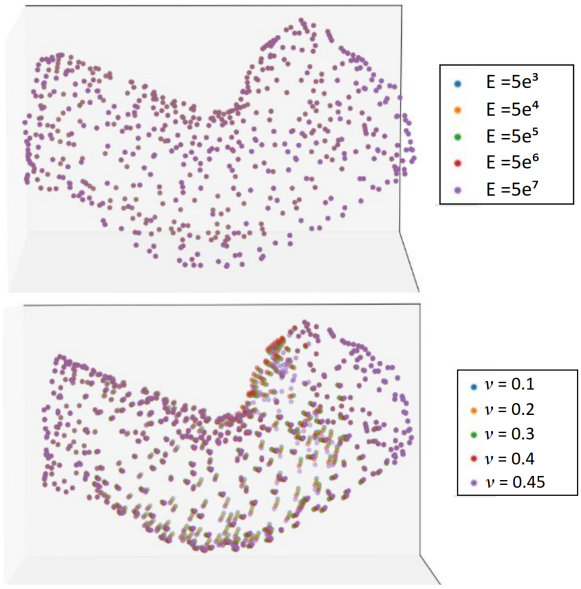


Figure 4. Application of the same displacement constraint for different material parameters (Young's modulus E and Poisson's ratio ν.

which can be mapped to an angle-axis vector $\delta\boldsymbol{\rho}_i$ to obtain $\delta\mathbf{r}_i$. The D robot pose variations are then stacked into the matrix $\boldsymbol{\Delta}\mathbf{R}$:

$$\boldsymbol{\Delta}\mathbf{R} = \begin{bmatrix}\delta\mathbf{r}_1 & \dots & \delta\mathbf{r}_D\end{bmatrix} \in \mathbb{R}^{k\times D}. \tag{8}$$

Target Selection

In contrast with most works on Deformable Object Manipulation, the target shape does not have to be reached beforehand by moving the robot or the object. Instead, we profit from SOFA: the user can select the target shape by displacing the rigid particles $\mathbf{pr}_l$ and $\mathbf{pr}_l$ via keyboard commands. This interface is fast, user-friendly, and does not require any knowledge on robot control.

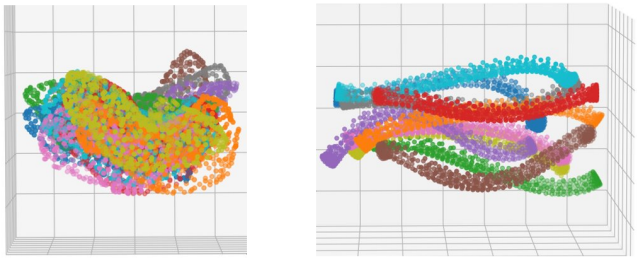


Figure 5. Examples of shapes (mesh nodes) obtained via $D = 10$ simulations with random rigid particle poses, for a sponge (left) and for a DLO (right).

Monitoring the Object's Stress

If the material parameters are known or well estimated, SOFA can compute the Von Mises stress during simulations.

The Von Mises criterion states that the body will yield if the applied stress is greater than a critical value σ_y, called the yield strength (Armenàkas 2016). The criterion is defined as:

$$\sigma_v \geq \sigma_y \tag{9}$$

This criterion is used to predict yielding for ductile materials. The Von Mises stress is represented visually in the SOFA graphic interface through a color scale (see Fig. 6, where red indicates higher internal stress). With this tool, the user can easily monitor if a target shape is hazardous or unreachable. Besides, it is possible to impose a stress limit on the mesh, so that:

- During the initial data collection, meshes with high internal stress are discarded;
- During manipulation, the controller automatically stops if the internal stress reaches high values, to avoid damaging the object.

Integrating Visual Feedback

Image Processing

We use the color of the robot's end-effectors to segment the object in the RGB image. Since the end-effectors' color remains unchanged regardless of the object being manipulated, the image acquisition needs no tuning from one manipulation to the next. Our setup uses the green color of the end-effectors, but it is straightforward to replace color segmentation by markers on the end-effectors, if needed.

The process (shown in Fig.7) starts by extracting the pixels of both end-effectors, from their known color (see Fig.7(b)). We remove end-effectors pixels, pixels exterior to the end effectors, and pixels whose depth is greater than a threshold (here, 0.5 m more than the nearest end-effector pixel).

We then run Hue Clustering (MacQueen 1967) on the first image, to extract the dominant color of the remaining pixels of the RGB image. Another Hue segmentation is applied to

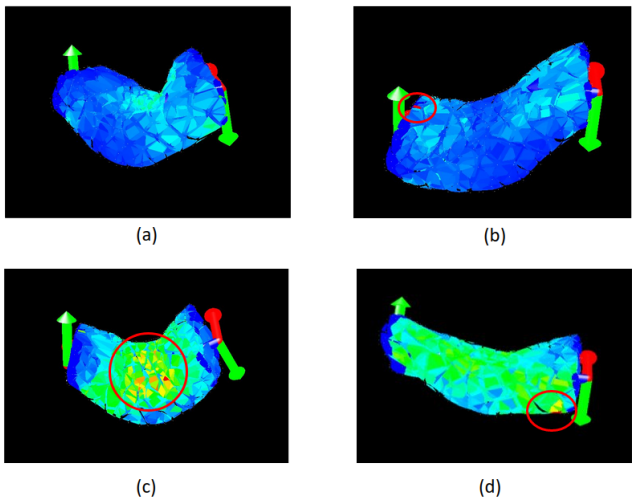


Figure 6. Visual representations of the Von Mises stress on a mesh in SOFA during target selection. On subfigures (b), (c) and (d), we circled in red the mesh areas presenting high Von Mises stress.

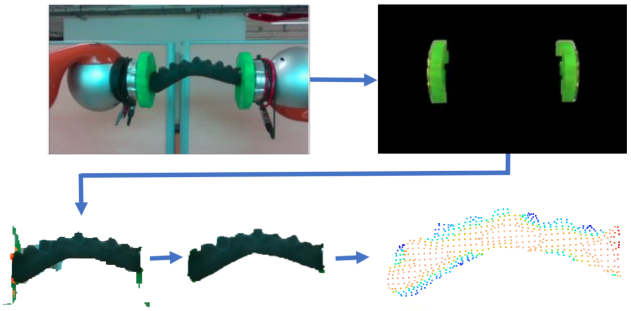


Figure 7. Steps of point cloud acquisition. From RGB image, the robot end-effectors are segmented by color. We take out the pixels belonging to the end-effectors, the pixels exterior to them, and the pixels belonging to the background, and obtain the third picture. The dominant color is then segmented, to get the fourth picture. We project it in 3D, resulting in the point cloud expressed in the camera frame (last picture).

each new RGB image to extract the pixels with dominant color. Finally, these pixels are projected into 3D coordinates using the camera intrinsic parameters and depth value. The object point cloud is obtained and transformed to the robot frame using the transformation matrix $\mathcal{T}_{cam}^{rob}$ to yield $\mathbf{pc}$.

Vision-Based Correction of the Simulated Mesh

We rely on visual feedback to correct the simulation, following the scheme shown in Fig. 8. The resulting meshes are denoted $\mathbf{m}_{i+1}$, $\mathbf{m}_{i+1}^{v}$. Since the camera pose in the robot frame $\mathcal{F}_{rob}$ is known, the transformation from the camera frame to the robot frame $\mathcal{T}_{cam}^{rob}$ is obtained. Then, the new point cloud of the object $\mathbf{pc}_{i+1}$ is transformed into the robot frame through the $\mathcal{T}_{cam}^{rob}$.

We run the open3D implementation[†] of Iterative Closest Point (ICP) between $\mathbf{pc}_{i+1,rob}$ and the visual mesh $\mathbf{m}_{i+1}^{v}$ to obtain the transformation $\mathcal{T}_{i+1}$, from point cloud to mesh. Given a source and a target set of 3D points, ICP returns a set of κ correspondences between the two. Naming n_q the number of points in the target set, we define the ICP fitness fit as:

$$fit = \frac{\kappa}{n_q} \in [0, 1] \tag{10}$$

We consider the alignment satisfactory, if the ICP outputs a fitness value $fit > 0.8$, i.e. if the point cloud and the nodes overlap by more than 80%. Otherwise, we conduct a correction step to improve the similarity between meshes and visual feedback.

This correction step is outlined in Fig. 9, and detailed hereby. Using KNN (K-Nearest Neighbors, (Maneewongvatana and Mount 1999)) algorithm, we pair the nearest visual mesh node to the corresponding point of the cloud. The result is a list of nodes $\mathbf{KNN}$, such that node $\mathbf{KNN}_i \in \mathbf{m}^v$ is the visual mesh node closest to point $\mathbf{pc}_i$. We seek the farthest pair $(\mathbf{pc}_\kappa, \mathbf{KNN}_\kappa)$, with:

$$\kappa \in \max_{\kappa \in [0,p]} ||\mathbf{pc}_\kappa - \mathbf{KNN}_\kappa||_2, \tag{11}$$

and rename the points: $\mathbf{p}* = \mathbf{pc}_\kappa$, $\mathbf{n}* = \mathbf{KNN}_\kappa$. Once $\mathbf{n}*$ is defined, we find the holding nodes $\mathbf{H}_n$: The tetrahedral mesh $\mathbf{m}$ is "sliced" along the plane normal to the end-effectors axis $\mathbf{p}_{pr}$, with a tolerance (0.01 m) on the distance to the plane, to include more nodes (see Fig.9(b)). The nodes in $\mathbf{H}_n$ are:

$$\left|\mathbf{p}_{pr} \cdot \mathbf{n} - \mathbf{p}_{pr} \cdot \mathbf{n}*\right| \leq 0.01, \mathbf{n} \in \mathbf{m}. \tag{12}$$

Then, the holding nodes $\mathbf{H}_n$ are rigidly attached to a particle $\mathbf{pr}_n$ created at the position $\mathbf{n}*$, see Fig.9(c). Since the correction is applied with translation alone, the orientation of the rigid particle is set to $\mathbf{q}_n = [0, 0, 0, 1]$. Finally, we apply a displacement constraint on the rigid particle $\mathbf{pr}_n$ to have it reach position $\mathbf{p}*$. Because of the rigid link, this displacement is also applied to the nodes in $\mathbf{H}_n$. The simulation runs and deforms the rest of the geometry accordingly, resulting in the corrected meshes $\tilde{\mathbf{m}}_{i+1}$ and $\tilde{\mathbf{m}}_{i+1}^{v}$ (Fig. 9(d)).

To avoid disrupting nodes that are already rigidly attached to an end-effector rigid particle, we ensure that there is no intersection between the corrective holding nodes $\mathbf{H}_n$ and the end-effectors holding nodes, $\mathbf{H}_l$ and $\mathbf{H}_r$. In cases where there is an intersection, the involved end-effector rigid particle is the one used to apply the correction (see Fig. 19). The matrix containing the past mesh nodes $\mathbf{M}$ is updated using the corrected mesh points $\tilde{\mathbf{m}}_{i+1}$ for the next iteration, see Fig.2.

Simulations for DLO Mesh Reconstruction

We now present a pipeline to reconstruct DLO meshes by using simulations in a way that is similar to our correction strategy. Many works consider the matter of DLO state estimation, usually representing DLOs as 1D objects. In Caporali et al. (2022), deep learning is used to extract features of a DLO through spline modeling. Wang and Yamakawa (2023) proposed a method to extract the DLO skeleton from depth images. Ma and Xiao (2023) conducts rope diameter estimation by the mean of different image processing steps. We instead propose to consider the DLO

[†] http://www.open3d.org/docs/release/tutorial/pipelines/icp_registration.html, *Point-to-plane ICP*

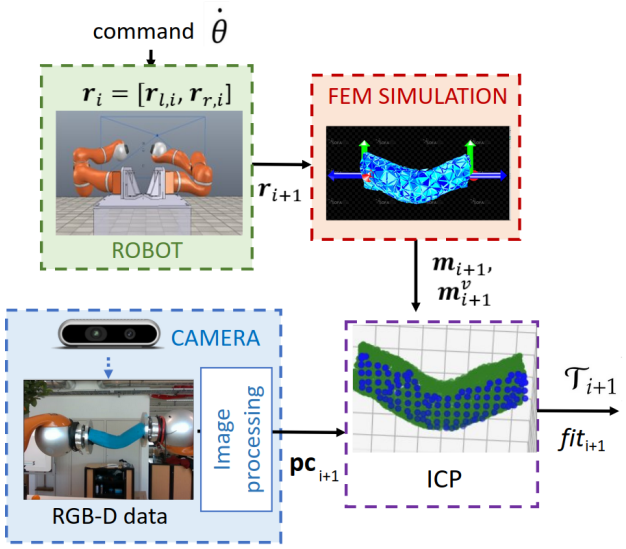


Figure 8. Process for obtaining the new shape by merging simulation and visual feedback.

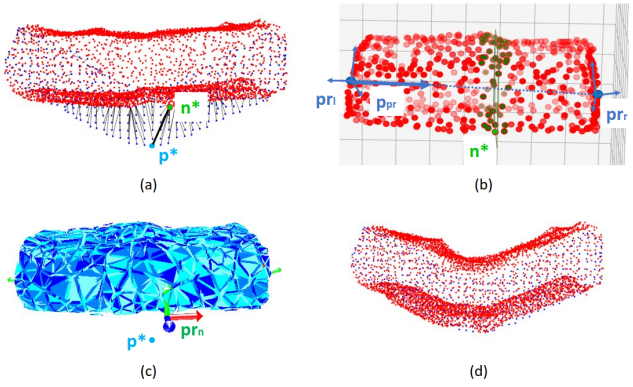


Figure 9. Steps of the visual feedback correction. (a) The mesh nodes are represented in red, the point cloud in blue. We find the farthest pair of visual mesh node $\mathbf{n}*$ and point cloud point $\mathbf{p}*$. (b) In the tetrahedral mesh, we select the holding nodes ($\mathbf{H}_n$, darker green), which are near the plane orthogonal to the end-effectors axis $\mathbf{p}_{pr}$. (c) A rigid particle $\mathbf{pr}_n$ is rigidly attached to all the points in $\mathbf{H}_n$, and displaced until $\mathbf{p}*$. (d) The rest of the mesh is deformed, resulting in $\tilde{\mathbf{m}}^v_{i+1}$.

as a cylinder (3D), and present a few image processing steps in order to extract the desired parameters (radius and length) of the considered DLO from camera feedback. The DLO is already grasped between the end-effectors, and entirely in the frame of the RGB-D camera. First, the pixels representing the DLO are segmented from the RGB image, Fig. 10(a). The largest contour $\mathbf{c}^{dlo}$ is extracted (Fig. 10(b)), and considered as the main DLO body.

Length estimation: A B-Spline interpolation is conducted on this contour (Fig. 10(c)), to obtain a set of 2D points constituting the DLO curve. These points are related to their closest pixel on the RGB image, giving the set of pixels of coordinates (i^{dlo}, j^{dlo}), and then projected into the 3D space using the corresponding depth $d(i^{dlo}, j^{dlo})$ and the camera intrinsic parameters. We obtain the set of 3D points $\mathbf{pc}^{dlo} \in \mathbb{R}^d \times 3$ (see Fig. 10(d)). The length of the DLO is then computed as:

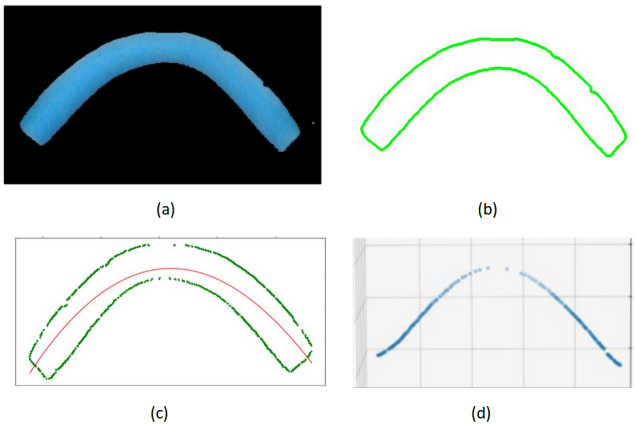


Figure 10. Extraction of the 3D points constituting the DLO for length estimation. The body of the DLO is segmented (a), then the largest contour is found (b). B-spline approximation gives the 2D curve (c) and is projected in 3D to give $\mathbf{pc}^{dlo}$ (d).

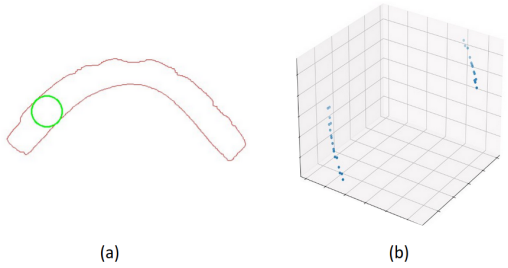


Figure 11. Image processing for radius estimation. The largest circle (green) fitting in the contour (red) is found (a), and the mask of their intersection is created. These pixels are projected in 3D (b) and used to estimate the radius of the DLO.

$$l = \sum_{k=0}^{n-1} \|\mathbf{pc}^{dlo}_{k+1} - \mathbf{pc}^{dlo}_{k}\|. \tag{13}$$

Radius estimation: We find the largest circle fitting in the contour $\mathbf{c}^{dlo}$, see Fig.11(a). The pixels contained in the intersection between this circle and $\mathbf{c}^{dlo}$ are projected in 3D (Fig. 11(b)), also using the corresponding depth $d(i^{dlo}, j^{dlo})$ and the camera intrinsic parameters and giving the set of 3D points $\mathbf{pc}^{rad}$. The Euclidean distance between each point in $\mathbf{pc}^{rad}$ is computed as $\mathbf{D}$, and the radius is defined as:

$$R = max(\mathbf{D})/2. \tag{14}$$

We obtain the parameters of the DLO, (R,l). Table 1 presents the result of the DLO parameters estimation from visual feedback for different objects. These results give a mean error of 7% for the radius and 2% for the length. The error for the radius is between 0% and 15%. The error for the length is between 0.8% and 4.5%. We use the python library *gmsh*[‡] to generate a cylinder from parameters (R, l) and create the triangle and tetrahedral meshes, as shown in Fig. 12.

The obtained meshes are cylinders, whereas the DLO may be deformed and therefore have a non-cylindrical configuration in its observed state, as seen in Fig.12. Hence,

[‡] https://gmsh.info/

Object	Real radius	Estimated radius	Real length	Estimated length
Rope	3	3	525	530
Foam noodle 1	30	29	630	625
Foam noodle 2	30	27	510	487
HIDRIA part	20	17	355	350

Table 1. Result of the DLO parameters estimation on different objects, values in mm. The real values were measured by hand. All objects are shown in Fig.14.

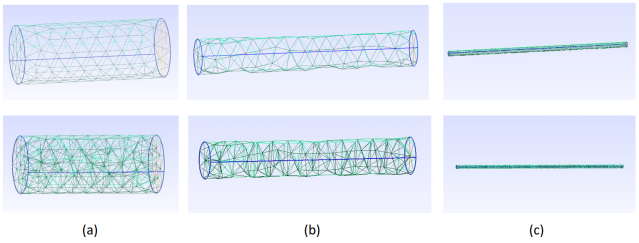


Figure 12. DLO meshes generated for different pairs of parameters (R, l): (a)$(R, l) = (0.1, 0.5)$, (b)$(R, l) = (0.05, 0.7)$, (c)$(R, l) = (0.005, 0.5)$. Top row shows the visual meshes $\mathcal{M}_0^v$ and bottom row the tetrahedral meshes $\mathcal{M}_0$.

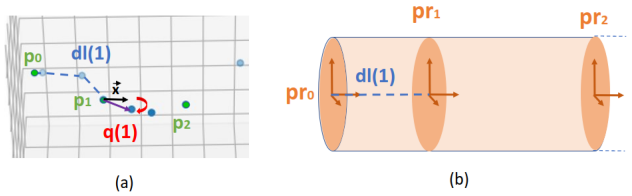


Figure 13. Sampling for alignment between the mesh in resting position and the current configuration of the DLO. (a) shows $\mathbf{pc}^{dlo}$ sampled into k_s points (in green). Each sample $\mathbf{p}_i$ is associated to the length separating it from the previous sample, $dl(i)$, and to a rotation from the x-axis, $\mathbf{q}(i)$. (b) shows the corresponding sampling of the mesh $\mathbf{m}$ into rigid particles $\mathbf{pr}_i$.

we need to conduct an extra step to align the meshes with the current observed DLO.

To this end, we subsample the 3D points constituting the DLO curve $\mathbf{pc}^{dlo}$ (see Fig.10 (d)) down to k_s samples. The length between sample $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$ is denoted $dl(i)$, $i \in [0, k_s]$ and it is calculated using all the points in $\mathbf{pc}^{dlo}$ in between. We take $dl(0) = 0$ for the first point in $\mathbf{pc}^{dlo}$. The vector between sample $\mathbf{p}_i$ and the next point in $\mathbf{pc}^{dlo}$ is also computed and expressed as a rotation from the x-axis unit vector, and denoted $\mathbf{q}(i)$, see Fig.13 (a). The last sample $\mathbf{p}_{k_s}$ is the very last point in $\mathbf{pc}^{dlo}$, and its rotation $\mathbf{q}(k_s)$ is calculated with the closest point in $\mathbf{pc}^{dlo}$.

A FEM simulation scene is created with the cylindrical meshes. The tetrahedral mesh of the DLO, $\mathcal{M}_0$, is also sampled: we create k_s rigid particles. The rigid particle $\mathbf{pr}_k$ is created at length $dl(k)$ from the axis of the cylinder, see Fig.13 (b). Each rigid particle is then rigidly attached to the mesh nodes around it, similarly to the correction strategy presented in Sec. Vision-Based Correction of the Simulated Mesh. Finally, translation to the corresponding sample $\mathbf{p}_k$ and rotation $\mathbf{q}(k)$ is applied to each rigid particle $\mathbf{pr}_k$ to

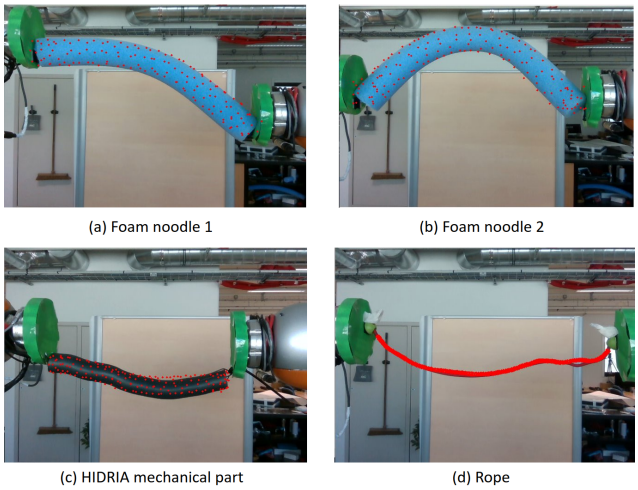


Figure 14. Result of the DLO mesh alignment for different objects. The triangle mesh nodes after alignment are projected on the corresponding RGB image, in red.

align it with the current configuration of the DLO. The FEM simulation is run, deforming the mesh parts which are not rigidly mapped to any rigid particle accordingly, until quasi-equilibrium is reached. The resulting meshes $\mathcal{M}$ and $\mathcal{M}^v$ are obtained. Examples of the resulting alignment of DLO meshes are compared to the camera view in Fig.14.

The number of samples k_s is tuned depending on the length and shape of the DLO: k_s small for short DLO or smooth curvatures as the foam noodles, and k_s bigger for thin and long objects like the rope. It should also be noted that in case of thin objects, depth camera precision may cause the depth map of the DLO to be incomplete - and possibly miss points on the extracted point cloud. This can cause some parts of the DLO not to be aligned, as observed on Fig.14(d).

Controller Design

In this Section, we detail each of the modules which compose our controller (refer to Fig. 2).

Principal Component Analysis

We start by decreasing the size of past mesh matrix $\mathbf{M}$, via PCA. First, we shift each column of $\mathbf{M}$ by the mean of all columns, $\bar{\mathbf{m}} = \frac{\sum \check{\mathbf{m}}_j}{D+1}, j = 0, ..., D$, to obtain:

$$\mathbf{M}_m = [\check{\mathbf{m}}_0 - \bar{\mathbf{m}} \cdots \check{\mathbf{m}}_D - \bar{\mathbf{m}}] \in \mathbb{R}^{3n(D+1)}. \quad (15)$$

We then compute $\mathbf{Q}$, the covariance matrix of $\mathbf{M}_m$ and via Singular Value Decomposition on $\mathbf{Q}$, the eigenvector matrix $\mathbf{U} \in \mathbb{R}^{3n \times 3n}$. We select the first k columns of $\mathbf{U}$ to control k DOF of our robot. These columns form the projection matrix $\mathbf{U}_k \in \mathbb{R}^{3n \times k}$, whose columns correspond to the k principal components (with highest variances) in the dataset, and therefore determine the directions of highest variability in the data. Then, at each iteration i, the reduced feature vector is (Zhu et al. 2020):

$$\mathbf{s}_i = \mathbf{U}_k^\top (\check{\mathbf{m}}_i - \bar{\mathbf{m}}) \in \mathbb{R}^k. \quad (16)$$

Estimation of the Inverse Interaction Matrix

The next step consists in estimating the inverse interaction matrix $\mathbf{L}^{-1}$ needed for control (see (2)). This matrix is unknown for a non-rigid object, and it should be inverted to control the robot pose. To this end, we first project the shifted mesh nodes matrix into the reduced space:

$$\mathbf{S} = \mathbf{U}_k^\top \mathbf{M}_m \in \mathbb{R}^{k \times D+1}, \tag{17}$$

then calculate the variation over a D-dimensional window:

$$\mathbf{\Delta S} = \begin{bmatrix} \mathbf{S}_1 - \mathbf{S}_0 & \dots & \mathbf{S}_D - \mathbf{S}_{D-1} \end{bmatrix} \in \mathbb{R}^{k \times D}. \tag{18}$$

Finally, the inverse interaction matrix is given by:

$$\mathbf{L}^{-1} = \mathbf{\Delta R \Delta S}^\dagger \in \mathbb{R}^{k \times k}, \tag{19}$$

with $\mathbf{\Delta S}^\dagger$ the pseudo-inverse of $\mathbf{\Delta S}$. The matrix $\mathbf{L}^{-1}$ is re-computed at each control iteration with the updated matrices $\Delta\mathbf{S}$ and $\Delta\mathbf{R}$ to ensure it is locally representative.

Joint Controller

The linear approximation (2) is only valid locally. Therefore, a control law based on $\mathbf{L}^{-1}$ cannot guarantee convergence if the initial and final shapes are too different. To solve this, at each control iteration i we generate an intermediary target mesh via linear interpolation:

$$\check{\mathbf{m}}_i^* = \frac{\check{\mathbf{m}}^* - \check{\mathbf{m}}_i}{n} \tag{20}$$

with n big enough to ensure small displacements. Correspondingly, we project $\check{\mathbf{m}}_i^*$ into the feature vector:

$$\mathbf{s}_i^* = \mathbf{U}_k^\top (\check{\mathbf{m}}_i^* - \bar{\mathbf{m}}) \in \mathbb{R}^k. \tag{21}$$

and compute the desired robot pose variations via:

$$\delta\mathbf{r}_i = \mathbf{\Lambda L}_i^{-1}(\mathbf{s}_i^* - \mathbf{s}_i) \in \mathbb{R}^k, \tag{22}$$

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. The robot pose variation command $\delta\mathbf{r} \in \mathbb{R}^{12}$ is mapped to the joint velocities $\dot{\boldsymbol{\theta}}$ through the standard QP (Quadratic Programming) optimization problem:

$$\begin{aligned} &\min_{\dot{\boldsymbol{\theta}}} ||\mathbf{J}\dot{\boldsymbol{\theta}} - \delta\mathbf{r}||_2 \\ &\text{subject to: joint limits.} \end{aligned} \tag{23}$$

We take into account joint limits on position, velocity and acceleration, and use

$$\mathbf{J} = \begin{bmatrix} \mathbf{J_l}^\top & \mathbf{J_r}^\top \end{bmatrix}^\top \tag{24}$$

with $\mathbf{J_l}$ and $\mathbf{J_r}$ the Jacobians of the left and right arm.

Once the command has been achieved, the new robot pose $\mathbf{r}_{i+1}$ is retrieved, and transferred to the simulator: the rigid particles $\mathbf{pr}_l$ and $\mathbf{pr}_r$ representing the end-effectors are constrained to reach the same pose. This constraint is, by proxy, imposed to the corresponding holding nodes $\mathbf{H}_{l,r}$, which are rigidly attached to the particles. The rigidly attached nodes are instantly displaced, but the rest of the shape needs to be deformed according to this new disturbance; this is where the FEM simulation runs to deform the rest of the mesh until quasi-equilibrium is reached, i.e. until we can consider that $\mathbf{m}$ stops varying following:

$$\|\mathbf{m}_{i+1} - \mathbf{m}_i\| \leq (1e^{-08} + 0.02\|\mathbf{m}_i\|) \tag{25}$$

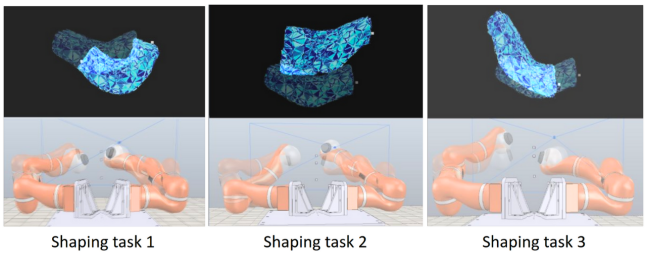


Figure 15. Example of 3 different simulated experiments. The two rows show the initial and final configurations of mesh (top) and robot (bottom) in simulation.

Experiments with the Robot Simulator

In our framework, the use of mesh nodes for shape feedback allows us to test out the control framework entirely in simulation, by sending the joint command to the simulator instead of sending it directly to the robot. This allows feeding the result of the FEM simulations directly into the data matrix $\mathbf{\Delta M}$, by skipping the vision related steps (point cloud extraction and correction of the mesh).

Fig. 15 presents the results of some shaping simulations. These results shows the dual-arm robot reaching successfully three different target shapes involving different DOFs in $\mathbb{SE}(3)$, and task with translating, bending, twisting, and stretching. Simulations are also very useful for investigating more in depth the dimension of the data matrices required in the framework, which would be cumbersome to do with real robot experiments. We discuss this in the following.

Choice of the Number of Samples

Let us look at the number of samples D which constitute data matrices $\mathbf{\Delta M}$ and $\mathbf{\Delta R}$, which respectively contain past shapes, and corresponding end-effector poses.

Since these data can be collected through simulations, we run three, with the goal of reaching the target shapes shown in Fig.15, referred to as *target shape 1*, *target shape 2* and *target shape 3* for columns 1 to 3 respectively. We attempt to reach each target shape with a variable number of samples, to explore the impact of D on the controller's success. The experiments are repeated thrice, with different initialization data for each trial. We consider as failed an experiment which has not reached the acceptable error threshold after 200 iterations of the control loop. The results are summarized in Table 2.

The simulated experiments show that a number of samples too small (13) or too high (40) leads to some experiments not converging. Indeed, the number of samples too small will lead to little variety, thus a bad representation of the available motions; on the contrary, too many samples imply losing local information about motion.

In terms of performances, by comparing the cases of $D = 15$ and $D = 25$, we notice that the lesser the samples, the fastest the computations. Keeping all of this in mind, it appears that choosing a number of samples between 15 and 25 would be a good trade-off between performance, locality while reducing the risk of under-representing the available motions. To ensure not an under-determined (similarly to experiments with $D = 13$), we choose $D = 17$ for real robot experiments.

D	Target shape	Average total time (s)	Successes
13	1	50	**2**/3
13	2	64	3/3
13	3	105	**2**/3
15	1	48	3/3
15	2	64	3/3
15	3	104	3/3
25	1	81	3/3
25	2	71	3/3
25	3	119	3/3
40	1	114	3/3
40	2	89	3/3
40	3	-	**0**/3

Table 2. Simulated experiments with different number of samples. $D = 15$ and $D = 25$ both give a 100% success rate on the three trials.

Choice of the Number of Features

It is not required for the interaction matrix $\mathbf{L}$ to be square. It is then worth questioning how many features are needed to represent the shape of an object. Therefore, in this section, we investigate the significance of the number of visual features and its impact on the command computations. We recall that the visual features are obtained after PCA on the past data matrix, see equation 16.

Since it is custom in visual servoing to consider a number of features at least as great as the number of DOF controlled Chaumette (2007) (to avoid redundant visual data), here, $[\mathbf{r}_l, \mathbf{r}_r]$, our first choice is to consider $k = 12$ features.

To verify the validity of this choice, we conduct a series of simulations. As metric, we use the explained variance, which indicates how much each of the principal components encodes the dataset variation. The variance of the i-th principal component is defined, for a column vector of size $3p$, as:

$$var(\lambda_i) = \frac{\lambda_i}{\lambda_1 + \lambda_2 + \ldots + \lambda_{3p}} \tag{26}$$

with λ_i the eigenvalue associated with the i-th principal component resulting from the SVD (refer to Sec. Principal Component Analysis). This metric allows to compare their significance, thus their encoding of useful information about the shape. We then observe the explained variance of the principal components, resulting from the PCA during 4 different trials, and summarize it in Table 3. For each trial, the PCA is performed on a new set of $D = 17$ randomly generated samples. For principal component i, Cumulative proportion of explained variance is given by:

$$CP(i) = \sum_{n=1}^{i} var(\lambda_n) \tag{27}$$

As Table 3 shows, more than 90 % of the shape is encoded by 6 to 8 principal components, but 10 to 12 principal components represent a variance close to 1, giving the best representation of the data. These simulations comfort us in the choice of taking $k = 12$ features for our control strategy.

In this section, we showed that the control framework could, with the exception of the modules involving visual feedback, be validated through simulations only, allowing us to tune most parameters offline, and to avoid cumbersome tests on the real robot.

Number of principal components (k)	6	8	10	12
Trial 1	0.89	0.97	0.99	0.99
Trial 2	0.89	0.96	0.98	0.99
Trial 3	0.9	0.98	0.98	0.99
Trial 4	0.9	0.97	0.97	0.99
Average	**0.9**	**0.97**	**0.98**	**0.99**

Table 3. Cumulative proportion of explained variance for different number of principal components, for different trials.

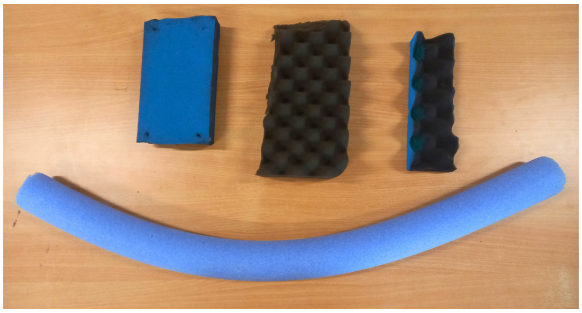


Figure 16. Different foam objects used for the experiments. Left to right: sponge, convoluted foam, thin convoluted foam. Bottom: foam noodle.

Experimental Results

We conducted experiments with different foam objects, of varying shapes, densities and rigidities (see Fig.16). The source code for robot manipulation is available on our gite. A video of the experiments can be found at here.

Experiments with Unknown Material Parameters

In a first series of experiments, we ignore the Young's modulus, Poisson's ratio, and material density and set them to default values in SOFA: $E = 2.5e^6$ Pa, $\nu = 0.3$ and $\rho = 10\ kg.m^{-3}$ respectively. The experiments consist in shaping the object held by the robot arms into a target shape. The deformation of the object is observed through a RGB-D camera, Intel Realsense D435.

At each control iteration i, the displacement of the end-effectors of the robot from the previous iteration is applied to the FEM simulations, to obtain deformed mesh $\mathbf{m}_i$. Meanwhile, the point cloud of the deformed object $\mathbf{pc}_i$ is extracted, and ICP returns $\mathcal{T}_i$, the transformation from $\mathbf{pc}_i$ to best fit $\mathbf{m}_i^v$. If the ICP fitness is less than 0.8, we perform the correction step explained in Sec. Vision-Based Correction of the Simulated Mesh. The resulting mesh is $\tilde{\mathbf{m}}_i$.

We consider the shape error at iteration i as the error norm between the *transformed* corrected mesh nodes and the target mesh nodes, in meters:

$$e_i = \|\mathbf{m}^* - \mathcal{T}_i^{-1}\tilde{\mathbf{m}}_i\|, \tag{28}$$

and stop the manipulation when the error is below a threshold set to $e = 0.05$ m.

Fig. 17 and 18 show the experiments, which are also summarized in Tab. 4. In the figures, the blue points represent

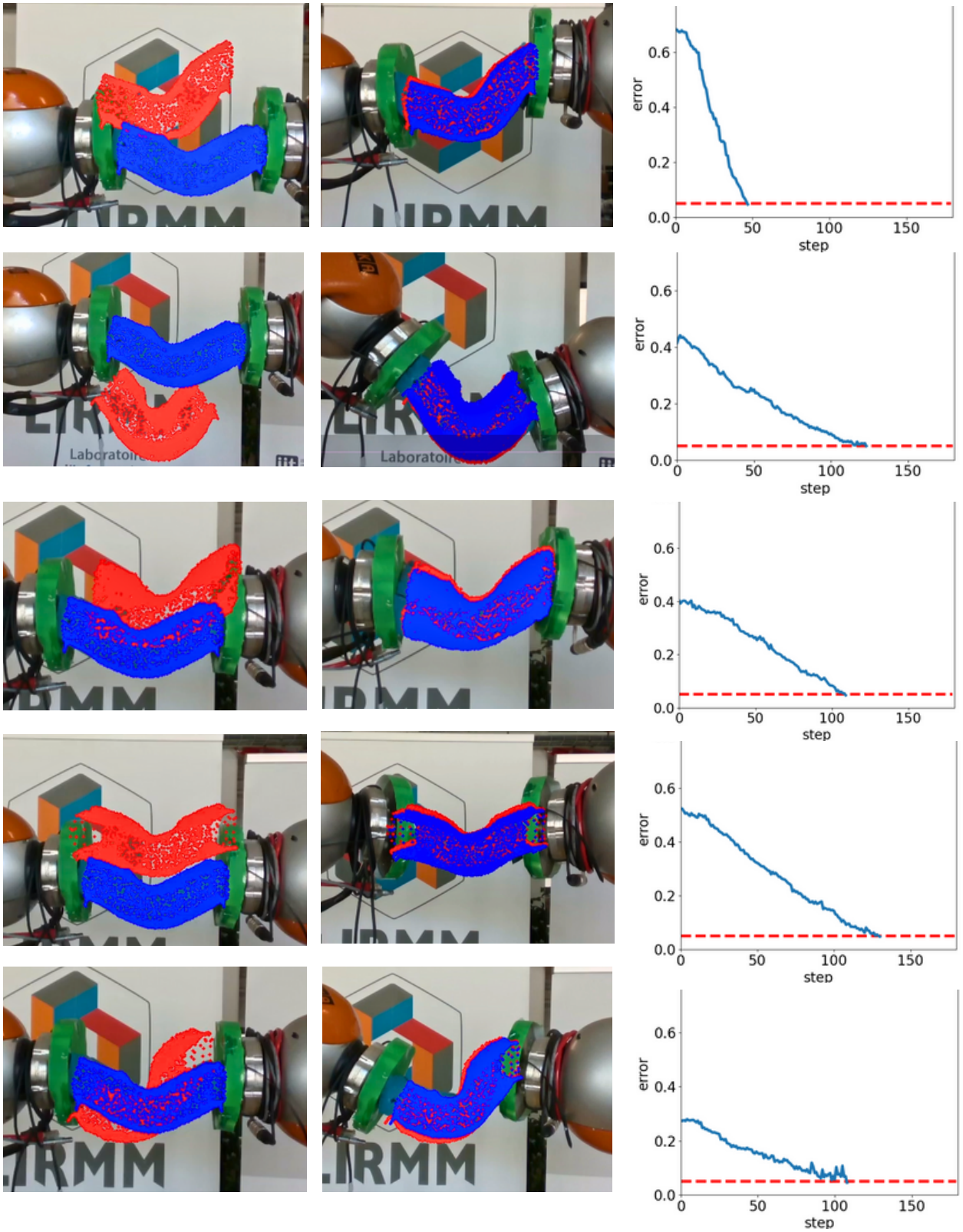


Figure 17. Experiments with the sponge. First column: initial shape of the mesh (blue) compared to target shape (red), both projected on the RGB image. Second column: obtained final shape. Third column: evolution of error e during the experiments, until threshold $e < 0.05m$ (red dotted line) is reached.

the corrected mesh nodes projected on the RGB image. The red points represent the target mesh nodes, also projected on the image.

Fig. 17 presents different experiments with the sponge. They include bending around different axes and torsion (third row). All succeed in reaching the target shape. The fourth row experiment also presents convergence to the target shape, despite the robot end-effectors visually occlude parts of the object. Fig. 18 shows different experiments with other foam objects. On the error plot of the second experiment with

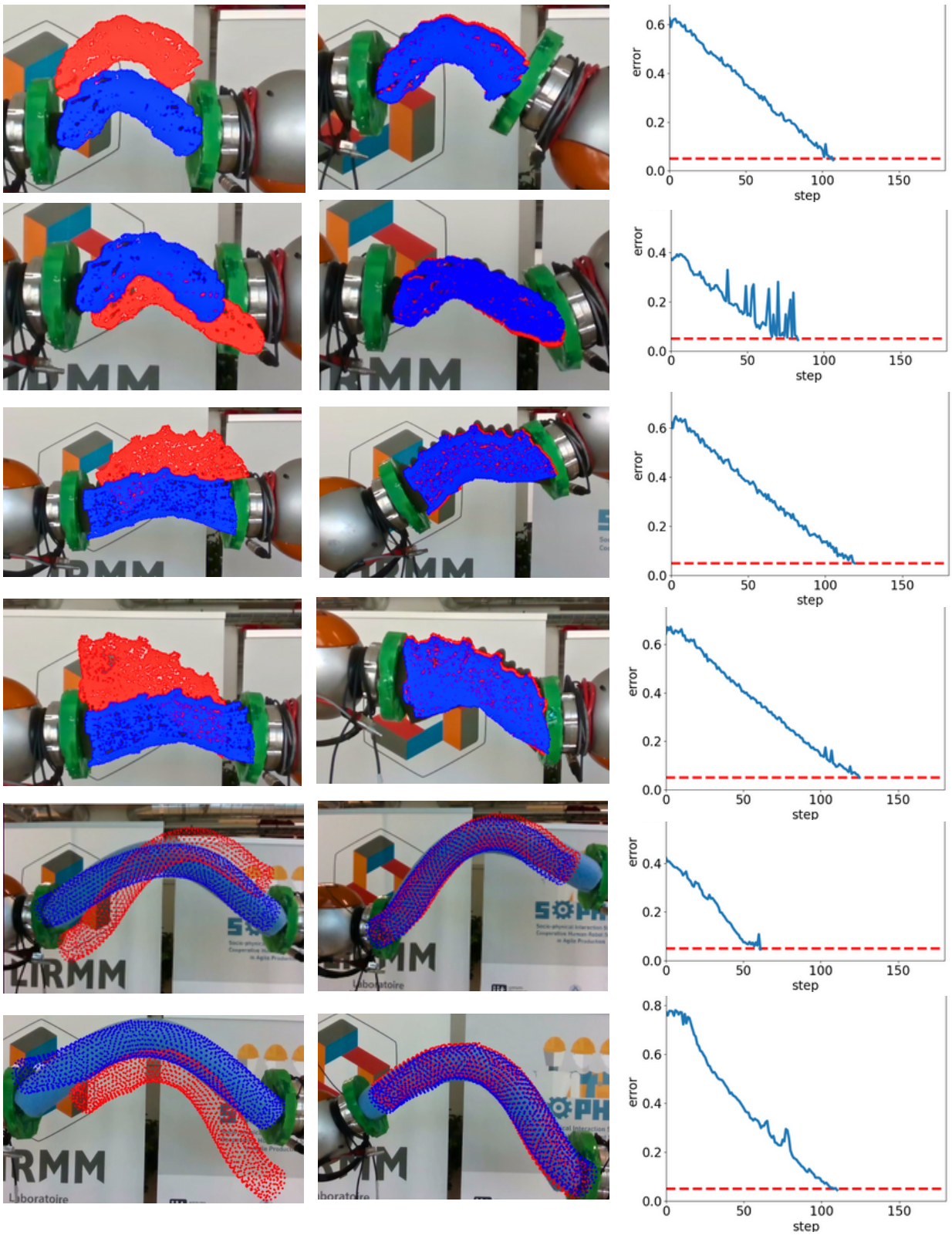


Figure 18. Experiments with two convoluted foams and a foam noodle. First column: initial shape (blue) and target shape (red). Second column: final shape compared to the target. Third column: evolution of error e until threshold $e < 0.05$ is reached.

the thin convoluted foam, presented on the second row of the figure, the task error e oscillates due to the ICP converging alternatively to different transformations. Since the data matrix $\mathbf{M}$ is not directly impacted by the result of ICP, the control algorithm manages nonetheless to decrease the task error, until convergence. Additionally, one can observe that on the last experiment with the foam noodle (fifth row), the initial mesh is not exactly aligned with the point cloud of the object. Yet, the shift between the two is corrected during manipulation.

Fig. 20 presents the results of experiments with the foam noodle reaching the same target shape with different

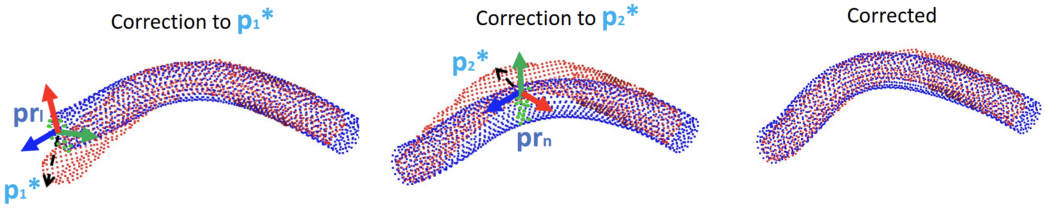
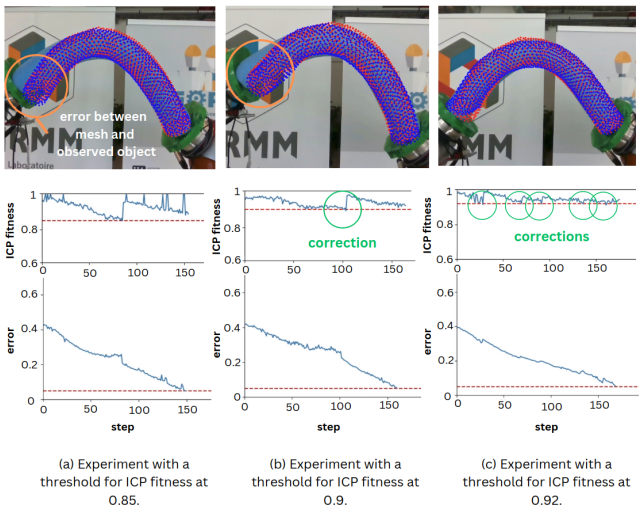


Figure 19. Two correction steps of the mesh nodes (blue) with the point cloud of the object (red), after ICP alignment (left) during an experiment with a noodle foam. The first correction involves holding nodes (green) that intersect with the left end-effector holding nodes; the rigid particle representing the left end-effector $\mathbf{pr}_l$ is then the one used to apply the correction to reach $\mathbf{p}_1*$. The second correction involves the corrective rigid particle $\mathbf{pr}_n$ that is translated to $\mathbf{p}_2$. The final result after both actions is shown on the right.

Object	Experiment	Tasks
Sponge	1	Bending
	2	Compression with rotation
	3	Torsion
	4	Bending (around y-axis)
	5	Bending with rotation
Thin convoluted foam	1	Bending with rotation
	2	Stretching
Convoluted foam	1	Bending, spinning
	2	Torsion
Foam noodle (DLO)	1	Out of plane deformation
	2	Out of plane deformation, with rotation

Table 4. Summary of the experiments performed.



(a) Experiment with a threshold for ICP fitness at 0.85.

(b) Experiment with a threshold for ICP fitness at 0.9.

(c) Experiment with a threshold for ICP fitness at 0.92.

Figure 20. Experiments showing how tuning the ICP fitness threshold may ensure adequate visual feedback correction. The first row shows the final shape in blue, compared to the target shape in red, projected on the RGB image. In orange are circled the difference misalignment between the mesh and the object. The second row shows the evolution of the ICP fitness score during the experiments, with the correction steps circled in green. The third row shows the evolution of the error during the experiments, reaching the threshold $e < 0.05$.

thresholds of ICP fitness, resulting in different occurrences of the correction step. We can observe that with a higher threshold - and therefore more correction steps during the experiment - the final shape of the mesh resembles more the observed one. It is then necessary to tune the threshold of the ICP fitness accordingly to obtain adequate visual feedback correction. Fig. 19 shows an example of consecutive correction steps of the initial ICP alignment.

Stress Monitoring Experiments

In this section, we present a bending experiment with simultaneous stress monitoring. The object used in this experiment is a planar test tube 3D-printed in Polylactic acid (PLA). The Young's modulus, Poisson's ratio and density of PLA are known and set in SOFA accordingly: $E = 3120MPa$, $\nu = 0.36$ and $\rho = 1240kg/m^3$ respectively.

We study the evolution of the deformation with the value of the stress applied to it, following a typical strain-stress relationship for ductile materials. At first, the deformation is *elastic*, and the object comes back to its original shape when the applied stress returns to zero. When the applied stress exceeds a first threshold (orange dotted line in Fig.21), the deformation becomes *plastic*. From there, even when the stress diminishes again, the object will remain deformed: it is permanently damaged. Finally, when the stress reaches the plastic limit (red dotted line in Fig.21), the object will reach its rupture point and break.

In Fig. 22 we present three experiments with the same target shape. During the experiment, we plot the Von Mises stress at each node. In the first experiment, a stress limit is applied, to ensure that the deformation stays in the elastic domain. In the second one, we allow the object to be damaged (i.e. the deformation is in the plastic domain) but apply a stress limit to avoid breaking. Finally, the last experiment shows the manipulation without stress limit. For each experiment, the evolution of the Von Mises stress is plotted (first row), and we show the resulting shape of object after experiment (second row). These experiments show that with our framework, if the material parameters are known, applying a stress limit during manipulation can avoid the damaging and/or breaking of the object manipulated.

Time Performances

One important aspect highlighted by these experiments is the time performance: with the simulations running in real time, shaping an object can take between 2 to 8 minutes depending on the complexity of the task and object considered.

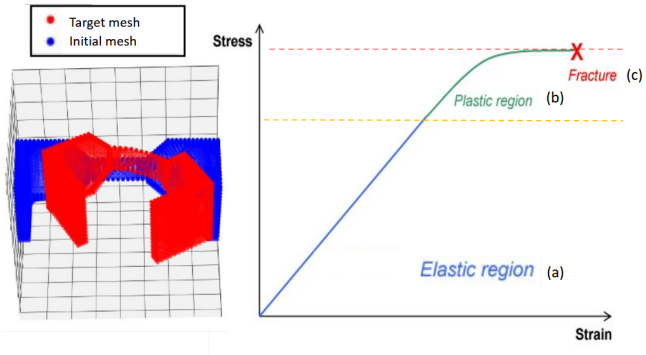


Figure 21. The goal of this experiment is to drive the initial shape (blue) to the target shape (red). We monitor the internal stress to observe the three types of deformation: elastic (blue), plastic (green) and finally, fracture (red). In the elastic region, the body recovers its original shape when the stress is released, whereas in the plastic region, the deformation is irreversible, until fracture is reached.

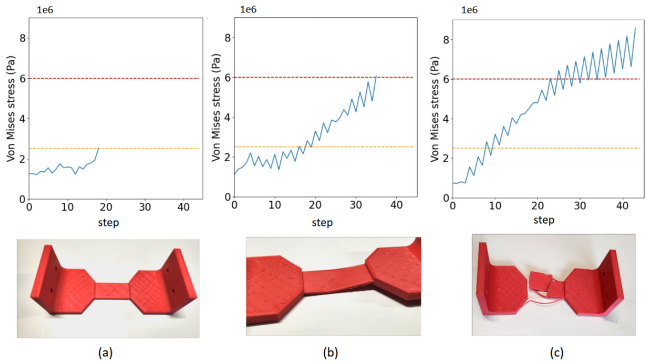


Figure 22. Different stress monitoring experiments. In the first experiment (first column), the deformation of the object stays elastic during manipulation, and the shape of object after experiment goes back to its original shape. In the second experiment (middle column), the deformation is plastic: the object is permanently damaged even after relaxation of the applied stress. The last experiment (last column) is conducted without stress limit, with the deformation going beyond plastic domain and breaking the object.

Let us analyze the time performances of our framework, on the sponge experiments in Fig. 17. The duration differs depending on the complexity of the target and of the initial error, but we can compare the average duration of one control iteration. An iteration is composed of four steps:

- *Simulation (SIM)* : application in the FEM simulation of the displacement constraint from the robot poses, until quasi-equilibrium is reached;
- *Image processing (IMPROC)*: acquisition of the object point cloud;
- *Correction (CORREC)*: execution of ICP and - if needed - correction;
- *Control (CTRL)*: update of data matrices and PCA, estimation of the inverse interaction matrix, computation of the command (Sec. Controller Design), sending the command to the robot, performing the command and acquiring the new robot poses.

Fig. 23 shows the average duration of each of these steps, for experiments (numbers corresponds to top to bottom of Fig. 17), and Table 5 summarizes the duration of each step, averaged over all five experiments. We see that the average duration of a single iteration is close to $3s$, with the biggest part of it dedicated to the *simulation* step (almost $2s$). The second most time-consuming step is *control*, which takes about one second per iteration. This step depends on the gains of the robot, since it waits for the motion of the end-effectors to be concluded, before obtaining the new poses. Point cloud acquisition takes around $0.045s$ on average. ICP algorithm runs quite quickly compared to the rest, with an average time bellow $0.02s$ when no correction step is needed. However, this time increases quickly with the correction step, since additional simulations are needed. For instance, regarding the last experiment in Fig.20, which required several correction steps, we obtain an average time on the whole experiment for the *correction* step of $0.0168s$ per iteration, versus $0.751s$ in average for the steps where a correction simulation is needed.

In summary, besides the code not being optimized, the controller is mostly slowed by the FEM simulations. One way of speeding them up, is to study the effect of the mesh

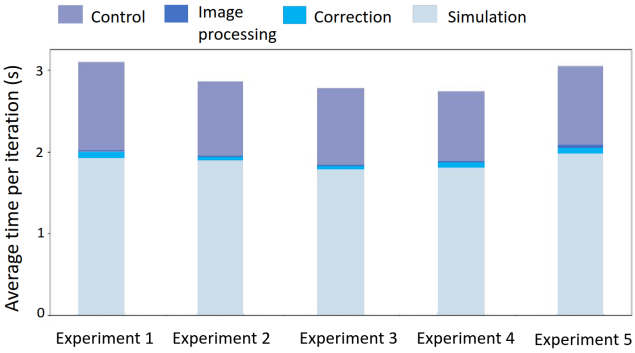


Figure 23. Average time per iteration of the steps for the five sponge experiments shown in Fig. 17.

size on the framework performance. We conduct additional experiments with meshes of different sizes (see Fig. 24). In Fig. 25, we plot the duration of the *simulation* and *correction* steps, as well as the average duration of one displacement constraint generated for the data initialization (named *initialization step*).

These experiments show that the simulation time largely depends on the size of the meshes, for the initialization and simulation steps. Both take an average time of 0.5 s for a mesh made of 100 nodes, and between 2 and 3 s for a mesh of 374 nodes. The *correction* step time is, in total, lower, as it is not performed often. In the sponge experiments, the tetrahedral mesh used for the computations contains 579 nodes and 1628 elements, which is quite significant, and which explains the slow pace of the simulations and as a result, of the iterations. Yet, it is important not to reduce too much the size of the mesh, so as to not lose shape data (e.g., note the curvature difference between the meshes of Fig. 24). It is also important to keep in mind that the least the number of elements, the more rigid the mesh. Choosing the size of the tetrahedral mesh consists in finding a satisfying trade-off between shape details, computation time and rigidity. The whole code could also be optimized to be faster, using multi threading for instance. The present work was more focused

Experiment	SIM	IMPROC	CORREC	CTRL
1	1.950	0.043	0.018	1.091
2	1.893	0.045	0.016	0.905
3	1.782	0.045	0.016	0.935
4	1.808	0.044	0.017	0.873
5	1.985	0.046	0.031	0.984
Average	**1.884**	**0.045**	**0.02**	**0.958**

Table 5. Average time per iteration of the steps for the experiments presented in Fig. 17.

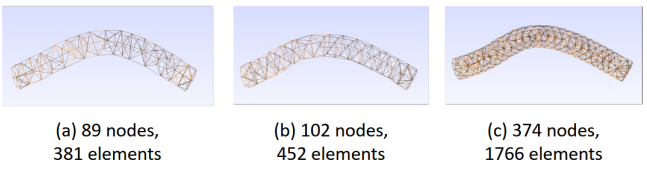
on the methodology, but this should be kept in mind for future work.

Conclusion and Discussion

In this article, we presented a framework for shaping soft objects, based on both visual-servoing and model-based methods. We use PCA to reduce the dimension of the object, and we estimate a linear local mapping between past features and corresponding robot poses. Further, we represent the object by a visual (triangular) and a tetrahedral mesh, to estimate the deformation of the object with regards to the motions of the end-effectors at every step, through FEM simulation. We validate our framework in a series of experiments by shaping various objects in 3D.

One of the limitations of our framework is that it requires meshes of the objects; yet, in industry, CAD models of manipulated objects are usually available. Otherwise, one can rely on one of the many existing methods for reconstructing meshes from scans or point clouds. In addition, we do not require exact knowledge of the material parameters for objects that are light and/or rigid, making the effect of gravity on the shape negligible compared to the effect produced by the robot position constraints. While it was the case for the objects used in our experiments (Sec. Experiments with Unknown Material Parameters), this can be limiting. Indeed, the material parameters and object weight should be known if gravity significantly affects the shape, or if the user wants to monitor stress during manipulation. Also note that the assumption of linear elasticity cannot be applicable to all objects; this could cause the FEM deformation estimation to diverge from the real observed object behavior. In the experiments we showcased, however, we demonstrate that our framework succeeds in multiple shaping tasks for various soft objects, in 3D, with no knowledge of the material. In all these cases, linear elasticity gives a good enough approximation, without needing a complex numerical model.

We also presented a thorough study of the time performances of our framework. As presented, to date, the framework run time is quite slow. Future work should involve code optimization, to allow faster running, by possibly reducing the size of the meshes while constructing them. For this purpose; we also aim at investigating the use of existing SOFA plugins, especially the *Model Order Reduction* (Goury and Duriez (2018)) for faster computations, or the *SoftRobots* plugins (Coevoet et al. (2017)) allowing to use an inverse model to solve constraints. Investigation of other types of objects (e.g., fabrics) and models (e.g., non-linear, anistotropic) could also be interesting.



(a) 89 nodes, 381 elements (b) 102 nodes, 452 elements (c) 374 nodes, 1766 elements

Figure 24. Meshes of different sizes for the foam noodle.

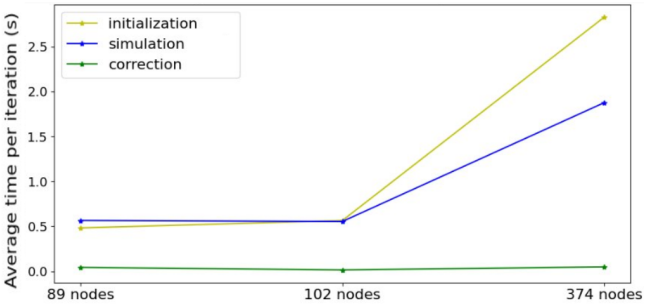


Figure 25. Average duration of an iteration for different mesh sizes.

Acknowledgements

This work was supported by ANR (PEPR PC2 PI2-IMM Organic Robotics) under grant 22-EXOD-0003, and in part by the Research Student Attachment Programme 2022/23 of The Hong Kong Polytechnic University (PolyU).

References

Aghajanzadeh O, Aranda M, López-Nicolás G, Lenain R and Mezouar Y (2022a) An offline geometric model for controlling the shape of elastic linear objects. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 2175–2181. DOI:10.1109/IROS47612.2022.9981466.

Aghajanzadeh O, Picard G, Ramon JAC, Cariou C, Lenain R and Mezouar Y (2022b) Optimal deformation control framework for elastic linear objects. In: *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. pp. 722–728. DOI:10.1109/CASE49997.2022.9926528.

Andronas D, Kampourakis E, Bakopoulou K, Gkournelos C, Angelakis P and Makris S (2021) Model-based robot control for human-robot flexible material co-manipulation. In: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. pp. 1–8. DOI:10.1109/ETFA45728.2021.9613235.

Armenàkas AE (2016) *Advanced mechanics of materials and applied elasticity*. CRC Press.

Azad A, Quentin H, Faiz BA and Véronique P (2023) Optimal cosserat-based deformation control for robotic manipulation of linear objects. In: *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. pp. 381–388. DOI:10.1109/AIM46323.2023.10196216.

Caporali A, Zanella R, Greogrio DD and Palli G (2022) Ariadne+: Deep learning–based augmented framework for the instance segmentation of wires. *IEEE Transactions on Industrial Informatics* 18(12): 8607–8617. DOI:10.1109/TII.2022.3154477.

Chaumette F (2007) *Visual Servoing*, chapter 6. John Wiley and Sons, Ltd. ISBN 9780470612286, pp. 279–336. DOI:https://doi.org/10.1002/9780470612286.ch6.

Coevoet E, Morales-Bieze T, Largilliere F, Zhang Z, Thieffry M, Sanz-Lopez M, Carrez B, Marchal D, Goury O, Dequidt J et al. (2017) Software toolkit for modeling, simulation, and control of soft robots. *Advanced Robotics* 31(22): 1208–1224.

Faure F, Duriez C, Delingette H, Allard J, Gilles B, Marchesseau S, Talbot H, Courtecuisse H, Bousquet G, Peterlik I et al. (2012) Sofa: A multi-model framework for interactive physical simulation. *Soft tissue biomechanical modeling for computer assisted surgery* : 283–321.

Ficuciello F, Migliozzi A, Coevoet E, Petit A and Duriez C (2018) Fem-based deformation control for dexterous manipulation of 3d soft objects. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4007–4013.

Goury O and Duriez C (2018) Fast, generic, and reliable control and simulation of soft robots using model order reduction. *IEEE Transactions on Robotics* 34(6): 1565–1576. DOI:10.1109/TRO.2018.2861900.

Hu Z, Han T, Sun P, Pan J and Manocha D (2019) 3-d deformable object manipulation using deep neural networks. *IEEE Robotics and Automation Letters* 4(4): 4255–4261. DOI:10.1109/LRA.2019.2930476.

Lagneau R, Krupa A and Marchal M (2020) Automatic shape control of deformable wires based on model-free visual servoing. *IEEE Robotics and Automation Letters* 5(4): 5252–5259.

Lee R, Hamaya M, Murooka T, Ijiri Y and Corke P (2022) Sample-efficient learning of deformable linear object manipulation in the real world through self-supervision. *IEEE Robotics and Automation Letters* 7(1): 573–580. DOI:10.1109/LRA.2021.3130377.

Ma Z and Xiao J (2023) Robotic perception-motion synergy for novel rope wrapping tasks. *IEEE Robotics and Automation Letters* .

MacQueen J (1967) Classification and analysis of multivariate observations. In: *5th Berkeley Symp. Math. Statist. Probability*. University of California Los Angeles LA USA, pp. 281–297.

Makiyeh F, Chaumette F, Marchal M and Krupa A (2023) Shape servoing of a soft object using fourier series and a physics-based model. In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. pp. 6356–6363.

Maneewongvatana S and Mount DM (1999) Analysis of approximate nearest neighbor searching with clustered point sets. *arXiv preprint cs/9901013* .

Navarro-Alarcon D, Yip HM, Wang Z, Liu YH, Zhong F, Zhang T and Li P (2016) Automatic 3-d manipulation of soft objects by robotic arms with an adaptive deformation model. *IEEE Transactions on Robotics* 32(2): 429–441. DOI:10.1109/TRO.2016.2533639.

Qi J, Ma G, Zhu J, Zhou P, Lyu Y, Zhang H and Navarro-Alarcon D (2022) 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. DOI:10.1109/TMECH.2021.3126383.

Qi J, Zhou P, Ran G, Gao H, Wang P, Li D, Gao Y and Navarro-Alarcon D (2024) Model predictive manipulation of compliant objects with multi-objective optimizer and adversarial network for occlusion compensation. *ISA Transactions* 150: 359–373. DOI:https://doi.org/10.1016/j.isatra.2024.05.015.

Ruan M, McConachie D and Berenson D (2018) Accounting for directional rigidity and constraints in control for manipulation of deformable objects without physical simulation. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 512–519. DOI:10.1109/IROS.2018.8594520.

Saghour C, Célérier M, Fraisse P and Cherubini A (2023) Visual servoing for dual arm shaping of soft objects in 3d. In: *IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*.

Seita D, Florence P, Tompson J, Coumans E, Sindhwani V, Goldberg K and Zeng A (2021) Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 4568–4575. DOI:10.1109/ICRA48506.2021.9561391.

Shetab-Bushehri M, Aranda M, Mezouar Y and Ozgur E (2022) Lattice-based shape tracking and servoing of elastic objects. *arXiv preprint arXiv:2209.01832* .

Shin C, Ferguson PW, Pedram SA, Ma J, Dutson EP and Rosen J (2019) Autonomous tissue manipulation via surgical robot using learning based model predictive control. In: *2019 International conference on robotics and automation (ICRA)*. IEEE, pp. 3875–3881.

Tsurumine Y, Cui Y, Uchibe E and Matsubara T (2019) Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation. *Robotics and Autonomous Systems* 112: 72–83. DOI:https://doi.org/10.1016/j.robot.2018.11.004.

Wang T and Yamakawa Y (2023) Edge-supervised linear object skeletonization for high-speed camera. *Sensors* 23(12). DOI:10.3390/s23125721.

Yu M, Lv K, Wang C, Jiang Y, Tomizuka M and Li X (2023) Generalizable whole-body global manipulation of deformable linear objects by dual-arm robot in 3-d constrained environments. *arXiv preprint arXiv:2310.09899* .

Zhou P, Zheng P, Qi J, Li C, Lee HY, Duan A, Lu L, Li Z, Hu L and Navarro-Alarcon D (2024) Reactive human–robot collaborative manipulation of deformable linear objects using a new topological latent control model. *Robotics and Computer-Integrated Manufacturing* 88: 102727.

Zhu J, Cherubini A, Dune C, Navarro-Alarcon D, Alambeigi F, Berenson D, Ficuciello F, Harada K, Kober J, Li X, Pan J, Yuan W and Gienger M (2022) Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics & Automation Magazine* 29(3): 67–77. DOI:10.1109/MRA.2022.3147415.

Zhu J, Navarro-Alarcon D, Passama R and Cherubini A (2020) Vision-based manipulation of deformable and rigid objects using subspace projections of 2d contours. *Robotics and Autonomous Systems* .

Zimmermann S, Poranne R and Coros S (2021) Dynamic manipulation of deformable objects with implicit integration. *IEEE Robotics and Automation Letters* 6(2): 4209–4216. DOI:10.1109/LRA.2021.3066969.