Lattice-Based Shape Tracking and Servoing of Elastic Objects

Mohammadreza Shetab-Bushehri, Miguel Aranda, Member, IEEE, Youcef Mezouar, Member, IEEE, and Erol Özgür

Abstract—In this article, we propose a general unified tracking-servoing approach for controlling the shape of elastic deformable objects using robotic arms. Our approach works by forming a lattice around the object, binding the object to the lattice, and tracking and servoing the lattice instead of the object. This makes our approach have full control over the deformation of elastic deformable objects of any general form (linear, thin-shell, and volumetric) in 3-D space. Furthermore, it decouples the runtime complexity of the approach from the objects’ geometric complexity. Our approach is based on the as-rigid-as-possible deformation model. It requires no mechanical parameter of the object to be known and can drive the object toward desired shapes through large deformations. The inputs to our approach are the point cloud of the object’s surface in its rest shape and the point cloud captured by a 3-D camera in each frame. Overall, our approach is more broadly applicable than existing approaches. We validate the efficiency of our approach through numerous experiments with elastic deformable objects of various shapes and materials (paper, rubber, plastic, and foam).

Index Terms—Manipulation of deformable objects, sensor-based control, visual servoing, visual tracking.

I. INTRODUCTION

Robotic manipulation of deformable objects is one of the branches of robotics that has attracted the most attention in recent years. This is because of many practical applications, including manufacturing [1], [2], [3], [4], surgical [5], [6], [7], and household [8], [9]. However, existing challenges in tracking and manipulating deformable objects prevented many of these applications from being put into practice. These challenges mainly stem from the well-known natural complexities of deformable objects compared with rigid objects, i.e., high degrees of freedom, various types of deformations, occlusions, and self-occlusions. In the literature, manipulation of deformable objects toward a specific desired shape using sensory feedback is named shape servoing [5]. In shape servoing, elastic objects have been the subject of interest in many practical applications. This stems from the fact that many objects around us behave elastically, i.e., tend to keep their rest shape during the manipulation. Shape servoing approaches for elastic deformable objects have been mainly categorized based on the general form of the object being manipulated, i.e., linear [1], [3], [10], [11], [12], [13], [14], [15], [16], [17], [18], thin-shell [4], [8], [9], [19], [20], [21], and volumetric [9], [21], [22], [23]. This is due to the distinct deformation characteristics and particular assumptions and simplifications that can be made for each one of these object forms. Among all of presented shape servoing approaches, there are several that can be applied on two or three different forms of the object [9], [20], [21]. These approaches, however, are merely tested on simple scenarios where the initial and desired shapes of the elastic deformable object are almost aligned. Indeed, the literature lacks a general shape servoing approach capable of being exploited as a universal solution for all forms of elastic deformable objects with any geometry and having full control over the object’s deformation in 3-D space.

In this article, we propose a novel general unified tracking-servoing approach that addresses this issue. The approach is composed of a tracking pipeline and a servoing pipeline. The tracking pipeline tracks the shape of the elastic deformable object in 3-D at every instant. The servoing pipeline employs the tracking data to drive the object to a desired shape. Our approach is general as it can handle linear, thin-shell, and volumetric elastic objects, not being limited to a specific form of object. It is unified as we solve jointly the two problems of tracking and servoing. A central idea of our approach is to form a 3-D lattice around the object. This lattice is bound to the object by geometrical constraints. We track and servo this lattice instead of the object itself. This is done by applying visual and deformation constraints on the lattice. We use the well-known as-rigid-as-possible (ARAP) model [24] as the deformation constraint. This model is based on a geometrical representation of elastic objects’ tendency for keeping their local rigidity. We present an analytical expression of the deformation Jacobian of the ARAP model. This deformation Jacobian, which we use on our lattice, is our...
model of how the robot motions deform the lattice. We obtain the Jacobian’s expression by rearranging the ARAP deformation equation and exploiting modeling tools presented in [3], [17], and [25]. Our Jacobian expression is directly applicable to both the whole object (full shape), or a region of it (partial shape). This gives us the useful ability to perform both full or partial shape servoing. We propose a robotic control law based on this Jacobian. Using this control law, we servo the lattice and thus the object toward the desired shape. The inputs to our approach are the point cloud of the object’s surface in its rest shape and the point cloud captured by a 3-D camera in each frame. We use ARAP both for tracking and for servoing. ARAP has been employed frequently for tracking elastic deformable objects [26], [27], [28], [29], including in robotic manipulation scenarios [9], [30]. We first proposed to use ARAP for servoing in [4]; our main insight was that, when used in a shape-feedback control loop, ARAP provides a precise enough representation of the elastic object’s shape dynamics. Overall, ARAP is a simple and well-established model and it allows working with elastic deformable objects without having to use mechanical parameters. In this article, we show how to achieve both tracking and servoing requiring only this single model.

The idea of exploiting a lattice to manage the object’s deformation has been previously utilized in manipulating shapes in computer graphics when the shape’s representative mesh is too high resolution to be directly manipulated [31]. We use the same idea to transfer the object’s deformation to the lattice and simplify the deformation. This is reasonable as for many objects, despite all the details on their surface (which are represented by a detailed mesh), the object cannot be deformed in more than several mostly simple ways. These deformations can, thus, be effectively represented by the lattice encapsulating the object. In fact, we solve the deformation equations for the lattice instead of the object. At the same time, we use geometrical constraints between the lattice and the object as binding constraints. This, concretely, decouples the runtime complexity of the approach from the objects’ geometric complexity and makes the whole tracking-servoing process run much faster. Furthermore, using a lattice makes our approach capable of being generalized for any form of the object (linear, thin-shell, and volumetric). This is due to the fact that we develop our tracking-servoing approach for a 3-D lattice that is formed in the same way for any object geometry. While our approach follows the standard steps in sensor-based shape control, our innovative use of a lattice and geometrical modeling allows us to improve in terms of generality, unification of tracking and servoing, efficiency, and simplicity.

Our specific contributions are detailed in the following.

**Contribution.** We list our contributions as follows.

1) We present a general unified tracking-servoing approach for elastic objects that can handle any object form (linear, thin-shell, and volumetric) with any geometry. Our approach does not require any mechanical parameter of the object to be known. Instead, we only use a point cloud that represents the object’s surface with sufficient accuracy.

2) Our approach does not use the texture of the object. It only employs the point cloud captured in each frame by a 3-D camera as the input. This brings the advantage to our approach that it works with objects without specific textures.

3) We present a novel analytical expression of the deformation Jacobian of the ARAP model. This avoids the need for numerical approximations.

4) Our approach has full control over the object’s deformation in 3-D space. This means that we can start servoing the object from an initial shape, translate, rotate, and deform the object toward a desired shape that is characterized by a totally different visible part of the object in comparison to the visible part of the object at the initial configuration. To our knowledge, this is the first approach that is proved to handle such scenarios in practice.

5) The idea of using a lattice makes tracking and servoing much faster as we deal with the deformation equations for the lattice and not the object. This makes the execution speed of the approach to a high extent independent of the object’s geometric complexities. As a result, our approach runs fast and is needless of any specific hardware. The execution speed of our tracking-servoing approach reaches 20–30 FPS during our experiments without any parallelization using only CPU.

6) The servoing pipeline can servo the whole or a part of the object. The definition of the servoed regions of the objects and its implementation in the servoing pipeline is easy and straightforward.

7) Our approach is easily scalable in terms of increasing the number of manipulating robots. Moreover, increasing the number of robots does not impose considerable additional execution time on the servoing approach.

We present a diverse experimental evaluation where our approach shows remarkable performance. Our setup is illustrated...
in Fig. 1. We use objects of different forms (linear, thin-shell, and volumetric) and materials (paper, rubber, plastic, and foam). Our approach can handle particularly difficult scenarios including: large nonplanar deformation of a cable, servoing of a convoluted foam with a spiky surface, partial servoing of disjoint object regions, or simultaneous twisting and bending of a volumetric foam. We also illustrate that our approach has limitations in certain scenarios that require additional techniques (e.g., driving the object through its singular shape).

II. RELATED WORK

A. Shape Tracking of Deformable Objects

Tracking deformable objects has been the subject of many studies in computer vision. The main criterion for categorizing these methods is the type of used sensor, i.e., monocular camera [2], [26], [27], [28], [29], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], or 3-D camera [43], [44], [45], [46]. Despite the considerable progress made in this area, the employment of these methods in robotic studies was limited. This stems from the following several reasons.

1) The codes of the deformable object tracking methods in the literature are not open access, or if they are, they are challenging to adapt to a new problem.

2) These methods are primarily object-specific, meaning that plenty of adjustments should be made to apply them to a new object. In the case of analytical methods, these adjustments might include creating a template of the object (comprising 3-D mesh, texture-map, and UV map) and setting the mechanical parameters of the object [26], [34], [36], [37], [43]. Regarding the neural-network-based methods, these adjustments include creating a new synthetic dataset of the new object of interest and training the whole network from the beginning with that new synthetic dataset, followed by a possible fine-tuning step [28], [29], [39], [40], [41].

3) Many of the presented tracking methods in the literature are specified to a particular object form, including linear [44] or thin-shell [32], [33], [34], [42], [45], [46]. As a result, they cannot be used as a general tracking solution to different object forms.

4) As these tracking methods are presented in the field of computer vision, they have not exploited the common priors existing in the field of robotics that can be considerably helpful throughout the tracking period, such as the known coordinates of the robotic grippers.

As a result of the mentioned reasons, researchers in the field of robotics have turned to track a simplified representation of objects instead of the whole shape. This simplified representation can be a handful of points on the surface of the object [2], [7], [17], [20], [23], [47], [48] or the point cloud of the visible part of the object that remains almost the same throughout the manipulation [8], [9], [22].

In contrast to the previous works in robotics, we present a robust shape tracking pipeline along with our shape servoing pipeline that fully tracks the shape of objects of any form and geometry. A point cloud of the object in its rest shape is the only information that our pipeline requires. It is also well designed for robotic applications as it exploits common priors, such as robotic grippers’ known coordinates. The novel idea of using a lattice to simplify object deformation places our shape tracking pipeline among the fastest methods in the literature. This is while our tracking pipeline does not need any special hardware to run. We highlight that the shape tracking pipeline can also be used independently of our servoing pipeline.

B. Model-Based Shape Servoing

Deformation models facilitate shape servoing of deformable objects by providing the knowledge of objects’ behavior under deformation. We categorize the deformation models as mechanical and geometrical models. Mechanical models predict the object’s shape by solving mechanical equations under constraints of force or displacement. Finite element method (FEM) has been the most widely-used mechanical model in the literature [3], [23], [49], [50]. In this context, Duenser et al. [49] presented an open-loop simulation-based control methodology wherein the desired deformations are directly mapped to joint angle commands. Ficuciello et al. [23] used FEM in an open-loop control approach for in-hand soft object manipulation. Koessler et al. [3] introduced a closed-loop controller employing a computationally-efficient FEM that exploits a partition of mesh nodes. The authors applied their approach to a linear object. Sanchez et al. [50] used only force feedback along with FEM to servo a volumetric object. Lv et al. [14] presented a dynamic model that simulated stretching, bending, and twisting deformation in linear objects. The authors exploited this model to deploy a linear object onto a plane using both single-arm and dual-arm setups. Although mechanical models can be considered a reliable solution for predicting object deformation, the computation is often costly and depends on the intrinsic mechanical properties of the object, which vary from one to another. On the contrary, our proposed shape servoing pipeline is fast and does not require any prior information on the mechanical parameters of the object.

Unlike mechanical models, geometrical models simulate the object deformation considering only geometrical constraints, such as preserving the local rigidity. This makes these models independent of the mechanical parameters of the object. Aranda et al. [2] drove the object toward the desired shape by defining intermediary shapes. The intermediary shapes were calculated by applying position-based dynamics [51] as the deformation model. In our prior work [4] we proposed to employ the ARAP deformation model to fully servo thin-shell objects. We computed a deformation Jacobian via numerical differentiation: specifically, in simulation, we perturbed each controlled DOF and computed the resulting change of the object’s shape using ARAP. This shape servoing approach was shown to be fast and could be used for objects of various materials. Later, the authors in [18] and [52] exploited the same deformation Jacobian with an optimal controller. Aghajanazadeh et al. [17] proposed a novel offline Jacobian to be used in servoing linear objects in 2-D space. This Jacobian was based on an as-similar-as-possible
deformation model, in which the object has a tendency to preserve its original shape up to a similarity transformation of that shape.

In comparison to these studies, which are restricted to either linear or thin-shell objects, our proposed shape servoing pipeline can be applied to objects of any form thanks to the novel idea of exploiting a lattice.

C. Model-Free Shape Servoing

Model-free shape servoing can be considered as the most studied approach in the literature. In these approaches, no prior information on the object deformation is required. We divide these approaches into two main categories: sensor-based deformation Jacobian, and geometric heuristics. As for the first category of approaches, online sensor measurements are employed to estimate a deformation Jacobian to be used in shape servoing [1], [5], [6], [8], [10], [21], [47], [48]. These sensor measurements are taken by a 2-D or 3-D camera while the object is manipulated by robots. This Jacobian is then used to control a simplified representation of the object, i.e., several sampled points on the object’s surface [6], [8], [47], [48], or the object’s contour [1], [5], [21] in the image space. In general, requiring the robot’s motion for calculating the Jacobian makes these approaches more complex, and sensitive to noise. Conversely, our proposed approach is less noise-sensitive, we can control the full shape of the object in 3-D space, and we compute our Jacobian without needing robot motions.

The work of authors in [19], [20], and [53] exploited a deformation Jacobian estimated using a geometric heuristic (diminishing rigidity). This approach is fast and has been tested in simulation and real experiments with simple deformations. However, the lack of an object deformation model in calculating the Jacobian limits its practicability in many real scenarios. In contrast to this approach, our Jacobian is calculated analytically based on a geometric deformation model.

D. Learning-Based Shape Servoing

In recent years, the field of deformable object manipulation has been aligned with the growing trend toward using learning-based approaches. Different studies have been conducted in this field [7], [9], [22], [54], [55], [56], [57] among which reinforcement learning (RL) was the most widely-used approach [54], [55], [56], [57]. Despite the relative success of these approaches, they still suffer from significant limitations. First, they are specified to a particular type of the object, such as linear [56], [57] or thin-shell [7], [54], [55], or just a surface of a volumetric object [9], [22]. Furthermore, they have been mostly tested in simulation, and if in real, the deformations have been simple [7], [9], [22]. Except in [9] where an online learning process was used to train the agent (i.e., robot) in real, the other studies trained their agents in simulation with objects of specific mechanical parameters. This makes them suffer from the well-known sim-to-real gap. In addition, training the agent for a new object requires a lot of data to provide, which is a tedious task for deformable objects.

In contrast to these approaches, we do not need any training and our shape servoing pipeline provides full control over the deformation of elastic objects of any form.

III. APPROACH DESCRIPTION AND NOTATION

The problem that we address is tracking an elastic deformable object with any general form (linear, thin-shell, and volumetric) and manipulating it by robotic arms in a feedback control loop so that its shape gradually conforms to a desired shape. To formulate this problem, we use the following notation throughout this article.

1) Scalars: Italic lower-case letters.
2) Vectors: Bold lower-case letters.
3) Matrices: Bold upper-case letters.
4) Sets: Calligraphic letters.

Note that for some developments, we represent vectors of points as 2-D arrays (i.e., with one column for each spatial dimension), following [24] and [31]. We define flattening a vector as transposing the rows of the vector and stacking them together in a one-column vector. For example, given a vector \( w \) of dimension \( m \times n \), we define the flattened \( w \) as \( w_f \) of dimension \( mn \times 1 \).

We start by considering a set of robots \( M = \{1, 2, \ldots, m\} \) that firmly grasp the object during manipulation. We also assume that the forward kinematic models of the robots are known. We use a 3-D camera as the sensor providing the input point cloud data. The relative poses between the camera and the robots are assumed to be known. Our goal is to track the object at each instant and introduce a control scheme that computes the 6-DOF velocity vectors associated with all the robots’ grippers stacked in a column vector of length \( 6m \): \( v = [v_1^T, \ldots, v_m^T]^T \). Fig. 2 illustrates the flowchart of our proposed approach.

We use a point cloud of the object’s surface to represent the object’s shape. This point cloud is generated when the object is undeformed, i.e., is at its rest shape. We use the \( n \times 3 \) vector \( p \) to represent this point cloud. The resolution of this point cloud should be fine enough to represent the object’s geometry with sufficient accuracy throughout the tracking period. This will be explained more in Section IV. We, then, define the lattice \( s \), an \( n_l \times 3 \) vector, encapsulating \( p \) where \( n_l \) is the total number of lattice nodes. The lattice \( s \) contains \( p \). Specifically, the metric length of the lattice \( s \) in each spatial dimension is chosen slightly larger than the length of the object \( p \). As the lattice will represent the deformation of the object, its orientation should be set in a way that the lattice axes be aligned with the main axes that the object bends around. Furthermore, the resolution of \( s \) is selected high enough in each direction to be able to represent the deformation of the object in that direction effectively. Typically \( n_l \) can be chosen much smaller than \( n, i.e. \), the lattice is a compact representation of the object that allows for faster processing. We generate a tetrahedral mesh over the lattice nodes in \( s \). An example of the formed lattice and the tetrahedral mesh around a shoe sole is illustrated in Fig. 3. We form the set \( T_f \) for each point \( p_j \) in \( p \) as the indices of lattice nodes belonging to the tetrahedral cell encapsulating \( p_j \). We can, then, express \( p_j \) as a linear combination of lattice nodes in

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 2. Flowchart of the proposed shape tracking-servoing approach. In each frame, the tracking pipeline takes the captured point cloud in the current frame and $s$ (and thus $p$) from the previous frame as the input and estimates $s^*$ (and thus $p^*$) in the current frame. $s^c$ and $R^b_i$ are sent to the servoing pipeline where an analytical expression for a deformation Jacobian is calculated. We use this Jacobian in a control law to send proper commands to the robots to guide $s^c$ toward $s^*$ and consequently $p^c$ toward $p^*$.

Fig. 3. Lattice $s$ encapsulating the point cloud $p$ of a shoe sole at its rest shape. We also generate a tetrahedral mesh over the lattice nodes.

$T_j$ using barycentric coordinates $\alpha_{i,j}$ as follows:

$$\sum_{i \in T_j} \alpha_{i,j}s_i = p_j. \quad (1)$$

This equation serves as a geometrical constraint between $p$ and $s$. In the rest of this article, by the shape of object and lattice, we refer to the 3-D coordinates of the points in $p$ and $s$, respectively. We also define the following superscripts that will henceforth be used to specify the state of different shapes:  
1) current shape $c$;  
2) desired shape $*$;  
3) undeformed shape $u$.

In the following sections, we describe the proposed tracking and servoing pipelines.

IV. TRACKING PIPELINE

Our proposed tracking pipeline is able to track a 3-D elastic deformable object in real time. It comprises four steps, namely: capturing and filtering point cloud, rigid registration, finding corresponding points, and finally, applying the deformation model. The first three steps of the pipeline are inspired by Petit et al. [43]. The results of different steps of the tracking pipeline in tracking a shoe sole are shown in Fig. 4.

The pipeline tracks the shape of the object in each frame from a known initial shape in 3-D space. Knowing the initial shape of the object at the beginning of tracking is common in the state-of-the-art [36], [37], [43], [44]. We initially set the object and lattice shapes ($p^c$ and $s^c$) by rigidly transforming $p^u$ (and thus $s^u$) to an initial pose (i.e., a reference shape) in 3-D space visible from the camera. For starting the tracking pipeline, we approximately align the object with the reference shape and trigger tracking. The alignment can be done by hand or by robots while grasping the object. To facilitate the process of alignment, we visualize the reference shape in 3-D space and its projection in 2-D. It should be noted that this alignment only needs to be partially consistent. This means that even if the real object is slightly displaced and deformed with respect to the reference shape, the tracking pipeline is able to infer a correct $p^c$ after several frames. This is mainly thanks to the second step of our pipeline, i.e., rigid registration. We will explain this further in Section IV-B.

A. Step 1: Capturing and Filtering Point Cloud

The pipeline starts by capturing and filtering the point cloud from a 3-D camera in each frame. The purpose of filtering is
is essential to deal with rapid movements of the objects as it compensates for the rigid nonalignment between $p^c$ and $d_f$ in each frame. This is necessary to have a fair initialization for executing the next step of the tracking pipeline (i.e., finding corresponding points). $p^c_i$ is determined in each frame through a two-step process. First, the normals of $p^c$ are calculated. Second, the points in $p^c$ with an angle of more than 90° between the normal of that point in $p^c$ and the sightline passing through that point are selected. With this procedure, the selected points are in the visible surface of the object. After rigidly registering the object, we update $p^c_i$ to be used in the next step of the tracking pipeline.

C. Step 3: Finding Corresponding Points

In this step, we find correspondences between the points in $p^c_i$ and $d_f$. We use the ICP-like algorithm suggested in [43]. In this algorithm, first, by employing K-d tree searches, nearest neighbor correspondences are determined, both from $p^c_i$ to $d_f$ and vice versa. Then, using these two sets of correspondences, a corresponding point in 3-D space is computed for each point in $p^c_i$. This is done by a two-step process: first, we average the coordinates of all the points in $d_f$, having the same nearest neighbors in $p^c_i$. Second, we average the resulting coordinates from the previous step and the coordinates of the points in $d_f$, which are the nearest neighbors of each point of $p^c_i$. In this stage, we remove the points in $p^c_t$ whose distance from their corresponding point is more than a specific threshold. We name the vector of 3-D coordinates of these remaining points in $p^c_t$ as $p^c_a$ and the vector of their corresponding 3-D coordinates in 3-D space as $p^c_{a'}$. We indicate the number of points in $p^c_a$ and $p^c_{a'}$ with $n_a$.

D. Step 4: Applying Deformation

In this step, we apply deformation to the inferred shape. This deformation is applied to the lattice and not to the object and the visual constraints related to the object are imposed as the constraints to this deformation. This way, we solve for the shape of $s^c$ and not $p^c$, which is a faster process as $n_t$ is normally much smaller than $n$. It should be noted that $s^c$ and $p^c$ remain connected due to the spatial constraints in (1). We deform $s^c$ by solving the following linear system, which is a modified version of the one suggested by [31]

$$\mathbf{(L_B + B^T B)} s^c = \mathbf{Gb + B^T b}. \quad (2)$$

The following two series of constraints are considered in this equation.

1) ARAP Constraints: These constraints ensure that $s^c$ and consequently $p^c$ try to keep their local rigidity. These constraints are imposed by matrix $L$ on the left-hand side and vector $b$ on the right-hand side of (2). $L$ is the $n_t \times n_t$ Laplacian matrix of the tetrahedral mesh formed over the lattice, and $b$ is a $n_t \times 3$ vector whose $i$th row is

$$b_i = \sum_{j \in N_i} \frac{w_{ij}}{2} \left( R_i + R_j \right) (s_i^j - s_j^i). \quad (3)$$

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
In this equation, \( \mathcal{N}_i \) is defined as the set containing the first-order neighbors of node \( i \) in \( s \), \( w_{ij} \) is a scalar encoding the connection between nodes \( i \) and \( j \) in \( s \), and \( \mathbf{R}_i \) is the optimal rotation matrix that conforms the nodes of the set \( \mathcal{N}_i \) in \( s^o \) to the same nodes in \( s^r \) with least-squares error. \( \mathbf{R}_i \) is computed using the singular value decomposition (SVD) (for more details see [24]).

2) Object Visual Constraints: These constraints try to minimize the error between the points in \( p^o_i \) and their corresponding points in \( p^r_i \). They are imposed by \( \mathbf{B}^\top \mathbf{B} \) on the left-hand side and \( \mathbf{B}^\top \mathbf{t} \) on the right-hand side of (2). \( \mathbf{t} \) is an \( n_s \times 3 \) vector containing the 3-D coordinates in \( p^o_i \) as rows. \( \mathbf{B} \) is an \( n_s \times n_l \) matrix containing the constraints from (1) as rows for each row of \( \mathbf{t} \).

We adjust the effectiveness of these two constraints using \( \Gamma \). We consider \( \Gamma \) as a diagonal matrix with the size of \( n_l \times n_l \) to attribute different values to each node of \( s^r \). This is concerned with the ARAP constraints as they strongly try to rigidly maintain the shape of \( s^r \) and consequently \( p^r \). We, thus, set two different values for each diagonal element of \( \Gamma \): 0.1 if for the nodes of \( s^o \) belonging to a tetrahedral cell that surrounds a point of \( p^o_i \), and 1 for other nodes. As a result, the nodes in \( s^r \) being subject to deformation by object visual constraints can flexibly move, and thus, the points in \( p^o_i \) can be driven toward their corresponding 3-D coordinates in \( p^o_i \). At the same time, the other nodes in \( s^r \) keep the local rigidity, and thus, the general shape of the lattice can be maintained. Solving for \( s^r \) is not possible through a one-step solution. Instead as explained in [31], an iterative flip-flop solution is used. In each flip-flop iteration, we calculate \( \mathbf{R}_i \) (that is a function of \( s^r \)) and use it in solving the linear system of (2) for \( s^r \). We consider (2) converged when the difference between the two successive calculated \( s^r \) is smaller than a certain value. In our experiments, the solution converges in 5–8 iterations. As \( \mathbf{B} \) is sparse, we solve (2) for \( s^r \) using the conjugate gradient solver of Eigen library for sparse least-square problems.

Another constraint that can be incorporated into the tracking pipeline is the known 3-D coordinates of several lattice nodes. These lattice nodes can be considered as rigidly attached to a known point in 3-D space. In practice, this known point can be, e.g., a fixed point of the object or a robot’s gripper. The indices of these lattice nodes are determined manually by the user before starting the tracking pipeline or automatically by selecting the closest lattice nodes to the known point in 3-D space at the beginning of tracking. In both cases, at the beginning of the tracking pipeline, the relative 3-D coordinates of the selected lattice nodes with respect to their corresponding 3-D known points are saved. These relative coordinates along with the coordinates of the known points at each instant give us the 3-D coordinates of the selected lattice nodes. As for the grippers, their poses in the camera frame can be computed knowing the configuration of the robot and the calibration between the robots and the camera. The implementation of this constraint can be performed by modifying (2) before solving it. This is done by removing corresponding rows and columns from the left-hand side and recalculating the right-hand side with the known 3-D coordinates of the relevant lattice nodes. The use of these constraints is optional; they make tracking more precise and robust, specifically in scenarios with large deformations or the presence of self or external occlusions. We provide illustrative videos of tracking-only experiments on the project website.\(^1\)

\[ \text{V. SERVOING PIPELINE} \]

In this section, we explain our shape servoing pipeline. The same as in our previous work [4], we use the ARAP deformation model to obtain a deformation Jacobian. The control law we propose, then, is based on this Jacobian. In [4], we computed the Jacobian numerically, by simulating perturbations in every DOF of the grippers that grip the object. In contrast, here we use an analytical expression of the Jacobian. The advantages of this are: the Jacobian we compute does not involve any numerical approximation, and its computation is fully scalable as the number of grippers grows. Another main difference is that we derive the formulation for the lattice and not the object. This not only generalizes the servoing formulation for any form of the object, but also decouples the runtime complexity of the servoing from the objects’ geometric complexity. We use this Jacobian to propose a control law. Finally, we apply the control law on the lattice to guide \( s^r \) toward \( s^o \) and consequently \( p^o \) toward \( p^r \) thanks to the spatial constraints between the lattice and the object in (1). We present the different steps of the servoing pipeline in the following.

A. ARAP Deformation Jacobian

The main ingredient of our shape servoing pipeline is the Jacobian that expresses how the infinitesimal gripper motions change the shape of the object. This is called deformation Jacobian in the literature. Most shape servoing works [3], [4], [5], [8], [17], [19], [21], [48], [49] assume that the object’s shape changes quasi-statically and that this change can be represented by such deformation Jacobian. In this article, we also make the same assumption. As in our previous work [4], the main principle of our servoing pipeline is approximating the true deformation Jacobian of the object by the deformation Jacobian of the ARAP model of the object. In particular, we present an analytic derivation of the deformation Jacobian of the ARAP model. We start with the main ARAP linear system

\[ \mathbf{L}_s = \mathbf{b} \]  

which is (2) with only deformation constraints. The principle of ARAP’s modeling [24] is that the object’s shape (in our case the lattice’s shape \( s \)) in quasi-static equilibrium minimizes the ARAP deformation energy. In particular, this energy is at a local minimum under the existing constraints (e.g., regions of the object that are grasped and moved by a robot). The gradient of this energy with respect to \( s \) has the following form:

\[ \mathbf{g} = \mathbf{L}_s - \mathbf{b} \]  

\(^{1}\) [Online]. Available: https://sites.google.com/view/tracking-servoing-approach
omitting multiplicative constants that are irrelevant to our purpose. \(b\) contains the \(b_i\) for each node \(i\) in \(s\), which can be written as in (3). As the stable shape \(s\) is a local minimum, one makes the gradient zero to find \(s\): this is what (4) expresses. In the standard ARAP formulation, the optimal rotations \(R_i\) in (3) depend on \(s\) via SVD. Note that in this section \(s\) represents any shape in the neighborhood of \(s^o\). This is because what we want to do is to compute the deformation Jacobian at \(s^o\). Then, our insight is that the optimal rotations for \(s\) will also be in the neighborhood of the optimal rotations for \(s^o\). Therefore, we can express the relative rotation between them as an infinitesimal rotation. We will show that this infinitesimal rotation can be parameterized as a linear function of the node positions, avoiding the SVD. With this linear parameterization, we will transform (5) in an expression where the dependency on \(s\) is fully linear. From this expression we will derive the deformation Jacobian.

We start by writing the rotation matrix \(R_i\) in (3) as the multiplication of a general rotation matrix \(R^g_i\) and an infinitesimal rotation matrix \(R^s_i\)

\[
R_i = R^g_i R^s_i. \tag{6}
\]

\(R^g_i\) is the optimal rotation matrix that best conforms the nodes of the set \(N_i\) in \(s^o\) to the same nodes in \(s^c\). Note that this rotation matrix \(R^g_i\) is known for us: it is the last optimal rotation matrix computed by our tracking pipeline. We are computing the Jacobian for changes of shape in the neighborhood of the current shape. Therefore, for the Jacobian computation, \(R^g_i\) is considered a fixed matrix. \(R^s_i\), the infinitesimal rotation matrix, can be expressed as a linear function of \(s_i\) using the estimation suggested by [25]. According to [25], when \(s_i\) is in the local neighborhood of \(s^c_i\), one can write an approximation for the transformation matrix \(T_i\) between \(s^c_i\) and \(s_i\) that is a linear function of \(s_i\). The transformation matrix \(T_i\) is a \(4 \times 4\) matrix including translation, rotation, and scale. Here, what we need is the rotation part of this transformation. We, consequently, isolate the rotation part of the \(T_i\) formulation presented in [25] and use it as \(R^s_i\). We can thus write \(R^s_i\) as the following infinitesimal rotation matrix:

\[
R^s_i = \begin{pmatrix}
1 & -h_{i3} & h_{i2} \\
h_{i3} & 1 & -h_{i1} \\
-h_{i2} & h_{i1} & 1
\end{pmatrix}. \tag{7}
\]

As this is a least-squares optimal rotation, \(h_{i1}\), \(h_{i2}\), and \(h_{i3}\) can be computed from

\[
\begin{pmatrix}
h_{i1} \\
h_{i2} \\
h_{i3}
\end{pmatrix} = (A^T_i A_i)^{-1} A^T_i w_i. \tag{8}
\]

We consider that the reference shape for this least-squares computation is \(s^c\). Therefore, \(A_i\) and \(w_i\) can be written as

\[
A_i = \begin{pmatrix}
s_{k_1} & -s_{k_2} & 0 & \cdots \\
-s_{k_1} & s_{k_2} & 0 & \cdots \\
s_{k_3} & -s_{k_4} & 0 & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{pmatrix}, \quad k \in \{i\} \cup N_i \tag{9}
\]

and

\[
w_i = \begin{pmatrix}
s_w \\
s_{k_1} \\
s_{k_2} \\
\vdots
\end{pmatrix}, \quad k \in \{i\} \cup N_i. \tag{10}
\]

\(A_i\) is a known \(3d_i \times 3\) matrix \((d_i\) is the number of elements in \(\{i\} \cup N_i\)) composed of the elements from the current shape, and \(w_i\) is a \(3d_i \times 1\) vector of unknowns comprising \(s_i\) and its first-order neighbors. By putting (9) and (10) in (8) and then in (7) we can have \(R^s_i\) as

\[
R^s_i = \begin{pmatrix}
1 & -M_{i3},w_i & M_{i2},w_i \\
M_{i3},w_i & 1 & -M_{i1},w_i \\
-M_{i2},w_i & M_{i1},w_i & 1
\end{pmatrix} \tag{11}
\]

where \(M_i = (A^T_i A_i)^{-1} A^T_i M_i\) is a \(3 \times 3d_i\) matrix and the subscript \(k, *\) indicates the \(k\)th row of the matrix \(M_i\). We rearrange \(R^s_i\) as the following:

\[
R^s_i = I + \begin{pmatrix}
0_{3d_i},w_i & -M_{i3},w_i & M_{i2},w_i \\
M_{i3},w_i & 0_{3d_i},w_i & -M_{i1},w_i \\
-M_{i2},w_i & M_{i1},w_i & 0_{3d_i},w_i
\end{pmatrix} \tag{12}
\]

where \(I\) is a \(3 \times 3\) identity matrix, and \(0_{3d_i}\) is a row vector of zeros with the length of \(3d_i\). Replacing (12) in (6) and then in (3) and rearranging the terms, we have

\[
b_i = \sum_{j \in N_i} \frac{w_{ij}}{2} I(R^o_i + R^g_j)(s^c_i - s^c_j) + \sum_{j \in N_i} \sum_{k \in N_i} q_i w_i + \sum_{j \in N_i} q_j w_j \tag{13}
\]

where \(q_i\) and \(q_j\) are known \(1 \times 3d_i\) and \(1 \times 3d_j\) vectors that can be written as the following:

\[
q_i = \frac{w_{ij}}{2}(u_i[1](M_{i3}, - M_{i2},.) \\
+ u_i[2](M_{i1}, - M_{i1,}.) \\
+ u_i[3](M_{i2}, - M_{i1,}.) \tag{14}
\]

\[
q_j = \frac{w_{ij}}{2}(u_j[1](M_{j3}, - M_{j2,}.) \\
+ u_j[2](M_{j1,} - M_{j1,}.) \\
+ u_j[3](M_{j2}, - M_{j1,}.)) \tag{15}
\]

In these equations, \(u_i = R^g_i(s^c_i - s^c_j)\) and \(u_j = R^g_j(s^c_i - s^c_j)\) are both \(3 \times 1\) vectors, and \([k]\) signifies the \(k\)th element of the vector.

On the right-hand side of (13), the first sum is a known and constant vector, while the second and third sums are linear functions of \(w_i\) and \(w_j\). One point that should be noted is that for solving the original formulation of ARAP in (4), as described in Section IV-D, a flip-flop solution is used. This solution comprises two steps: calculating optimal rotations and solving the ARAP linear system. In calculating optimal rotations, the directions of \(x, y,\) and \(z\) are dependent in the SVD. In solving the ARAP linear system, however, having the optimal rotation matrices calculated, one can solve (4) for each
direction independently regardless of the other directions. In our new formulation of $b$ in (13), we propagate the dependency between these directions (coming from the second and third sums) into the equation. This dependency stems from the usage of the estimation for rotation matrices [see (7)–(10)]. In order to integrate this dependency with other terms in the original formulation of ARAP, we rewrite each term in (4). This is done as follows.

i) We define $s'$ as the flattened form of $s$ with the size of $3n_l \times 1$.

ii) We define $L'$ as a $3n_l \times 3n_l$ matrix such that $L' = L \otimes I$ where $I$ is a $3 \times 3$ identity matrix and $\otimes$ denotes the Kronecker product.

iii) We define $b'$ as the following:

$$b' = c + Qs'$.$} \hspace{1cm} (16)$$

In this equation, $b'$ with the size of $3n_l \times 1$ is the flattened form of $b$ with the size of $n_l \times 3$. $c$ is a $3n_l \times 1$ vector that represents the general form of the first sum of (13) consisting of $n_l$ vectors of all lattice nodes (each of size $3 \times 1$) concatenated together. The multiplication of $Qs'$ includes all the elements in the second and third sums in (13). The same as in $ii)$, $s'$ is the flattened form of $s$ with the size of $3n_l \times 1$. $Q$ is a $3n_l \times 3n_l$ known matrix that is initially filled with zeros and then the values from $q_i$ and $q_j$ will be added to their corresponding elements. Each consecutive three rows in $Q$ belong to three directions of one lattice node. We set these three rows identical. This comes from the fact that we apply the constraints from the second and the third sums in (13) that depend on the three directions to each one of the directions identically. Each consecutive three columns of $Q$ are dedicated to the three directions of each lattice node. The elements in $q_i$ and $q_j$ are summed with the elements in the rows corresponding to the $i$th lattice node at their corresponding columns regarding their indices and their directions. The process of filling $Q$ can be formulated by the following equation:

$$Q(3(i - 1) + r, 3(\mathcal{K}[k] - 1) + c) = \sum_{j \in \mathcal{N}_i} q[j(3(k - 1) + c)]$$ \hspace{1cm} (17)

where $q \in \{q_i, q_j\}$, $i \in \{1, \ldots, n_l\}$, $r, c \in \{1, 2, 3\}$ and $\mathcal{K} = i \cup \mathcal{N}_i$, if $q = q_i$;

$\mathcal{K} = j \cup \mathcal{N}_j$, if $q = q_j$.

Using flattening and (16), we can write (5) as

$$g' = L's' - b' = Hs' - c$$ \hspace{1cm} (19)

where $g'$ with the size of $3n_l$ is the flattened form of $g$, and $H = L' - Q$ is a known $3n_l \times 3n_l$ matrix. We will define the deformation Jacobian in terms of velocities (i.e., time variations). Note that for this Jacobian computation, $H$ and $c$ are constant, as they depend on fixed quantities. Therefore, by taking derivative from (19) with respect to time, we have $Hs' = g'$.

The final step of our derivation is to obtain the deformation Jacobian from this equation. For this, we apply on our ARAP model an approach that is based on node partitioning. This approach has been previously applied in [3] on a FEM model, and in [17] on an offline geometrical model. In [3] and [17], this approach was defined for linear objects deforming in 2-D; here, we widen that scope as we consider objects of arbitrary form (linear, thin-shell, and volumetric), which deform in 3-D.

We, first, categorize the nodes of the lattice into three sets; free, servoed, and gripped, having in turn $n_f$, $n_s$, and $n_g$ elements such that $n_f + n_s + n_g = n_l$. Accordingly, we divide the flattened position vector of the lattice nodes $s'$ into $s'_f \in \mathbb{R}^{3n_f}$, $s'_s \in \mathbb{R}^{3n_s}$, and $s'_g \in \mathbb{R}^{3n_g}$. Likewise, we partition matrix $H$. Under ARAP modeling, the gripped nodes are moved externally, while the motions of the free and servoed nodes are determined only by the ARAP deformation energy. As the object is always in quasi-static equilibrium, its shape corresponds to a locally minimum energy. Therefore, for the free and servoed nodes the ARAP energy gradient has to be always zero. Hence, the time derivative of the gradient is also zero. Therefore, the expression $Hs' = g'$ above takes the following form:

$$\begin{bmatrix} H_{gg} & H_{gs} & H_{gf} \\ H_{sg} & H_{ss} & H_{sf} \\ H_{fg} & H_{fs} & H_{ff} \end{bmatrix} \begin{bmatrix} s'_g \\ s'_s \\ s'_f \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$ \hspace{1cm} (20)

From (20), we can obtain the following expression linking the velocities of the gripped and the servoed nodes:

$$s'_s = J_{sg} s'_g$$ \hspace{1cm} (21)

As in [3], [17], and [58], we assume the matrices that have to be inverted are full-rank. This stems from the initial made assumption that the shape is fully constrained by the grippers. Thus, $J_{sg}$ is the ARAP deformation Jacobian, which we have obtained analytically from the knowledge of the current shape $s^c$, the optimal rotations $R^o$, and the ARAP model parameters.

B. Control Law for Shape Servoing

We use $J_{sg}$ to propose a proportional control law to drive the positions of the servoed nodes of the lattice toward their desired values. We, first, define the servoing error as

$$e_s = s'^c - s'^v.$$ \hspace{1cm} (23)

Using (21) and (23) we can write our proportional control law as the following:

$$v_g = -k_g J_{sg} e_s$$ \hspace{1cm} (24)

where $k_g$ is a positive gain, $\dagger$ signifies the pseudoinverse, and $v_g$ is the vector of translational velocities to be applied to the gripped lattice nodes. In a common robotic application, the object is controlled by a robotic gripper with 6 DOFs. Hence, we transfer the calculated translational velocities of the gripped lattice nodes to the 6-DOF velocities of their corresponding grippers. Note that the grippers firmly hold the object and, thus, are coupled with the lattice due to the constraints in (1). We categorize the gripped nodes corresponding to each gripper in the sets $\mathcal{G}_{l}$ such that $\sum_{l=1}^{m_l} |\mathcal{G}_{l}| = n_g$ where $|.|$ signifies the number of nodes in the set. We can, then, write the following equation:
between \( \mathbf{v}_{p_l} \), the 6 \( \times \) 1 velocity vector of the gripper \( l \), and \( \mathbf{v}_{g_l} \), the velocity vectors of the nodes in \( \mathcal{G}_l \) stacked together:

\[
\mathbf{v}_{g_l} = \mathbf{G}_{gp_l} \mathbf{v}_{p_l} \tag{25}
\]

where \( \mathbf{G}_{gp_l} \) is the grasp matrix of the gripper \( l \) and can be written as

\[
\mathbf{G}_{gp_l} = \begin{pmatrix}
1 & 0 & 0 & r_{l_{1x}} & -r_{l_{2y}} \\
0 & 1 & 0 & -r_{l_{1y}} & 0 \\
0 & 0 & 1 & r_{l_{1y}} & -r_{l_{1x}} \\
\vdots & \vdots & \vdots & \vdots & \vdots
\end{pmatrix}, j \in \mathcal{G}_l \tag{26}
\]

where \( r_{l_{1x}}, r_{l_{1y}}, \) and \( r_{l_{2y}} \) represent the three components of the vector from the gripper \( l \) to the lattice node \( j \) at each instant. We extend (26) to write a general equation for all the grippers and their corresponding lattice nodes

\[
\mathbf{v}_g = \mathbf{G}_{gp} \mathbf{v}_p \tag{27}
\]

where \( \mathbf{v}_g \) is a 3\( n_g \times 1 \) vector containing the translational velocities of all the gripped nodes of the lattice stacked together, \( \mathbf{v}_p \) is a 6\( m \times 1 \) vector containing the velocities of all the grippers stacked together, and \( \mathbf{G}_{gp} \) is the total grasp matrix with the size of 3\( n_g \times 6 \)\( m \) assembled from all the grasp matrices \( \mathbf{G}_{gp_l} \)

\[
\mathbf{G}_{gp} = \begin{pmatrix}
\mathbf{G}_{gp_1} & 0 & \cdots & 0 \\
0 & \ddots & \vdots & \vdots \\
0 & \cdots & \ddots & \vdots \\
0 & \cdots & \cdots & \mathbf{G}_{gp_l}
\end{pmatrix}, l \in \mathcal{M}. \tag{28}
\]

We finally define the control law

\[
\mathbf{v}_p = -k_p \mathbf{J}_{sp}^\top \mathbf{e}_s \tag{29}
\]

where \( k_p \) is a positive gain, and \( \mathbf{J}_{sp} = \mathbf{J}_{sg} \mathbf{G}_{gp} \) is the Jacobian that relates the lattice servoed nodes to the grippers. One can use a diagonal gain matrix instead of \( k_p \) to weight differently translation and rotation velocities.

In [4], we carried out shape servoing with a Jacobian-based proportional control law. In contrast to this previous work, our new control law in (29) is based on an analytic formulation of the deformation Jacobian, is applicable on objects of all forms, and can be used for both full and partial shape servoing. This new control law allows an exponential decrease of the shape servoing error \( \mathbf{e}_s \) toward zero if the ARAP deformation Jacobian approximates the object’s true deformation Jacobian well. The practical performance of the control law is illustrated and discussed in detail in the experiments section.

VI. EXPERIMENTS

In this section, we validate the effectiveness of our approach through a diverse set of experiments covering various forms and materials of elastic deformable objects. The videos of these experiments and of experiments using only the tracking pipeline can be found on the project website. The objects of interest comprise a linear object, i.e., a cable, two thin-shell objects, including a sheet of A4 paper and a convoluted foam, and two volumetric objects, including a shoe sole and a foam octagonal cylinder. We apply large deformations and do both full and partial shape servoing. Full shape servoing is when we servo all the lattice nodes (except for the gripped nodes), i.e., \( n_f = 0 \). Partial shape servoing is, however, when we servo a portion of the lattice nodes, i.e., \( n_f \neq 0 \). The ability to perform partial shape servoing is interesting in practice (e.g., for tasks where a specific part of the object has to be assembled on another object), and it highlights the versatility of our approach.

A. Experimental Setup

The experiments are conducted using a dual-arm setup made of two Franka Emika Panda robots each with 7 DOFs. A Realsense D435 camera facing the manipulation area provides the input for the tracking pipeline. We use the camera resolution 424 \( \times \) 240 in all the experiments except for the ones with linear objects in which we double this resolution to apply image filters (see Section VI-B). We use MoveIt Hand-eye calibration plugin to externally calibrate the camera with the robots. The setup is shown in Fig. 1. The whole code is written in C++ and runs on a single robot operating system (ROS) node on a Dell laptop with a ninth generation Intel Core i7 CPU. No parallelization is employed to run the ROS node. Our approach calculates the grippers’ velocities and sends them to a Cartesian velocity control ROS node that controls each robot. We use the point cloud library (PCL) for handling point clouds.

Depending on the complexity of the objects’ geometry, we form the undeformed point cloud of each object \( \mathbf{p}^u \) either by scanning them (using a Kinect and Skanect software) or by drawing simple shapes in Blender. We, then, form the lattice \( \mathbf{s}^u \) around each object point cloud. In all cases, we consider the size of the lattice to be 1 cm larger than the size of the object in each direction. We rigidly transform the created \( \mathbf{p}^u \), as the initial shape \( \mathbf{p}^c \), somewhere in front of the camera where it is reachable by the robots. We, then, align the real object with the initial shape \( \mathbf{p}^c \) while being grasped by the robots. Note that a partial alignment is sufficient (see Section IV).

We, then, trigger the tracking pipeline. After the tracking pipeline successfully starts tracking the shape of the object, we activate the use of grippers’ 3-D coordinates in the tracking pipeline. We select the eight closest lattice nodes to each gripper as the lattice nodes with known coordinates to be used in the tracking pipeline. In this stage, we also select the gripped lattice nodes in the servoing pipeline. We use the same selected lattice nodes with the known coordinates in the tracking pipeline as the gripped lattice nodes. Next, we set the lattice desired shape \( \mathbf{s}^d \). To this end, we manually deform the object by moving the robotic arms while grasping the object. This is a natural way of defining the desired shape. One can also define the desired shape without the robots holding the object; our approach does not have any constraint in this respect. We store the lattice desired shape \( \mathbf{s}^d \) which is corresponding to the object’s desired shape \( \mathbf{p}^d \).

The next step is to manually move the robots to set the initial shapes of the lattice and the object. This is done in the same way as setting the desired shapes. Finally, we start the servoing pipeline to drive the lattice (and thus the object) from its initial
shape to the desired shape. As a common occurrence in research robots, the robots’ movement might be aborted by reflex errors. This is mainly due to sudden and nonsmooth movements. To avoid these reflexes, we gradually increase the gain (from zero to a final constant value) at the beginning of the servoing tasks. We also saturate the velocities (translational and rotational) sent to the robots.

In total, we define 13 tasks, each with specific features and challenges. Table I presents the main parameters of each task. We categorize the tasks based on the general form of the object under manipulation. The tasks’ results are presented in Figs. 5–10. In each figure, the elements corresponding to the current and desired shapes are visualized with green and red colors, respectively. Furthermore, the sections of the object or lattice belonging to the servoed regions are indicated in brighter colors while the ones belonging to the free regions (which are present in partial shape servoing) are indicated in darker colors. Finally, for each task, an RMSE graph (RMS of $e_x$) illustrates the servoing error during the task. We set the control gain $k_p$ as 0.1 in full shape servoing tasks and 0.05 in partial shape servoing tasks. We tune these gain values empirically. They allow us to obtain good performance while avoiding reflex errors that cause the robots to stop moving. In the following, we explain the tasks in more detail.

### B. Linear Objects

For the experiments with linear objects, we use an electric cable with the length of 70 cm. As for creating $p^w$, we form a cylinder with the same dimension as the cable. However, instead of considering the whole point cloud on the surface of this cylinder, we consider merely half of the points, i.e., a semi-cylinder. This is done to prevent the inferred object shape and the lattice from rotating around the longitudinal axis during tracking due to axial symmetry. We form a $15 \times 3 \times 3$ size lattice around $p^w$ in a way that the lattice direction with 15 nodes is in line with the length of the cable. The cable is manipulated from its two ends by the two robots. We define three tasks with this cable: two in-plane, and one out-of-plane. We put a board between the two robots on which we lay the cable. This board serves two purposes; first, the in-plane manipulations take place on the surface of this board, and second, we employ the difference in colors of the board and the cable to filter out unwanted captured point cloud coming from the board. The results of the tasks can be found in Figs. 5 and 6. In Fig. 5, the plotted points represent the mean coordinates of the lattice nodes belonging to the cross sections of the lattice along the cable length. The main feature

| Task | Object Type | Object | Template creation | Object point cloud size | Lattice Dimension | Full/partial shape servoing | Main feature |
|------|-------------|--------|-------------------|------------------------|------------------|---------------------------|-------------|
| T1.1 | Linear      | Cable  | Blender           | 1734                   | $15 \times 3 \times 3$ | f                         | In-plane deformation, large deformation |
| T1.2 | Thin-shell  | A4 Paper | Blender         | 1024                   | $8 \times 8 \times 3$ | f                         | In-plane deformation |
| T2.2 | Volumetric  | Shoe sole | Scan           | 502                    | $8 \times 4 \times 4$ | f                         | Large deformation |
| T2.3 |            | Convulated foam | Scan       | 5535                   | $8 \times 8 \times 3$ | f                         | Only translation, large deformation |
| T3.1 |            | Foam octagonal cylinder | Blender   | 2352                   | $8 \times 4 \times 4$ | f                         | Severe rotation, change of view |
| T3.2 |            |        |                  |                        |                  | f                         | Twist |
| T3.3 |            |        |                  |                        |                  | f                         | Twist + Bending deformation |
We then select the lattice nodes encapsulating this region as the servoed nodes of the lattice.

3) **T1.3**: This experiment aims to servo the cable through an out-of-plane deformation. To this end, we remove the constraints regarding keeping the grippers’ relative pose with respect to the board. This is done by setting the grippers higher in comparison to the board’s surface while defining the desired shape. This can be observed in Fig. 6. We also send the full translational and rotational velocities to the robots.

**C. Thin-Shell Objects**

The next experiments are conducted with thin-shell objects. Our objects of interest are a blank A4 paper and a convoluted foam. For both objects, we form an $8 \times 8 \times 3$ size lattice where the $8 \times 8$ side is aligned with the surface of the objects and the direction with three nodes is in line with the width of the objects. We start with the paper. We define four tasks, which are explained in the following. Fig. 7 presents the results of these tasks.

1) **T2.1**: In this task, we fully servo the paper toward a desired shape with large deformation.

2) **T2.2**: This task aims to do partial servoing with the paper. To this end, similarly to T1.2, we divide the paper into five sections along its longer dimension. We, then, select the lattice nodes encapsulating the second and the fourth sections of the paper as the servoed lattice nodes.

3) **T2.3**: This task is similar to T2.2. There are, however, two main differences: first, the servoed region is smaller, i.e., one fourth of the object, and second, we merely apply translation to the robots’ grippers without any rotation. The latter is done by updating only the translational velocities of the robots.

4) **T2.4**: In this task, we do partial shape servoing with two-fourth middle region of the paper as the servoed region. The important change in this task is that we significantly displace the grippers on the paper while setting the initial shape (each in the opposite corner of the paper) in comparison to the desired shape (both in the middle of the paper). We also update the grasped lattice nodes after setting the initial shape. This task is performed with a slightly thicker A4 paper so that the corners of the paper do not entirely loosen due to the large distance from the grippers.

The next series of experiments are with a convoluted foam, that is widely used in packaging industry. The goal is to demonstrate that the surface of the thin-shell object should not necessarily be flat. This is thanks to the generality that using the lattice brings to our approach as we can track and servo objects of any geometry. Another point is that as the convoluted foam
Fig. 8. Tasks with a thin-shell object: A convoluted foam. Top row: Task with full shape servoing. Bottom row: Task with partial shape servoing. Two small separated regions of the foam are servoed.

is thin and has a low stiffness, it does not follow the rotations of the grippers. We, thus, similarly to T2.3, update merely the translational velocities of the robots. We define two tasks with the convoluted foam, which are described in the following. Fig. 8 presents the results of these tasks.

1) T2.5: In this task, we conduct full shape servoing toward a desired shape with a large deformation with respect to the initial shape.

2) T2.6: This task aims to conduct a partial shape servoing with two small separated servoed regions of the convoluted foam. To this end, we initialize select the same servoed regions as in T2.2 and then inversely halve each region relative to the center-line of the convoluted foam. In contrast to the previous tasks, we define the servoed regions’ desired shapes in a way that makes the convoluted foam undeformed at the end of the task.

D. Volumetric Objects

The final set of experiments is carried out with two volumetric objects: a bulky shoe sole and a foam octagonal cylinder. For both objects, we form an $8 \times 4 \times 4$ size lattice where the direction with eight nodes is in line with the longer direction of the objects. We start with the shoe sole. We define two tasks with the shoe sole, which are explained in the following. Fig. 9 presents the results of these tasks.

1) T3.1: In this task, we fully servo the shoe sole toward a desired shape with a large deformation.

2) T3.2: This task is similar to T3.1. The main difference is that the desired shape is defined with a severe rotation with respect to the initial shape. This can be observed in Fig. 9. In fact, the shoe sole should be rotated and flipped during the servoing. Consequently, the part of the shoe sole that is visible in the desired shape is considerably different from the one in the initial shape.

As the last set of experiments, we define two tasks with the foam octagonal cylinder. These tasks include twisting the foam. Hence, in order to ensure that the foam follows the rotations of the grippers, we select the foam to be relatively dense. This applies an intense rotational force to the robots during the servoing. We, thus, considerably decrease the saturation values of the grippers’ velocities to avoid reflex errors in the robots. We also paint the foam lengthwise to better illustrate the applied twist. Fig. 10 presents the results of these tasks. The defined tasks are described in the following.

1) T3.3: In this task, the desired shape is set by applying a pure twist to the foam.

2) T3.4: In this task, we apply twist and bending deformation at the same time.

E. Results and Discussion

As shown through various experiments, our proposed unified tracking-servoing approach is able to track different forms of the objects and fully and partially servo them toward largely deformed desired shapes. The servoing error graphs in the rightmost side of the Figs. 5–10 verify the efficiency of our approach.
in different scenarios. The variety of objects’ materials employed in these experiments confirms the robustness of our proposed approach in dealing with many of the elastic deformable objects around us without having a knowledge of their mechanical parameters. Using a 3-D lattice makes it possible to use the same tracking and servoing approach for an object with any form. Furthermore, as shown, our approach can handle tasks with partial shape servoing with one or multiple small servoed regions. Defining servoed regions is quite straightforward and is done by just specifying the corresponding servoed lattice nodes encapsulating the object’s servoed regions. Another point that should be noted here is that our approach provides full control over the deformation of the object in 3-D space; i.e., the object can be simultaneously deformed, rotated, and translated. This can be particularly observed in T3.2 where the visible side of the shoe sole in its desired shape is totally different in comparison to the one in its initial shape. Applying twist and bending deformation in T3.3 and T3.4 is another manifestation of this full control over the object’s deformation in 3-D space. To the best of the authors’ knowledge, no approach in the literature possesses this feature. The servoing can be even performed in the existence of some noise in the tracking as can be seen in T3.4. When dealing with these cases, our proposed servoing pipeline drives the object toward a shape with a small residual error that corresponds with a global minimum with a nonzero value. This can be observed in T2.3, T2.5, and T2.6.

2) Rigid Motion and Time Horizon: Our servoing pipeline does not make a distinction between rigid and nonrigid components of the shape servoing error. This is the common approach in the state-of-the-art [5], [19], [49]. Our extensive experiments are carried out in scenarios close to those of interest in real-world industrial applications, and they involve simultaneous rigid and nonrigid object motions. As our experimental results demonstrate, our servoing pipeline performs with efficiency and accuracy in these scenarios. In cases with a very large rigid motion of the object between the initial and desired shapes, it may be interesting to distinguish rigid from nonrigid components of the shape servoing error. This could enable finer control over the robotic arm motions and the evolution of the object’s shape. In this respect, we see no hurdles to combining our approach with a specific module for rigid-body motion handling (which is a well-studied problem). In addition, we conducted preliminary simulations that suggest that our Jacobian estimation could be applicable for a certain time horizon in an MPC-style controller; especially, for moderate deformations. These aspects, which exceed the scope of our article, are interesting directions for future work.

3) Computational Cost: Regarding the execution speed of our approach, during the experiments, we reached 20–30 FPS for the whole process (tracking and servoing) without any parallelization using only CPU. This range is the same as in [4]. The main difference is that our approach handles both tracking and servoing at the same time while in [4], the execution speed is reported only for servoing, and tracking is performed by a
separate algorithm. One point that should be noted here is that in calculating the analytical Jacobian, the inversion of matrices in (22) would be costly when applied to a very large mesh. In our approach, however, this does not cause a problem as we calculate the deformation Jacobian for the lattice whose shape has a much smaller size (in terms of number of nodes) than the object.

4) Nonconvex Object Geometries: In this work, we only consider objects with convex geometry, and we form a uniform lattice over them. If the object has a nonconvex geometry, using a uniform lattice can result in unnecessary geometric constraints between different parts of the object that are not directly connected. For instance, in the case of a doll toy, such constraints may arise between the hands and legs, leading to unrealistic deformations where the movement of one hand affects the position of a leg. To overcome this challenge, a method proposed in [31] can be used, where unnecessary links in the lattice are disregarded. This can be achieved by creating a uniform lattice around the nonconvex object and then removing all tetrahedral cells and their corresponding lattice nodes that lie entirely outside the input geometry. We note that our approach can be applied to the modified lattice without any modifications.

5) Correspondences Between the Current and Desired Shapes of the Lattice: Our servoing pipeline is applied to the lattice, and requires correspondences between the servoed nodes of the current and desired shapes of the lattice. Next, we explain in more detail how the experimental process described in Section VI-A gives us these correspondences. As the first step of this process, we launch our tracking pipeline. Then, we manually set the desired shape. After that, we manually set the initial shape, i.e., the current shape at time zero, and finally, we launch the servoing pipeline. Note that the tracking pipeline is running continuously, i.e., tracking the object and lattice continuously, throughout all these steps. Hence, we get the correspondences between all the lattice nodes, including the set of servoed nodes, of the current and desired shapes. This type of approach is practical and we have used it previously [4]. One of its advantages is that it employs short-baseline registration between each two consecutive frames during tracking (see Section IV-C), instead of wide-baseline registration between the current and desired shapes, which can be more challenging.

6) Task-Specific Adjustments: To conclude, we discuss the implications of the adjustments we made in several experiments. We restricted the robots’ action space in several tasks: in T1.1 and T1.2, we wanted to mimic a planar manipulation scenario, so as to have a more diverse set of experiments; but this restriction is not needed, as shown by our full-action-space experiments with 3-D gripper motion. Similarly, in T2.3, T2.5, and T2.6, we did not use rotation of the grippers. We did this to illustrate the behavior in cases where this limitation exists, but this is not a limitation of our approach. We also chose the initial and desired shapes in our tests to illustrate a diversity of situations, including large deformations and cases of interest in industrial application (e.g., folding or unfolding). In addition, it is common and reasonable to select a lattice that represents the object’s deformation suitably (as discussed in Sections III and VI-E4), and that also facilitates tracking of the object (e.g., the lattice for the elastic cable in our experiments). Clearly, in practice, every particular scenario has its own particular features that need to be considered. However, neither this fact nor the adjustments described above detract from the generality of our approach.

F. Comparison With State-of-the-Art

In this section, we compare our shape servoing pipeline with our previously presented ARAP shape servoing (ARAP-SS) scheme [4] through an experiment. ARAP-SS considered thin-shell objects, and it did not provide unified tracking and servoing as we do here. We design two identical tasks for the two approaches with the thick A4 paper from T2.4. The initial and the desired shapes are considered the same for the two tasks as can be seen in Fig. 11. This is done by storing the robots’ configuration for both initial and desired shapes and keeping the grippers’ poses on the object unchanged in the two tasks.

ARAP-SS was designed for thin-shell objects, as it was based on a surface (not volumetric) deformation model. Thus, we use only the nodes and interconnections on the outer surface of the lattice as the template to ARAP-SS. We thus servo only these lattice nodes from the initial to the final shape. In order to have a fair comparison, we do a partial shape servoing with the same outer lattice nodes with our proposed servoing pipeline. We also use the same gripped nodes and control gains in both approaches. Similarly to the previous tasks, we saturate the translational and rotational velocities sent to the robots. In the right-hand side of Fig. 11, the shape servoing errors of the two approaches are compared. As seen, no significant difference can be observed between these graphs. This validates the precision of our servoing approach in comparison to the precision (which is state-of-the-art) of ARAP-SS. This precision, along with other features of our proposed approach, including the analytical expression for the Jacobian, having full control over the object’s deformation in 3-D space and scalability, privilege our approach with respect to other existing approaches.

G. Failed Cases

In this section, we present a failed case of our approach. Our failed case concerns driving the object through its singular shape, i.e., deforming it from an upward curved to a downward curved shape or vice versa. In general, this is a complicated task as it requires certain techniques that might differ from one
object to another. We show this difficulty by defining tasks with the thick A4 paper from T2.4. These tasks can be observed in Fig. 12. We start with a task in which the initial shape is upward curved, and the desired shape is downward curved. This is shown in the first row of Fig. 12. As seen, the shape servoing pipeline can successfully drive the object through the flat shape, i.e., the singular shape. It should be noted that, for this task, the direction of gravity is favorable throughout the servoing. In order to make the task more challenging, this time, we try to start from a downward curved shape and drive the object to an upward curved shape. This is shown in the second row of Fig. 12. As seen, despite the severe rotations applied to the paper by the robots, the paper cannot pass through the flat shape. This continues until the robots reach their rotational limits. We show that it is possible to solve this problem using a planning strategy. In particular, this can be done by firstly unwrapping the paper to a certain extent and then applying the required rotation to drive the paper through its singular shape. We applied this approach by defining two intermediary desired shapes for the paper: one nearly unwrapped downward curved shape and one nearly unwrapped upward curved shape. We switch from one desired shape to the next one when the shape servoing RMS error becomes smaller than 15 mm. The last two rows of Fig. 12 present this process. As seen, using this solution, the task can successfully be carried out.

VII. CONCLUSION

In this article, we present a general tracking-servoing approach that is capable of deforming elastic objects toward a desired shape. Our approach has full control over the deformation of elastic objects of any form (linear, thin-shell, and volumetric) and any geometry in 3-D space. Next, we mention several limitations of our approach. For objects with low stiffness (like a cloth), it might be required to incorporate gravity which is not considered in our approach. We cannot servo any arbitrarily complex deformation of an object, but only those deformations that can be captured by the lattice. Besides, reliably driving the object through a singular shape requires additional techniques. Furthermore, handling contact with the environment is not included in our approach. Finally, as we used our Jacobian with a Cartesian velocity controller, there is no constraint for keeping the robots in comfortable configurations throughout the task. This might lead to the failure of the task. A direction for future work is to employ additional constraints for alleviating the last limitation. Considering contact with static objects in the scene or incorporating a planning algorithm to define feasible intermediary shape targets in particularly challenging scenarios are two other interesting future directions. Furthermore, as we use a 3-D lattice for any object, it is possible to transfer deformation from one lattice to another and consequently from one object to another. This feature can be useful in motion transfer applications concerning elastic deformable objects. We also consider using the idea of employing a lattice for improving generalization in RL for elastic deformable objects of any shape.

REFERENCES

[1] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini, “Dual-arm robotic manipulation of flexible cables,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2018, pp. 479–484.
[2] M. Aranda, J. A. C. Ramon, Y. Mezouar, A. Bartoli, and E. Özgür, “Monocular visual shape tracking and servoing for isometrically deforming objects,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2020, pp. 7542–7549.
[3] A. Koessler, N. R. Filella, B. Bouzgarrou, L. Lequievre, and J.-A. C. Ramon, “An efficient approach to closed-loop shape control of deformable objects using finite element models,” in Proc. IEEE Int. Conf. Robot. Autom., 2021, pp. 1637–1643.
[4] M. Shetab-Bushehri, M. Aranda, Y. Mezouar, and E. Özgür, “As-rigid-as-possible shape servoing,” IEEE Robot. Autom. Lett., vol. 7, no. 2, pp. 3898–3905, Apr. 2022.

[5] D. Navarro-Alarcon and Y.-H. Liu, “Fourier-based shape servoing: A new feature-matching method to actively deform soft objects into desired 2-D image contours,” IEEE Trans. Robot., vol. 34, no. 1, pp. 272–279, Feb. 2018.

[6] F. Alambeigi, Z. Wang, R. Hegeman, Y. Liu, and M. Armand, “Autonomous data-driven manipulation of unknown anisotropic deformable tissues using unmodelled continuum manipulators,” IEEE Robot. Autom. Lett., vol. 4, no. 2, pp. 254–261, Apr. 2019.

[7] J. Shi, P. W. Ferguson, S. A. Pedram, J. Ma, E. P. Dutson, and J. Rossignac, “Autonomous manipulation via surgical robot using learning based model predictive control,” in Proc. Int. Conf. Robot. Autom., 2019, pp. 3875–3881.

[8] Z. Hu, P. Sun, and J. Pan, “Three-dimensional deformable object manipulation using fast online Gaussian process regression,” IEEE Robot. Autom. Lett., vol. 3, no. 2, pp. 979–986, Apr. 2018.

[9] Z. Hu, T. Han, P. Sun, J. Pan, and D. Manocha, “3-D deformable object manipulation using deep neural networks,” IEEE Robot. Autom. Lett., vol. 4, no. 4, pp. 4255–4261, Oct. 2019.

[10] R. Lagneau, A. Krupa, and M. Marchal, “Automatic shape control of deformable wires based on model-free visual servoing,” IEEE Robot. Autom. Lett., vol. 5, no. 4, pp. 5252–5259, Oct. 2020.

[11] J. Qi et al., “Composites-based manipulation of composite rigid-deformable objects with finite time model estimation and shape/position control,” IEEE/ASME Trans. Mechatron., vol. 27, no. 5, pp. 2985–2996, Oct. 2022.

[12] T. Bretl and Z. McCarthy, “Quasi-static manipulation of a Kirchhoff elastic rod based on a geometric analysis of equilibrium configurations,” Int. J. Robot. Res., vol. 33, no. 1, pp. 48–68, 2014.

[13] A. Sintov, S. Macenski, A. Borum, and T. Bretl, “Motion planning for dual-arm manipulation of elastic rods,” IEEE Robot. Autom. Lett., vol. 5, no. 4, pp. 6065–6072, Oct. 2020.

[14] N. Lv, J. Liu, and Y. Jia, “Dynamic modeling and control of deformable linear objects for single-arm and dual-arm robot manipulations,” IEEE Trans. Robot., vol. 39, no. 4, pp. 2341–2353, Aug. 2022.

[15] C. Wang et al., “Offline-online learning of deformation model for cable manipulation with graph neural networks,” IEEE Robot. Autom. Lett., vol. 7, no. 2, pp. 5544–5551, Apr. 2022.

[16] O. Aghajanazadeh, M. Aranda, J. A. C. Ramon, C. Cariou, R. Lenain, and Y. Mezouar, “Adaptive deformation control for elastic linear objects,” Front. Robot. AI, vol. 9, 2022, Art. no. 868459.

[17] O. Aghajanazadeh, M. Aranda, G. López-Nicolás, R. Lenain, and Y. Mezouar, “An offline geometric model for controlling the shape of elastic linear objects,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2022, pp. 2175–2181.

[18] O. Aghajanazadeh, G. Picard, J. A. C. Ramon, C. Cariou, R. Lenain, and Y. Mezouar, “Optimal deformation control framework for elastic linear objects,” in Proc. IEEE 19th Int. Conf. Autom. Sci., 2022, pp. 722–728.

[19] D. Berenson, “Manipulation of deformable objects without modeling and simulating deformation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2013, pp. 4525–4532.

[20] D. McNamara, A. Dobson, M. Ruan, and D. Berenson, “Manipulating deformable objects by interleaving prediction, planning, and control,” Int. J. Robot. Res., vol. 39, no. 8, pp. 957–982, 2020.

[21] J. Zhu, D. Navarro-Alarcon, R. Passama, and A. Cherubini, “Vision-based shape servoing: A new feature-matching method to actively deform soft objects into desired 2-D image contours,” IEEE Trans. Robot., vol. 34, no. 1, pp. 272–279, Feb. 2018.

[22] F. Ficuciello, A. Migliozzi, E. Coevoet, A. Petit, and C. Duriez, “FEM-based deformation control for dexterous manipulation of 3D soft objects,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2018, pp. 4007–4013.

[23] O. Sorkine and M. Alexa, “As-rigid-as-possible surface modeling,” in Proc. Eurographics Symposium on Geometry Processing, 2007, pp. 109–116.

[24] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, “Laplacian surface editing,” in Proc. Eurographics/ACM SIGGRAPH Sympos. Geometry Process., 2004, pp. 175–184.

[25] S. Collins, A. Bartoli, N. Bourdell, and M. Canis, “Robust, real-time, dense and deformable 3D organ tracking in Laparoscopic videos,” in Proc. Int. Conf. Med. Image Comput. Comput. Assist. Intervention, 2016, pp. 404–412.
[50] J. Sanchez, K. M. E. Dine, J. A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, “Blind manipulation of deformable objects based on force sensing and finite element modeling,” Front. Robot. AI, vol. 7, 2020, Art. no. 73.

[51] M. Müller, B. Heidelberger, M. Henrix, and J. Ratcliff, “Position based dynamics,” J. Vis. Commun. Image Representation, vol. 18, no. 2, pp. 109–118, 2007.

[52] V. H. Giraud, M. Padrin, M. Shetab-Bushehri, C. Bouzgarrou, Y. Mezouar, and E. Ozgur, “Optimal shape servoing with task-focused convergence constraints,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2022, pp. 2197–2202.

[53] M. Ruan, D. McConachie, and D. Berenson, “Accounting for directional rigidity and constraints in control for manipulation of deformable objects without physical simulation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2018, pp. 512–519.

[54] J. Matas, S. James, and A. J. Davison, “Sim-to-real reinforcement learning for deformable object manipulation,” in Proc. Conf. Robot Learn., 2018, pp. 734–743.

[55] R. Jangir, G. Alenya, and C. Torras, “Dynamic cloth manipulation with deep reinforcement learning,” in Proc. IEEE Int. Conf. Robot. Autom., 2020, pp. 4630–4636.

[56] R. Laezza and Y. Karayiannidis, “Learning shape control of elastoplastic deformable linear objects,” in Proc. IEEE Int. Conf. Robot. Autom., 2021, pp. 4438–4444.

[57] M. H. D. Zakaria, M. Aranda, L. Lequipèvre, S. Lengagne, J. A. C. Ramón, and Y. Mezouar, “Robotic control of the deformation of soft linear objects using deep reinforcement learning,” in Proc. IEEE 18th Int. Conf. Automat. Sci., 2022, pp. 1516–1522.

[58] M. Yu, H. Zhong, and X. Li, “Shape control of deformable linear objects with offline and online learning of local linear deformation models,” in Proc. Int. Conf. Robot. Automat., 2022, pp. 1337–1343.

Mohammadreza Shetab-Bushehri received the Ph.D. degree in electrical, electronic, and engineering systems from the Université de Clermont Auvergne, Clermont-Ferrand, France, in 2023. He is currently a Research Fellow with Ecole Centrale de Lyon, Ecully, France. His current research interests include the perception and manipulation of deformable objects and the development of multitask robotic agents.

Miguel Aranda (Member, IEEE) received the Ph.D. degree in systems engineering and computer science from Universidad de Zaragoza, Zaragoza, Spain, in 2015. He held postdoctoral research positions with Institut Pascal Laboratory, Clermont-Ferrand, France. He is currently a Research Fellow with Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza. His current research interests include multiagent control and robotic manipulation.

Youcef Mezouar (Member, IEEE) received the Ph.D. degree in automation and computer science from the University of Rennes, Rennes, France, in 2001, and the Habilitation degree in robotics from Université Blaise Pascal, Clermont-Ferrand, France, in 2009. He spent one year as a Postdoctoral Associate with Robotic Lab, Computer Science Department, Columbia University, NY, New York. From 2002 to 2011, he was Assistant Professor with Physics Department, Blaise Pascal University, Clermont-Ferrand. Since 2012, he has been a Full Professor with Clermont Auvergne INP-SIGMA Clermont, Clermont-Ferrand. His research interests include sensor-based control and computer vision.

Erol Özgür received the Ph.D. degree in robotics and vision from Université Blaise Pascal (UBP), Clermont-Ferrand, France, in 2012. From 2012 to 2015, he was a Postdoctoral Fellow with Institut Pascal-UBP/CNRS/IFMA, Clermont-Ferrand. From 2015 to 2018, he joined as an Assistant Professor with Université d’Auvergne, Clermont-Ferrand, where he is currently an Assistant Professor with Clermont Auvergne INP - Sigma Clermont (2018). His research interests include vision-based robot control and computer vision.