As-Rigid-As-Possible Shape Servoing

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

***Abstract*—Shape servoing is the problem of controlling the shape of a deformable object via robotic manipulation, exploiting sensory feedback. In many shape servoing scenarios of practical interest, the object has a natural tendency to preserve local rigidity. Motivated by this observation, we propose to use the As-Rigid-As-Possible deformation model for shape servoing. Specifically, we propose a novel control scheme based on a deformation Jacobian computed from this model. The proposed scheme drives the object to a desired 3D shape using as feedback the object's measured current 3D shape. The scheme is simple to implement, and it avoids some typical requirements in existing work: specifically, *(i)* we do not need to know the object's mechanical deformation parameters, and *(ii)* we do not use a Jacobian computed from data collected over a time window while the robots move. We test the proposed scheme in bi-arm shape servoing experiments with a variety of deformable objects of different material (paper, rubber, plastic). Tracking of the deformable object's surface in 3D is performed in our experiments via an algorithm based on monocular vision. The experimental results validate the practicality of our scheme.**

***Index Terms*—Visual Servoing, Perception for Grasping and Manipulation, Sensor-based Control.**

Fig. 1. View of our experimental setup during testing of bi-arm visual shape servoing of a rubber tire tread with an overhead monocular camera.

I. Introduction

DEFORMING an object into a desired shape by means of robotic manipulation is an important task in diverse domains. Examples include manufacturing [1]–[3], surgical [4]–[6] and household [7], [8] applications. The problem of carrying out this task using sensory feedback has been given the name *shape servoing* [4]. In many practical instances of shape servoing, the manipulated object deforms elastically (as opposed to plastically). This is the case, e.g., with products such as elastic rods, tubes, cables, sheet metal, rubber layers, sponges, shoe soles, etc. In such instances, during the servoing task the manipulated object has a natural tendency to resist deformation and preserve rigidity locally.

In this paper, we propose a novel shape servoing scheme based on a geometrical representation of that tendency. Specifically, the representation we use is the well-known As-Rigid-As-Possible (ARAP) model [9]. We compute numerically a deformation Jacobian for the object by simulating this model, and propose a robotic control law based on this Jacobian. Using the perception of the object's current 3D shape, the scheme drives the object to a desired 3D shape.

This work was supported by project SOFTMANBOT, which received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 869855.

The authors are with CNRS, Clermont Auvergne INP, Institut Pascal, Université Clermont Auvergne, Clermont-Ferrand, France. *Corresponding author: M. Shetab-Bushehri, email:* m.r.shetab@gmail.com

Digital Object Identifier (DOI): 10.1109/LRA.2022.3145960.

Contribution. Compared with some existing literature, we do not use any mechanical model of the object's deformation. Instead, we just need to define a geometric mesh that represents the object's surface during the task with sufficient accuracy. Moreover, other existing methods use deformation Jacobians estimated from sensor data collected over a time interval. This data is collected while the robots move, and the estimation requires initialization. Our Jacobian, in contrast, is computed using only the measured shape of the object at the current instant. Therefore, our Jacobian is simpler to compute and to apply, and does not need initialization.

The main advantages of our scheme are its simplicity and generality. We show in experiments (the setup is illustrated in figure 1) that the scheme can be applied on objects with different properties (paper, rubber, plastic) without needing any specific mechanical modeling. The scheme controls accurately the complete shape of these objects in 3D space.

II. Related Work

A. Shape servoing with mechanical deformation models

Mechanical models are valuable tools for robotic manipulation of deformable objects [10]. In particular, the knowledge of the object's deformation dynamics is useful to reduce the difficulty of a shape servoing task. The Finite Element Method (FEM) for modeling elastic objects has been the most popular choice in the literature. In [11] and [12] this model was used to define control actions performed in an open

loop fashion. The work [3] addressed this issue exploiting a reduced, computationally efficient FEM model defined from a partition of the nodes in the model's mesh. This was applied to perform vision-based closed-loop shape servoing of a Deformable Linear Object (DLO). In [13], FEM was used to control a pose on the body of a deformable object, using force (not vision) as sensory feedback. Instead of FEM, the Reproducing Kernel Particle Method was used in [14] for a shape servoing task. Unlike the works that exploit mechanical physics-based deformation models, our scheme does not use mechanical deformation parameters of the object, and is thus simpler and less object-specific. It achieves shape servoing via geometrical modeling and sensing, not requiring explicit modeling or sensing of force.

B. Shape servoing with model-free estimation of sensor-based deformation Jacobian

Most works in the recent literature on shape servoing are based on the use of a deformation Jacobian estimated from sensor measurements. This is a technically sound approach to the problem and it has important advantages: for instance, these works do not need to know any deformation model of the object, and they have low computational cost. Examples include [4], [7], [15]–[17]. Some of these works are tailored to a specific type of objects, e.g., DLOs in [1], [18], or bodily tissue in surgical applications in [5], [6]. Several works have used machine learning to obtain the deformation Jacobian [6], [8]. Generally all these approaches require estimating, over a time window and under the robot motions, the variation of certain shape features defined from sensor measurements. Such estimation schemes are known to be sensitive to noise. Moreover, degeneracy may occur if some Degrees Of Freedom (DOF) are not excited during the robot motions, and one needs to initialize the estimator by means of certain motions to be executed prior to the control task. Our approach, in contrast, uses only the current shape to compute the control law, and it needs no initialization. In addition, most of these methods control a reduced representation of the object's shape, often expressed in the sensor space (e.g., contours). In some cases they do not actuate all the 6 DOF of each robot end-effector pose. By comparison, we control a mesh that represents the complete shape of the object. We control it in 3D and with 6 DOF per end-effector.

C. Shape servoing with geometric heuristics

Authors have recognized the interest of performing shape servoing based simply on geometric criteria. An example is [19], where the control scheme relied on an approximated deformation Jacobian obtained from a heuristic (diminishing rigidity). In contrast, our deformation Jacobian is obtained from a complete deformation model (ARAP); therefore, it can represent with more precision and richness the deformation behavior. Our scheme performs stably and with low computational cost, thus overcoming the main issues that prompted [19] to avoid the use of a deformation model in the Jacobian. More recently, the shape servoing controller of [19] was used in [20] within an interleaved planning-control approach. The work [2] addressed shape servoing of isometrically deforming objects. It did not use a deformation Jacobian; instead, the control strategy was based on a heuristic shape interpolation. This interpolation employed a geometric deformation model, Position-Based Dynamics (PBD) [21]. In contrast, here we do use a deformation Jacobian, and our control law is not heuristic, but based on a formalized deformation model. Moreover, our scheme is more general because ARAP can handle not only isometric deformations, but also stretching. We also validate the scheme with more diverse shape servoing experiments than [2], [19], [20] did.

D. ARAP

The core idea of ARAP is to model geometrically the tendency of an object to preserve local rigidity. The model is based on an energy measure that expresses the deviation from rigidity as the sum of deviations in local regions of the object. Stable shapes of the object correspond to local minima of that measure. The ARAP concept has proven powerful and popular in diverse applications. It has seen widespread use in computer graphics for shape interpolation, editing and animation [9], [22], [23]. ARAP has also been employed as deformation constraint in Shape-from-Template [24] and as regularization prior during tracking of nonrigid scenes [25], [26]. Recently, [8], [27] have used ARAP for regularization and inference of occluded regions during deformable object tracking under robotic manipulation. To our knowledge, our paper is the first to use ARAP for shape servoing. A main contribution of our work is thus to show that ARAP can be employed to guide shape servoing tasks successfully with a variety of diverse objects.

III. Problem formulation

The shape servoing problem we address consists in manipulating a deformable object in a feedback control loop to drive it to a desired state. A flowchart of our proposed solution can be found in figure 2. We consider a set of robots $\mathcal{M} = \{1, ..., m\}$ whose end-effectors grasp the object. A sensor (camera) viewing the object is used to provide the feedback. We assume the relative poses of the camera frame and robots' frames are all known, as well as the robots' kinematic models. We want to design a control scheme that will compute a 6-DOF velocity input associated with every end-effector frame: $\mathbf{v}_i = [\upsilon_i^T, \omega_i^T]^T$, where we stack the velocities for all in a column vector of length $6m$: $\mathbf{v} = [\mathbf{v}_1^T, ..., \mathbf{v}_m^T]^T$.

We define the shape of the object as the shape of its surface. We use a triangular surface mesh to represent this shape. The resolution of the mesh is chosen such that it represents with sufficient accuracy the geometry of the surface both in its initial and desired states. We call the set of mesh nodes $\mathcal{N} = \{1, ..., n\}$. Henceforth, by *shape* we refer to the 3D positions of the nodes of the chosen mesh. These positions are expressed in the camera frame. We define different shapes to be used in the control scheme:

- Current shape, $\mathbf{s}$.
- Desired shape, $\mathbf{s}^*$, which defines the control goal.
- Undeformed shape, $\mathbf{s}^u$, used in the ARAP model.

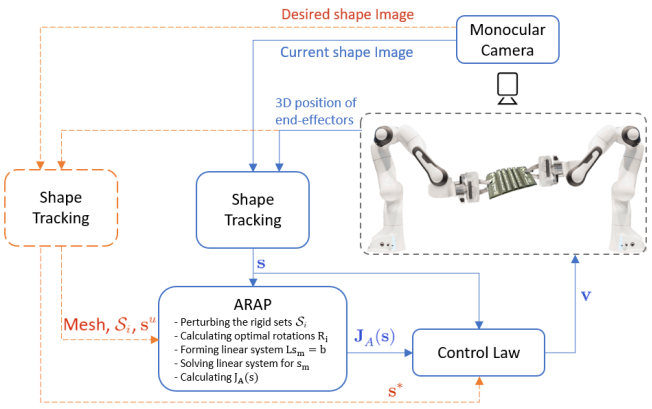

Fig. 2. Flowchart of the proposed shape servoing scheme. Online components in solid blue lines, offline components in dashed orange lines.

- Initial shape, $\mathbf{s}^0 = \mathbf{s}(t=0)$.

Each of them has size $3n \times 1$ and is a stacking of the node positions. More precisely, the current position of node i is named $\mathbf{s}_i$, and we define $\mathbf{s} = [\mathbf{s}_1^T, ..., \mathbf{s}_n^T]^T$ (analogously for the other shapes). $\mathbf{s}^*$ and $\mathbf{s}^u$ are assumed to be known before starting the servoing task. The measurement of $\mathbf{s}$ obtained from the camera at every time instant is available to the control scheme. We formulate the shape servoing task in terms of the following error:

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^*. \tag{1}$$

Driving this error to zero means the task has been completed.

A. Assumptions on object and robots

We assume the following regarding the object's behavior and its interaction with the robots:

- The robot controller can set exactly the 6-DOF velocity at which the end-effectors move at every instant. The grippers grasp the object firmly throughout the task. The object's shape stays statically stable (i.e., in quasi-static equilibrium), and reacts smoothly to the robot motions; i.e., infinitesimally, the change of shape under end-effector displacements can be modeled by a deformation Jacobian $\mathbf{J}_O(\mathbf{s})$ [1], [4], [19], [28]. Therefore, the following relation holds:

$$\dot{\mathbf{s}} = \mathbf{J}_O(\mathbf{s})\mathbf{v}. \tag{2}$$

- The object has a tendency to resist deformation and to maintain local rigidity, to the extent allowed by the external forces acting on it. This corresponds to an elastic (not plastic) behavior.
- The desired shape is a shape that the object can take (i.e., it is *feasible*). It is also *reachable*, *(i)* with the available actuation (i.e., for the number of grippers used and for their specific placement on the object), and *(ii)* by a monotonic decrease of the shape error $\mathbf{e}$, starting from the initial shape.

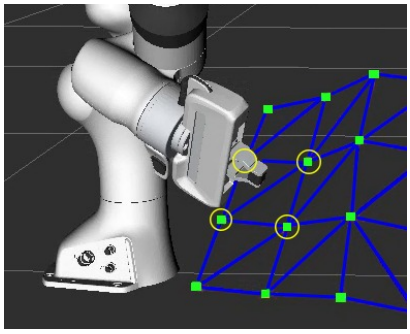

Fig. 3. Definition of a rigid set coupled with the end-effector. The four mesh nodes forming the rigid set are encircled in yellow.

IV. Deformation Jacobian Computation

Our idea is to compute, using the ARAP model, an approximation of the unknown Jacobian at the current shape $\mathbf{J}_O(\mathbf{s})$. Specifically, we compute numerically a Jacobian $\mathbf{J}_A(\mathbf{s})$ by simulating the ARAP model. The procedure is described next.

A. Finite-difference estimation

To compute the Jacobian, we need to link the robot end-effector frame with the ARAP mesh. To do this, for each robot i we define a *rigid set* $\mathcal{S}_i \subset \mathcal{N}$, with $|\mathcal{S}_i| = n_{\mathcal{S}_i}$ such that the nodes contained in $\mathcal{S}_i$ are rigidly coupled with i's end-effector. A simple way, which we used in our tests, to define $\mathcal{S}_i$ is to choose it as the set of nearest neighbors: i.e., the four nodes that surround the end-effector i (see an illustration in figure 3). This set can be identified from the knowledge of $\mathbf{s}^0$ and the end-effector position in 3D. As $\mathcal{S}_i$ contains at least three nodes whose positions are not aligned, we can associate a Cartesian frame, $\mathcal{F}_i$, to it. To do so, let us define the positions of the nodes in $\mathcal{S}_i$ as $\mathbf{s}_{\mathcal{S}_i} \in \mathbb{R}^{3n_{\mathcal{S}_i}}$. Then, we establish a mapping $f_i : \mathbb{R}^{3n_{\mathcal{S}_i}} \rightarrow SE(3)$. This mapping gives the frame's pose $\xi_i = f_i(\mathbf{s}_{\mathcal{S}_i})$. Conversely, the positions of the nodes for the frame's pose are obtained as: $\mathbf{s}_{\mathcal{S}_i} = f_i^{-1}(\xi_i)$.

We parameterize the motion of $\mathcal{F}_i$ by 6 DOF (3 translational, 3 rotational). Starting from the current shape $\mathbf{s}$, we simulate a perturbation of one DOF. This is the standard first step in finite-difference estimation. It gives us a new ξ_i. We compute the positions of the nodes $\mathbf{s}_{\mathcal{S}_i} = f_i^{-1}(\xi_i)$, and run the simulation of ARAP (see Sect. IV-B). This gives the new stable shape of the object after the perturbation. Using this new stable shape we apply standard forward finite differences to estimate the column of $\mathbf{J}_A(\mathbf{s})$ corresponding to the perturbed DOF using equation 2. Repeating this procedure for all DOF (6 for every $\mathcal{S}_i$) gives the full matrix.

In experiments, we also test a reduced, translation-only, Jacobian. This is computed by considering a single mesh node (i.e., a rigid set having only one element), and the 3 translational DOF only. This has the advantage of greater simplicity and is appropriate when the object's region moving rigidly with the end-effector is small (e.g., a small contact area on a slack object). Sect. VII includes more details.

B. ARAP simulation

We implemented ARAP following the algorithm of Sorkine and Alexa [9]. This algorithm is well-known and we refer to [9] for the details missing in the description we provide next. We will highlight the specific aspects where we tailored the algorithm to our application. ARAP relies on an energy E that expresses the deviation from rigidity. A *cell* is defined for each node in the mesh, comprising its first-order neighbors. We call this neighborhood $\mathcal{N}_i$ for cell i. Then the global energy is formulated as the sum of energies over all cells, i.e., $E = \sum_{i \in \mathcal{N}} E_i$, where:

$$E_i = \sum_{j \in \mathcal{N}_i} w_{ij} ||\mathbf{s}_i - \mathbf{s}_j - \mathbf{R}_i(\mathbf{s}_i^u - \mathbf{s}_j^u)||^2. \tag{3}$$

Each w_{ij} is a scalar encoding the connection between nodes i and j in the mesh. E_i expresses the deviation from rigidity (i.e., from the undeformed shape) of the cell i. This is because $\mathbf{R}_i$ is computed as the optimal rotation that minimizes E_i for given $\mathbf{s}^u$, $\mathbf{s}$. Therefore, E_i measures only the non-rigid component of the difference between the two cell shapes. ARAP works by computing a shape for the full object that corresponds to a minimum of E. This is explained next.

1) Optimal rotations: ARAP requires computing the optimal rotation $\mathbf{R}_i$ for every cell $i \in \mathcal{N}$. For this we define $\mathbf{e}_{ij} = \mathbf{s}_i - \mathbf{s}_j$, $\mathbf{e}_{ij}^u = \mathbf{s}_i^u - \mathbf{s}_j^u$, and the covariance matrix:

$$\mathbf{S}_i = \sum_{j \in \mathcal{N}_i} w_{ij} \mathbf{e}_{ij}^u \mathbf{e}_{ij}^T = \mathbf{P}_i^u \mathbf{D}_i \mathbf{P}_i^T, \tag{4}$$

where $\mathbf{P}_i^u$ and $\mathbf{P}_i$ contain respectively $\mathbf{e}_{ij}^u$ and $\mathbf{e}_{ij}$, arranged in columns for $j \in \mathcal{N}_i$, and $\mathbf{D}_i$ is a diagonal matrix that contains the w_{ij} for $j \in \mathcal{N}_i$. Then one computes the Singular Value Decomposition (SVD) of the matrix, i.e., $\mathbf{S}_i = \mathbf{U}_i \mathbf{\Sigma}_i \mathbf{V}_i^T$. From this, the optimal rotation is obtained:

$$\mathbf{R}_i = \mathbf{V}_i \ \mathrm{diag}(1, 1, \det(\mathbf{V}_i \mathbf{U}_i^T)) \ \mathbf{U}_i^T. \tag{5}$$

Due to the requirement of computing an SVD for each $\mathbf{R}_i$, this is the most computationally expensive part of our implementation. We avoid some unnecessary calculations by realizing that after perturbing a given DOF of a given rigid set (Sect. IV-A), $\mathbf{R}_i$ changes only for the cells i that contain a node in the rigid set that has been perturbed. Therefore, for the remaining cells, $\mathbf{R}_i$ only needs to be computed once, for the current shape $\mathbf{s}$ before perturbation.

2) Linear system solution: ARAP divides the nodes into two groups: *handled* and *free*. The handled nodes are those whose position is fixed directly by external constraints. The main idea of ARAP is that the object will remain as rigid as possible under these externally fixed positions. Concretely, the stable shape of the object in quasi-static equilibrium after the motion will be at a local minimum of the energy E. For this, one enforces the gradient $\partial E / \partial \mathbf{s}_i$ for every free node i to be zero. This results in this equation for a free node i:

$$\sum_{j \in \mathcal{N}_i} w_{ij}(\mathbf{s}_i - \mathbf{s}_j) = \sum_{j \in \mathcal{N}_i} \frac{w_{ij}}{2}(\mathbf{R}_i + \mathbf{R}_j)(\mathbf{s}_i^u - \mathbf{s}_j^u). \tag{6}$$

On the other hand, the positions of the handled nodes are defined as fixed. In our case, the handled nodes are the nodes belonging to the rigid sets. We fix their positions as those computed after perturbation (see Sect. IV-A), which we call $\mathbf{c}_i$ of size $n_{\mathcal{S}_i} \times 3$ for rigid set i. Considering the equations for all nodes gives a linear system expression: $\mathbf{L}\mathbf{s}_m = \mathbf{b}$, where $\mathbf{L}$ is an $n \times n$ size Laplacian matrix, $\mathbf{s}_m$ is the shape to be computed (i.e., the new stable shape after perturbation), expressed in $n \times 3$ format, and $\mathbf{b}$ contains the right-hand side of (6). The handled node constraints are included as $rows_{\mathcal{S}_i}(\mathbf{s}_m) = \mathbf{c}_i$ for every rigid set i. The linear system expression consists of an individual system for each of the three coordinates. Since our meshes are relatively small (tens of nodes), solving this linear system does not have a relevant effect on the overall computation time. We employ a standard least-squares solution method via the pseudoinverse of the system matrix.

ARAP uses an alternating minimization strategy where the shape that locally minimizes E is found by iterating the described procedure: i.e., the optimal rotations (Sect. IV-B1) are recomputed with the new $\mathbf{s}$ and the linear system is solved again. In [9], 2-3 iterations of the alternating minimization are used for a scenario where a user interactively moves the handled nodes in a visualization application. In our case, the motions of the handled nodes are small, so as to estimate the Jacobian. For this reason, we use a single iteration. That is to say, we run each of the two steps (computation of optimal rotations and solution of linear system) once. This provides suitable results in practice and a fast computation time.

C. Discussion

An interesting fact to highlight is that we compute the Jacobian using only the current shape $\mathbf{s}$, with no need for dynamic estimators. Moreover, the ARAP model is perfectly suited to the quasi-static scenario we consider because this model computes the new stable shape directly without having to simulate, and keep track of, velocities of the mesh nodes. The Jacobian computation is fast enough for real-time shape servoing: our non-optimized implementation runs at 20 Hz or more, for two robots (12 DOFs), with meshes ranging from 24 to 70 nodes. More details are given in Sect. VII.

V. Control law

We propose the following proportional control law, based on the ARAP Jacobian computed as explained in Sect. IV:

$$\mathbf{v} = -\lambda \, \mathbf{J}_A^+(\mathbf{s}) \, \mathbf{e}, \tag{7}$$

where λ is a positive scalar gain and $\mathbf{J}_A^+(\mathbf{s})$ is the pseudoinverse of the ARAP Jacobian. Note that $\mathbf{J}_A(\mathbf{s})$ was computed with respect to motions of $\mathcal{F}_i \ \forall i \in \mathcal{M}$, which are Cartesian frames defined to represent the object's rigid sets. Therefore $\mathbf{v}$ are velocities to be applied to these rigid set frames. We transform for each robot i the velocity of the rigid set frame to the velocity of the end-effector frame rigidly coupled with it (Sect. IV-A). Note that for every robot i the poses of these two frames are known at each instant. The resulting end-effector Cartesian velocities are sent to the robots. We assume the rigid sets are fixed (i.e., a robot is coupled with the same region of the object throughout the task). However, note that we do not

assume a specific fixed geometry of the coupling between end-effector and object. Therefore, an important fact to highlight is that our control scheme is robust to minor changes, such as moderate slippage, of the grasping conditions during servoing.

A. Stability discussion

The control law (7) is a classical one in visual servoing. Our control scheme is underactuated. For typical mesh sizes the number of features to be controlled ($3n$) clearly exceeds the number of actuated degrees of freedom ($6m$). Therefore, global stability (i.e., convergence to the desired shape from any initial condition) cannot be guaranteed when using (7). As discussed in, e.g., [29], with this control law the system is stable locally around the equilibrium $\mathbf{s}^*$ if $\mathbf{J}_A^+(\mathbf{s})$ and $\mathbf{J}_O(\mathbf{s})$ have full rank and the product $\mathbf{J}_A^+(\mathbf{s})\,\mathbf{J}_O(\mathbf{s})$ is a positive definite matrix.

Two important facts to note are: *(i)* $\mathbf{J}_A^+(\mathbf{s})\,\mathbf{J}_O(\mathbf{s})$ can be positive definite as long as $\mathbf{J}_A(\mathbf{s})$ approximates $\mathbf{J}_O(\mathbf{s})$ not too coarsely [29]. Therefore, precise knowledge of the Jacobian $\mathbf{J}_O(\mathbf{s})$ is not needed. *(ii)* We are just approximating the object's instantaneous reaction to forces, not its behavior over a long time horizon. Doing the latter would be challenging without knowledge of a deformation model for the specific object being manipulated. Considering these two facts and under the assumptions made (Sect. III-A), it is reasonable to use ARAP to represent, via $\mathbf{J}_A(\mathbf{s})$, the object's instantaneous tendency to preserve rigidity when subjected to forces. ARAP's principle is to approximate geometrically the behavior of real physical objects, and its ability to do so has been extensively validated in diverse applications. We verify the stability and suitability of the scheme in our experiments.

VI. Shape Tracking

To measure $\mathbf{s}$, we employ a shape tracking pipeline that takes as starting point the method proposed in [2]. The pipeline employs monocular vision and is based on Shape-from-Template (SfT) [30]–[32]. A monocular camera provides precise performance while being small, light-weight, and low-cost. This motivates using it for 3D shape tracking instead of an RGB-D sensor [27], [33], [34]. In this paper, our focus is on shape servoing; therefore, we will only provide brief details about the shape tracking pipeline. The pipeline is illustrated in figure 4. It needs the knowledge of a template, which is a reference model of the object. The information in the template includes a mesh, a deformation law (PBD is used for this) and the appearance (via a texture map) of the object. We use for shape tracking the same mesh as in the ARAP model.

The pipeline needs the object's surface to have visual texture, for matching purposes. In addition, it assumes that the object's surface deforms isometrically. The known 3D positions of the end-effectors, which are coupled with the object, are used as constraints in the inference of $\mathbf{s}$. This makes the inference more precise and robust. The process of applying these constraints is to firstly identify in the camera image the closest mesh node to each end-effector. Then, during tracking, the 3D position of that node can be defined as equal to the known 3D position of that end-effector in the camera

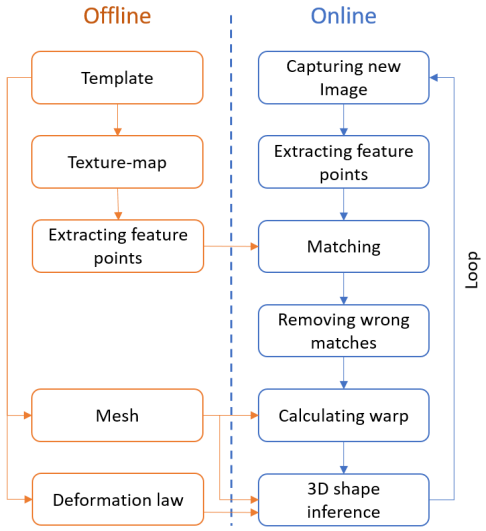

Fig. 4. Diagram showing the modules of the shape tracking pipeline.

frame. However, due to the possible errors in robot-camera relative calibration, we use a *soft constraint* instead. In this soft constraint, for each end-effector, we consider a sphere with a small radius centered at the 3D position of the end-effector. The constraint is, then, that if the corresponding mesh node is outside this sphere, it will be absorbed to the closest point on the surface of the sphere.

The tracking algorithm we use here provides remarkable improvements with respect to the one used in [2]. For example, it can work with natural textures, and it is faster (up to 30 FPS) despite being methodologically more complex.

VII. Experiments

In this section, we present the experimental results. A **video** of our experiments is attached and can also be found at `https://youtu.be/1w2tbgjLrUs`.

A. Experimental setup

We validate the effectiveness of our proposed scheme through different tasks conducted with two Franka Emika robot arms each with 7 DOF. The input for the shape tracking pipeline is provided by a calibrated Logitech C270 webcam installed on top of the two robots facing downward. We also externally calibrate the camera with the two robots using Moveit Hand-eye calibration plugin. The entire setup is shown in figure 1. Both the shape tracking and shape servoing codes were written in C++ and ran on a Dell laptop with an Intel Core i7 CPU, an NVIDIA Quadro T1000 GPU and 16GB RAM. The shape tracking and shape servoing are implemented as two separate ROS nodes that communicate in parallel with each other. We also implement the control of the robots in ROS using Cartesian velocity control.

B. Shape servoing tasks

We report the results of 7 shape servoing tasks involving 4 distinct objects: a Spiderman poster, a place mat, a piece of tire tread and a shoe sole. The general information of

TABLE I
PARAMETERS OF THE SHAPE SERVOING TASKS.

Task	1	2	3	4	5	6	7
Mesh Type (Regular/Irregular)	R	R	R	R	R	R	I
Mesh Nodes	7x10	4x6	6x8	6x8	6x8	6x8	40
Control Computation (Hz)	20	30	24	24	24	24	26

each task including the tested object, initial, desired, and final shapes of the object, several snapshots of the evolution of the task, and finally the error (RMS of $\mathbf{e}$) graph are shown in figure 5. The process of each task starts by firstly setting the desired shape of the object. This is done by manually moving the two arms while grasping the object. We infer and store this desired shape using our shape tracking pipeline. This desired shape is indicated as a red mesh in the RViz visualization shown in figure 5. Afterwards, we manually move the two robot arms to set the deforming object in its initial shape. We then start the task. The current shape $\mathbf{s}$ is inferred by our shape tracking pipeline at each time step. This current shape is indicated as a blue mesh with green nodes in the RViz representation. In order to avoid reflexes (which abort the motion) in the robots caused by sudden and non-smooth movements, we increase the gain progressively at the beginning of the robots' movement. Moreover, we saturate the translational and rotational velocities sent to the robots.

To validate the generality of the proposed scheme, the defined tasks cover diverse materials with different stiffness, various deformations, and several mesh sizes. We also evaluate the behavior under effects such as uncertainty of the grasping and unstable shape tracking conditions, as described below. The main parameters of each task are tabulated in table I. In the following, we explain the tasks in more detail.

1) Task 1: This task aims to deform a Spiderman poster printed on an A4 paper by just applying translation to the robot end-effectors without any rotation (see Sect. IV-A). To this end, in each side of the paper, we select only one mesh node as rigid set. As a result, ARAP shape servoing controller merely updates the translational velocity of the robots. In order to achieve a reasonably low error at the end of task, we tried to keep the rotational pose of the end-effectors unchanged during setting both the desired and initial shapes of the Spiderman poster.

2) Task 2: This task is conducted with a place mat that is made out of plastic which is stiffer than paper. In this task, in contrast to Task 1, we start from a highly deformed initial shape to a slightly deformed desired shape. The desired shape is both translated and rotated in different directions with respect to the initial shape.

3) Task 3: This task is carried out with the place mat but with a concave desired shape and a denser mesh. This illustrates that the servoing scheme is not tied to a specific mesh for a given object. One can use different meshes as long as they capture the object's geometry precisely enough.

4) Task 4: This task is also performed with the place mat with almost the same initial and desired shapes as Task 2. The major change that we made is displacing end-effectors at the initial shape with respect to their locations when the desired shape was formed. Specifically, we displaced each end-effector to its neighbor rigid set on the place mat in inverse directions with respect to each other. We then run the control scheme and try to reach the desired shape using these two new rigid sets.

5) Task 5: The object used in this task is a piece of tire tread from a heavy vehicle. In order to make it recognizable by our shape tracking pipeline, we added some texture on its surface by writing on it using a white marker.

6) Task 6: This task is also performed with the tire tread but with a convex desired shape.

7) Task 7: The shoe sole made of rubber used in this task is the stiffest among our deforming objects. We use an irregular triangular mesh to represent this object in both shape tracking and shape servoing algorithms. In order to make the shoe sole detectable by our shape tracking pipeline, we highlighted the available grooves on it and also drew some new lines on its surface using a black pen.

C. Results and Discussion

The proposed ARAP shape servoing scheme is able to accurately accomplish all the tasks covering different materials, desired shapes, and mesh sizes. As shown in the RViz representations in figure 5, in each task, the controller moves the robots gradually to conform the shape of the deforming object to the desired shape. The shape RMS error for each task is illustrated in the last column of the figure 5. As can be observed, some noise is present in a few of the shape RMS error graphs. This noise is introduced by the shape tracking pipeline. One could apply a temporal averaging to the positions of the object mesh nodes between consecutive frames in the shape tracking pipeline. This could diminish the noise and provide a more stable 3D inferred shape. However, our goal here is to show that our ARAP shape servoing scheme is robust against these noises and converges successfully in their presence.

Another point that should be noted is that the control scheme can be robust to changes of the grasping configuration. To clarify this, we illustrate in figure 6 the pose of the end-effectors in the desired and final shapes of Task 4 and Task 5 with green and red dashed lines, respectively. In Task 4, the relative poses of the end-effectors and their corresponding rigid sets are changed manually before starting the task. In Task 5, in turn, the right end-effector rotated due to the slippage during the task while remaining in the same rigid set. Despite these changes in the relative pose of the end-effectors with respect to their grasping points, the shape servoing scheme managed to accomplish both tasks thanks to guiding the rigid sets under a closed control loop. As long as the end-effectors remain coupled with the guiding rigid sets, the deforming object can converge to the desired shape.

We note that if the desired shape is unreachable or infeasible, the scheme converges to a shape corresponding to a local minimum of the error. Finally, we compare the prediction capability of our ARAP deformation Jacobian and the diminishing rigidity (DR) Jacobian used in [19], [20]. For this we use some of the less noisy data from our servoing tasks. As comparison metric we use the cosine similarity between the

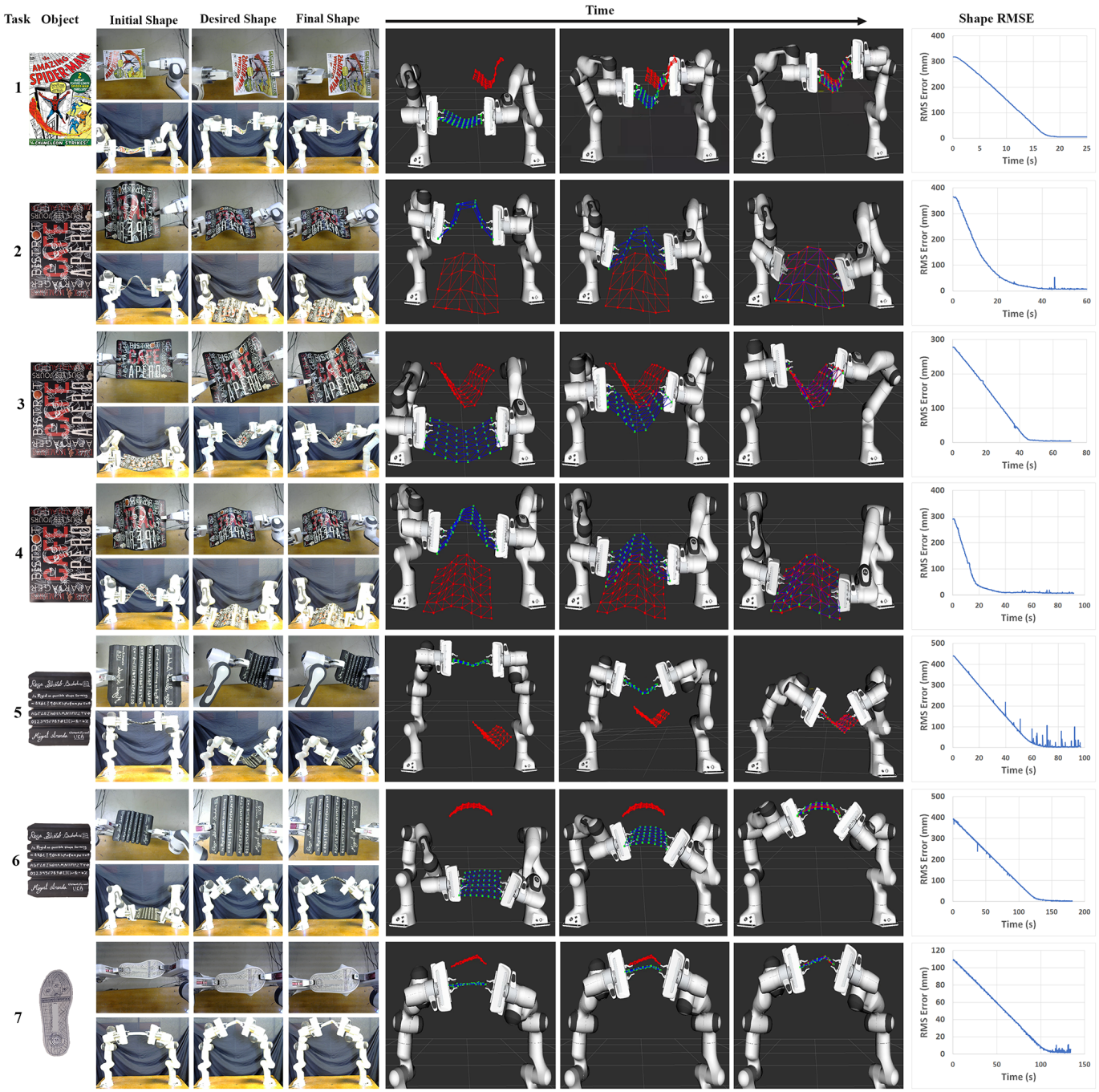

Fig. 5. Results for the seven tasks. For each we show, from left to right: image of the undeformed object; initial, desired and final shape (camera images on top, images from an external camera on bottom); initial, intermediate and final states represented in RViz; and evolution of error (RMS of $\mathbf{e}$) over time.

measured $\dot{\mathbf{s}}$ vector and each of the two predicted ones. The closer the values of this metric to 1, the better the alignment between the two vectors. From table II, the ARAP Jacobian exhibits better accuracy (higher averages, Avg) and stability (lower standard deviations, Std) overall.

TABLE II
PERFORMANCE EVALUATION OF SHAPE CHANGE PREDICTION.

Task	1		3		5	
Jacobian	Avg	Std	Avg	Std	Avg	Std
ARAP	0.839	0.045	0.816	0.052	0.890	0.029
DR	0.663	0.388	0.822	0.144	0.728	0.273

VIII. CONCLUSION

The proposed shape servoing scheme based on ARAP is simple yet effective. Its limitations include the following: as we use a surface ARAP model, the servoing is limited to thin-shell objects or surfaces of volumetric objects. The scheme is not designed to go across singular configurations (e.g., a perfectly flat shape for a planar object), or to servo non-elastic deformations. It needs continuous tracking of the shape, which may be difficult for large deformations. In this paper we did

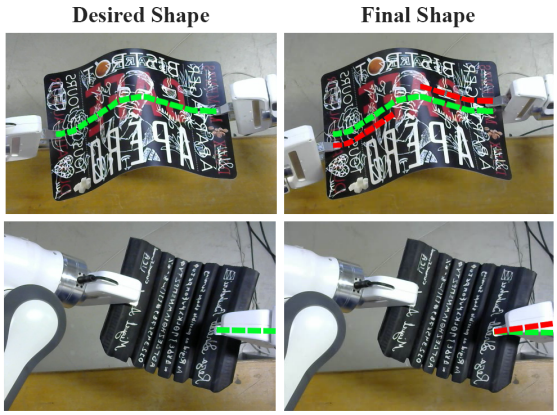

Fig. 6. Comparison between the poses of the end-effectors in the desired (green lines) and the final (red lines) shapes in Tasks 4 (top) and 5 (bottom). In Task 4 the end-effectors were displaced manually. In Task 5, the right end-effector rotated due to slippage during the task.

not consider the effect of gravity. We did not need to for the objects we handled. Still, for low-stiffness objects (i.e., cloth) incorporating gravity may be necessary. We would also like to explore extensions such as full volumetric shape servoing, scenarios with poorer shape visibility, or handling of contact with the environment.

IX. Acknowledgement

We thank Michelin and Plastinher Urban for providing, respectively, the sample of tire tread and the shoe sole that we used in our experiments.

References

[1] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini, "Dual-arm robotic manipulation of flexible cables," in *IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems (IROS)*, 2018, pp. 479–484.

[2] M. Aranda, J. A. Corrales Ramon, Y. Mezouar, A. Bartoli, and E. Özgür, "Monocular visual shape tracking and servoing for isometrically deforming objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7542–7549.

[3] A. Koessler, N. Roca Filella, B. C. Bouzgarrou, L. Lequièvre, and J. A. Corrales Ramon, "An efficient approach to closed-loop shape control of deformable objects using finite element models," in *IEEE Int. Conf. on Robotics and Automation (ICRA), to appear*, 2021.

[4] D. Navarro-Alarcon and Y.-H. Liu, "Fourier-Based Shape Servoing: A New Feedback Method to Actively Deform Soft Objects into Desired 2-D Image Contours," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 272–279, 2018.

[5] 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 Robotics and Automation Letters*, vol. 4, no. 2, pp. 254–261, 2019.

[6] C. Shin, P. Ferguson, S. Pedram, J. Ma, E. Dutson, and J. Rosen, "Autonomous tissue manipulation via surgical robot using learning based model predictive control," *IEEE International Conference on Robotics and Automation*, pp. 3875–3881, 2019.

[7] Z. Hu, P. Sun, and J. Pan, "Three-dimensional deformable object manipulation using fast online Gaussian process regression," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 979–986, 2018.

[8] Z. Hu, T. Han, P. Sun, J. Pan, and D. Manocha, "3-D deformable object manipulation using deep neural networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4255–4261, 2019.

[9] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Eurographics Symp. on Geometry Processing*, 2007, pp. 109—116.

[10] P. Jiménez, "Survey on model-based manipulation planning of deformable objects," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 154–163, 2012.

[11] S. Duenser, J. Bern, R. Poranne, and S. Coros, "Interactive robotic manipulation of elastic objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3476–3481.

[12] F. Ficuciello, A. Migliozzi, E. Coevoet, A. Petit, and C. Duriez, "FEM-based deformation control for dexterous manipulation of 3D soft objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 4007–4013.

[13] J. Sanchez, K. Mohy El Dine, J. A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Blind manipulation of deformable objects based on force sensing and finite element modeling," *Frontiers in Robotics and AI*, vol. 7, p. 73, 2020.

[14] J. Smolen and A. Patriciu, "Deformation planning for robotic soft tissue manipulation," in *Intern. Conf. on Advances in Computer-Human Interactions*, 2009, pp. 199–204.

[15] D. Navarro-Alarcon, Y.-H. Liu, J. Romero, and P. Li, "Model-free Visually Servoed Deformation Control of Elastic Objects by Robot Manipulators," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1457–1468, 2013.

[16] ——, "On the Visual Deformation Servoing of Compliant Objects: Uncalibrated Control Methods and Experiments," *International Journal of Robotics Research*, vol. 33, no. 11, pp. 1462–1480, 2014.

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

[18] R. Lagneau, A. Krupa, and M. Marchal, "Automatic shape control of deformable wires based on model-free visual servoing," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5252–5259, 2020.

[19] D. Berenson, "Manipulation of deformable objects without modeling and simulating deformation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 4525–4532.

[20] D. McConachie, A. Dobson, M. Ruan, and D. Berenson, "Manipulating deformable objects by interleaving prediction, planning, and control," *Int. J. of Rob. Research*, vol. 39, no. 8, pp. 957–982, 2020.

[21] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

[22] M. Alexa, D. Cohen-Or, and D. Levin, "As-rigid-as-possible shape interpolation," in *Proc. of the 27th Annual Conf. on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '00, 2000, p. 157–164.

[23] Z. Levi and C. Gotsman, "Smooth rotation enhanced as-rigid-as-possible mesh animation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 2, pp. 264–277, Feb. 2015.

[24] S. Parashar, D. Pizarro, A. Bartoli, and T. Collins, "As-rigid-as-possible volumetric shape-from-template," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 891–899.

[25] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[26] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi, "Motion2fusion: Real-time volumetric performance capture," *ACM Trans. Graph.*, vol. 36, no. 6, 2017.

[27] T. Han, X. Zhao, P. Sun, and J. Pan, "Robust shape estimation for 3D deformable object manipulation," *Communications in Information and Systems*, vol. 18, no. 2, pp. 107–124, 2018.

[28] D. Navarro-Alarcon, H. Yip, Z. Wang, Y.-H. Liu, F. Zhong, T. Zhang, and P. Li, "Automatic 3D Manipulation of Soft Objects by Robotic Arms with Adaptive Deformation Model," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 429–441, 2016.

[29] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Rob. & Aut. Mag.*, vol. 13, no. 4, pp. 82–90, 2006.

[30] A. Bartoli, Y. Gérard, F. Chadebecq, T. Collins, and D. Pizarro, "Shape-from-Template," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 2099–2118, 2015.

[31] D. T. Ngo, J. Östlund, and P. Fua, "Template-based monocular 3D shape recovery using Laplacian meshes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 172–187, 2016.

[32] E. Ozgur and A. Bartoli, "Particle-SfT: a Provably-Convergent, Fast Shape-from-Template Algorithm," *International Journal of Computer Vision*, vol. 123, no. 2, pp. 184–205, 2017.

[33] J. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking deformable objects with point clouds," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 1130–1137.

[34] A. Petit, V. Lippiello, G. A. Fontanelli, and B. Siciliano, "Tracking elastic deformable objects with an RGB-D sensor for a pizza chef robot," *Robotics and Auton. Systems*, vol. 88, pp. 187–201, 2017.