Model-Free 3-D Shape Control of Deformable Objects Using Novel Features Based on Modal Analysis

Bohan Yang, Bo Lu, *Member, IEEE*, Wei Chen, Fangxun Zhong, and Yun-Hui Liu, *Fellow, IEEE*

***Abstract*—Shape control of deformable objects is a challenging and important robotic problem. This article proposes a model-free controller using novel 3-D global deformation features based on modal analysis. Unlike most existing controllers using geometric features, our controller employs physically based deformation features designed by decoupling global deformation into low-frequency modes. Although modal analysis is widely adopted in computer vision and simulation, its usage in robotic deformation control is still an open topic. We develop a new model-free framework for the modal-based deformation control. Physical interpretation of the modes enables us to formulate an analytical deformation Jacobian matrix mapping the robot manipulation onto changes of the modal features. In the Jacobian matrix, unknown geometric and physical models of the object are treated as low-dimensional modal parameters, which can be used to linearly parameterize the closed-loop system. Thus, an adaptive controller with proven stability can be designed to deform the object while online estimating the modal parameters. Simulations and experiments are conducted using linear, planar, and volumetric objects under different settings. The results not only confirm the superior performance of our controller, but also demonstrate its advantages over the baseline method.**

***Index Terms*—Adaptive control, deformable object manipulation, modal analysis, visual servoing.**

I. Introduction

INTRODUCING autonomy of deformable object manipulation (DoM) can promote significant developments of robots in sectors of medical robotics [1], public service [2], industrial manufacturing [3], etc. However, unlike rigid object manipulation, DoM raises many difficult problems due to their complex deformation with varying physical properties and infinite dimensions.

During the past two decades, the field of DoM has gained distinctive progress. Early works mainly focused on simple point-based positioning [4], shape control with only simulation analysis [5], and specific task-oriented manipulation [6]. Recently, significant efforts have been devoted to more complex tasks, where researchers developed algorithms using advanced control strategies [7], [8], deformation representation methods with better description abilities [9], [10], and frameworks with multisensory feedback [11], [12]. The latest survey papers [13], [14], [15] reviewed the modeling, sensing, planning, and control strategies of robotic DoM. Among the reviewed research topics, automatic (local) deformation control is one of the most fundamental issues. Despite the efforts of the latest advances, real-time control of 3-D shape deformation without accurate models is still an open problem. Targeting this problem, we develop a model-free method for 3-D shape control of deformable objects using stereo vision feedback.

A. Related Work

1) Deformation Control: Existing deformation control strategies can be classified into three categories: model-based methods [5], [11], Jacobian-based methods [16], [17], and learning-based methods [8]. Due to difficulties in obtaining accurate object models, increasing attentions have been put on the Jacobian- and learning-based model-free deformation controllers. For model-free methods using deformation Jacobian approximations, Berenson et al. [16] proposed the concept of diminishing rigidity to approximate deformation Jacobian matrices for rope- and cloth-like objects. Shetab-Bushehri et al. [18] adopted the As-Rigid-As-Possible model for the Jacobian approximation of planar objects. However, these works need to know the geometry of the object-gripper configuration, which adds extra requirements for object sensing. Online estimation of the deformation Jacobian matrix is another popular model-free method. Navarro-Alarcon et al. [17] developed a series of works for geometric features of 3-D control points [7], 2-D curve descriptors using spline-based methods [19], as well as 2-D contour descriptors using

Manuscript received 2 May 2022; revised 8 December 2022; accepted 29 March 2023. Date of publication 9 May 2023; date of current version 8 August 2023. This work was supported in part by the Shenzhen Portion of Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone under Grant HZQB-KCZYB-20200089, in part by HK RGC under Grant T42-409/18-R and Grant 14202918, in part by Multi-Scale Medical Robotics Centre, InnoHK, and in part by CUHK T Stone Robotics Institute through VC Fund under Grant 4930745. This paper was recommended for publication by Associate Editor Dongheui Lee and Editor Mark Yim upon evaluation of the reviewers' comments. *(Corresponding author: Yun-Hui Liu.)*

Bohan Yang, Wei Chen, Fangxun Zhong, and Yun-Hui Liu are with the T Stone Robotics Institute, Department of Mechanical and Automation Engineering, Chinese University of Hong Kong, Hong Kong, SAR, China (e-mail: bhyang@mae.cuhk.edu.hk; weichen@link.cuhk.edu.hk; fxzhong@cuhk.edu.hk; yhliu@cuhk.edu.hk).

Bo Lu is with the T Stone Robotics Institute, Department of Mechanical and Automation Engineering, Chinese University of Hong Kong, Hong Kong, SAR, China, and also with the Robotics and Microsystems Center, School of Mechanical and Electric Engineering, Soochow University, Suzhou 215021, China (e-mail: blu@suda.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2023.3269347.

Digital Object Identifier 10.1109/TRO.2023.3269347

Fourier-based [9] and moment-based [10] methods. Lagneau et al. [20] proposed an active deformation controller for marker-based and markerless 3-D point features using online Jacobian estimation with a sliding window. Nevertheless, as these geometric deformation features become more complex or global, the deformation Jacobian matrix becomes more difficult to formulate and estimate. These works need to recalibrate local Jacobian matrices at different regions or offline testing deformation is required for preestimation of the Jacobian matrix. Moreover, the influence of the online Jacobian estimation on their control performances is hard to be analyzed and proved mathematically.

Learning-based methods serve as powerful tools for model-free deformation control because of their potential to generate complex feature-manipulation relationships using structures of neural networks [1], [3], [21], Gaussian process regression [22], and deep neural networks [8]. Early works of data-driven methods mainly studied deformation models [23] or manipulation strategies [2] without high accuracy requirements. For better accuracy, researchers have combined learning-based techniques with traditional control methods. Shin et al. [1] proposed two learning-based model predictive control algorithms to manipulate points on tissues. Hu et al. [8], [22] designed online learning methods within visual servoing frameworks to control 3-D deformation described by different geometric and point-cloud-based features. However, more complex deformation representations lead to higher requirements for data collecting and training. Although efforts have been put into different offline and online mechanisms, control performances of these learning-based methods highly rely on the learning architectures and parameters, which is hard to explain at both the mathematical and control levels. Combining the advantages of neural network structures and stable adaptive control laws, Yu et al. [24] and Li et al. [3] proposed deformation controllers for linear and planar objects. They employed adaptive neural networks to approximate the deformation Jacobian matrix with guaranteed control stability. Even though it is a promising technique, how to extend the method to more complex 3-D objects and shape control is still an issue to be further addressed.

2) Deformation Representation: Another crucial problem for deformation control is 3-D shape representation. Deformable shape modeling has been extensively studied in the fields of computer vision, computer graphics, and medical imaging. Review papers [14], [25] classified geometric methods into discrete representations (such as meshes [26], piecewise patches [27], and point clouds [28]), implicit curves or surfaces [29], explicit parameterized representations (such as the spline- [30] and Fourier- [31] based decompositions), and the free-form deformation [32]. Nevertheless, the relationship between geometric descriptors and robot manipulation is hard to formulate. Researchers have explored the potential to embed physical models into the deformation representation under robotic scenarios. Fugl et al. [33] formulated deformable curves as physical-model-based functions of material parameters, mesh geometry, and gripper pose. Other researchers [11], [34] represented deformation by relating sensor measurements with physical mesh models. However, directly using mesh models in the robotic manipulation strategy design presents many technical challenges due to the high-dimensional nature of common modeling techniques and the difficulties to obtain accurate models. For more compact shape representations, researchers also investigated a class of physically based methods using parameterized deformation. These methods [35], [36] first treated solid shapes as deformed results from some reference shapes, and then parameterized their deformation with low dimensions. To formulate the parametrized deformation, finding the reduced basis for subspace deformation [37], [38] is a hard key issue. Metaxas et al. [39] formulated the basis using deformable superquadrics but need to know the deformation form in advance. Krysl et al. [40] adopted the reduced basis using principal component analysis (PCA) but required an example dataset. Modal analysis is also a popular technique to generate the reduced basis for linear [41], [42] and nonlinear [37], [43] deformation. It is a standard tool for deformation reduction [41], [43] and 3-D shape recovery [44], [45]. However, robotic deformation control using modal analysis is still an open problem.

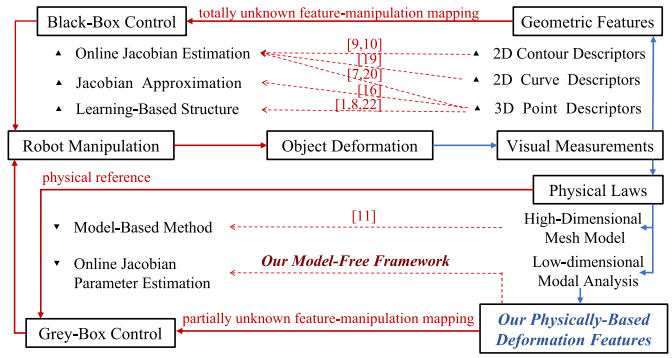


Fig. 1. Comparisons with the SOTA deformation controllers without accurate models. The blue lines are for deformation representations while the red lines are for deformation control.

Under the same representation-control framework in Fig. 1, we compare the reviewed state-of-the-art (SOTA) deformation controllers without accurate models. Existing model-free controllers use geometric features that directly describe the object geometry. Nevertheless, finding the analytical mapping relationship between the geometric features and robot manipulation is difficult because object deformation behaviors are highly coupled systems of high-dimensional object geometry and complex material properties. With unknown object models, they had to design the black-box control [46] laws with totally unknown feature-manipulation mapping. On the contrary, this article studies physically based representations because the physical laws employed in the object description also provide a physical reference to formulate the gray-box control [46] laws with partially unknown feature-manipulation mapping. In addition, unlike the model-based controller [11] using high-dimensional mesh models, our model-free controller adopts the low-dimensional modal analysis.

B. Our Contributions

This article proposes a model-free 3-D deformation controller using modal-based global deformation features. The advantages of designing deformation features with modal analysis are: common deformable shapes can be described with only a small number of low-frequency deformation modes [47]; physical interpretation of the modes enables an analytical formulation of the deformation Jacobian matrix mapping changes of the modal features with motions of the robot manipulation. However, technical challenges arise because linear modal analysis [48] relies on object models and has limited abilities for large deformation. To overcome these challenges, instead of analyzing object dynamics by performing modal analysis on the object, we only use low-frequency modes of a base mesh to span a feature space, where the object's global deformation can be uniquely represented with low dimensions. Then, based on the deformation features, we design an adaptive deformation controller that tackles the object's unknown geometric and physical models by online estimating unknown modal parameters in the Jacobian matrix. The stability of the designed controller is proved using Lyapunov-based methods. Compared to other SOTA model-free methods, our method:

1) proposes novel physically based, global deformation features using modal analysis. To the best of our knowledge, this is the first work adopting deformation modes to design deformation features for robotic 3-D deformation control;

2) designs a new analytical deformation Jacobian matrix where unknown object models are treated as unknown modal parameters with only low dimensions. These parameters can be used to linearly parameterize the closed-loop system, which enables us to develop stable control laws without requiring accurate model learning or parameter identifications;

3) can deal with different types (i.e., linear, planar, and volumetric) of objects and stereo features (i.e., points, curves, and contours). Extensive simulations and experiments are conducted for validation. The results confirm the superior performance of our method under different settings.

The rest of this article is organized as follows: Section II states the problem and overviews the algorithm flow. Section III defines our modal-based deformation feature space. Section IV presents the method to compute the deformation features in the feature space from object measurements. Section V derives the deformation Jacobian matrix between the deformation features and robot manipulation. Section VI designs an adaptive controller based on the deformation Jacobian matrix. Simulations and experiments are presented and discussed in Sections VII and VIII, respectively. Section IX presents discussions and conclusions. Finally, Section X concludes this article.

II. Problem Statements

A. Notation

In this article, we use lowercase bold letters to denote column vectors, capital bold letters to denote matrices, and italic letters to denote scalar quantities. Leading superscripts denote reference frames. Variables without a leading superscript are defined under the default reference frame. ${}^{a}\mathbf{T}_{b} \in \mathbb{R}^{4\times4}$ denotes a homogeneous transformation matrix from frame b to frame a. The symbol $\hat{}$ means estimated values; the symbol * means desired values.

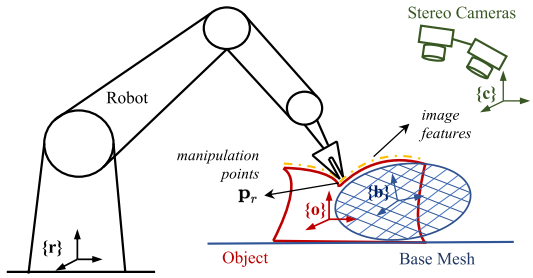


Fig. 2. System configuration of the 3-D deformation control problem. We consider four coordinate frames: the object frame $\{o\}$, the robot frame $\{r\}$, the camera frame $\{c\}$, and the base mesh frame $\{b\}$.

B. Problem Definition

We study the 3-D shape control of a deformable object with unknown physical properties and undeformed geometry. As shown in Fig. 2, the object is manipulated by a robot via manipulation points $\mathbf{p}_r$ (of size k). The object is monitored with a pair of static calibrated stereo cameras. To clarify our problem, the following assumptions are made.

Assumption 1: The object is rigidly and firmly grasped by the robot.

Assumption 2: Some image features (such as curves, contours, points, etc.) can be tracked on the surface of the object by both of the cameras in real time.

Assumption 3: The robot manipulating motion is sufficiently slow such that we can only consider the quasi-static elastic deformation of the object [7].

Assumption 4: The deformable points sampled from the object's image features and manipulated by the robot are elastically constrained in all directions around the equilibrium [7].

Consider the object with a point set $\mathbf{p}$ of size L ($L \to \infty$), its 3-D shape ${}^{o}\mathbf{x}(\mathbf{p},t) \in \mathbb{R}^{3L}$ (i.e., the position vector of $\mathbf{p}$) is high dimensional. To represent the shape with low dimensions, we design deformation features $\mathbf{s} \in \mathbb{R}^{m}$ ($m \ll 3L$) using low-frequency deformation modes of the base mesh. Then, based on the deformation features, we define the following 3-D shape control problem:

Problem: Given the desired shape ${}^{o}\mathbf{x}^{*}(\mathbf{p}) \in \mathbb{R}^{3l}$ and deformation features $\mathbf{s}^{*} \in \mathbb{R}^{m}$, control the velocities $\mathbf{v}(\mathbf{p}_r,t) \in \mathbb{R}^{3k}$ of $\mathbf{p}_r$ using feedback of the deformation feature errors $\mathbf{e}_s(t) = \mathbf{s}(t) - \mathbf{s}^{*} \in \mathbb{R}^{m}$, such that as $\mathbf{s}(t) \to \mathbf{s}^{*}$, ${}^{o}\mathbf{x}(\mathbf{p},t) \to {}^{o}\mathbf{x}^{*}(\mathbf{p})$.

C. Algorithm Outline

As shown in Fig. 3, the major steps of our algorithm are summarized as follows.

1) We introduce the base mesh and compute its low-frequency modes. By combining the modes according to the object-mesh mapping, we formulate the deformation

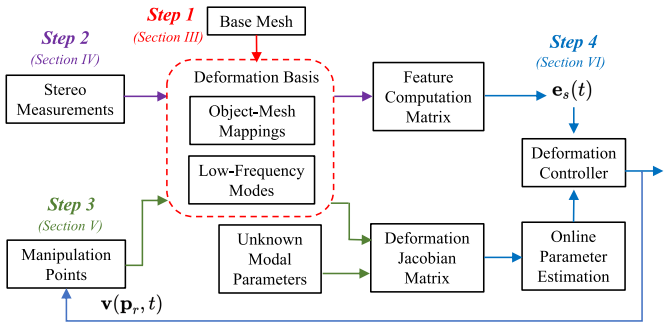


Fig. 3. Schematic representation of the proposed algorithm flow.

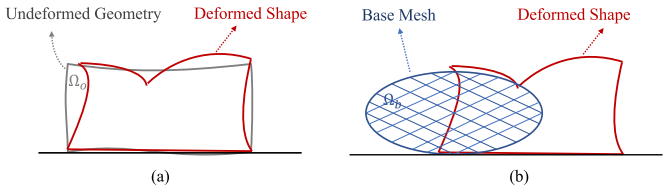


Fig. 4. Formulation of the object's global deformation relative to: (a) the undeformed geometry of the object; (b) the base mesh.

basis that defines a low-dimensional deformation feature space.

2) Given the deformation basis, we derive the deformation feature computation matrix, using which stereo measurements of the object can be projected to the feature space.
3) Given the deformation basis, we formulate the deformation Jacobian matrix with unknown modal parameters.
4) Using the feature computation and the Jacobian matrices, we design a deformation controller that sends manipulation commands while online estimating the unknown parameters.

III. Modal-Based Deformation Feature Space

Modal analysis [41], [48] is a well-known tool for the dimension reduction of global deformation. However, it requires deformation models of the object, which are hard to be obtained accurately on practical occasions. In this section, a low-dimensional deformation feature space is generated using modal analysis but with unknown object models.

A. Modal Decomposition of 3-D Global Deformation

We begin with a brief introduction of how modal analysis is commonly used to parameterize global deformation with low dimensions. As shown in Fig. 4(a), 3-D global deformation can be defined by mapping the object (with domain Ω_o) to itself with new deformed coordinates [49]

$$^o\mathbf{x}(\mathbf{p}, t) = {}^o\mathbf{x}(\mathbf{p}) + {}^o\mathbf{u}(\mathbf{p}, t) \tag{1}$$

where the position vector $^o\mathbf{x}(\mathbf{p}, t)$ quantifies the 3-D shape of the object during deformation; $^o\mathbf{x}(\mathbf{p}) \in \mathbb{R}^{3L}$ is the undeformed position vector of the object points $\mathbf{p}$ (i.e., the undeformed geometry of the object); $^o\mathbf{u}(\mathbf{p}, t) \in \mathbb{R}^{3L}$ is the displacement vector of $\mathbf{p}$ (i.e., the displacement field of the object). Then, according to the modal decomposition techniques [41], [48], the displacement field can be decoupled into a set of frequency-ordered, orthogonal modes

$$^o\mathbf{u}(\mathbf{p}, t) = \sum_{j}^{3L} \boldsymbol{\phi}_o^j c_j(t) \tag{2}$$

where $\boldsymbol{\phi}_o^j \in \mathbb{R}^{3L}$ is the jth deformation mode of the object whose coefficient is $c_j(t)$. As discussed in [50], high-frequency modes typically have little energy and less effect on the overall 3-D shape. Thus, by discarding high-frequency modes, the truncated mode coefficients can form a low-dimensional approximation for the 3-D global deformation of the object.

Another advantage of using modal descriptions in robotic DoM is that the physical interpretation of the modes facilitates an analytical formulation of the deformation Jacobian matrix. Specifically, the mode shape vector $\boldsymbol{\phi}_o^j$ describes how positions of the object points $(^o\mathbf{x}(\mathbf{p}_i, t) = (x_i(t), y_i(t), z_i(t))^T, i = \{1, \ldots, L\})$ change in the mode as a function of the coefficient $c_j(t)$ [47]

$$\boldsymbol{\phi}_o^j \propto \left(\frac{\partial x_1}{\partial c_j}, \ \frac{\partial y_1}{\partial c_j}, \ \frac{\partial z_1}{\partial c_j}, \ \cdots \ \frac{\partial x_L}{\partial c_j}, \ \frac{\partial y_L}{\partial c_j}, \ \frac{\partial z_L}{\partial c_j} \right)^T. \tag{3}$$

In this way, it naturally defines a Jacobian relationship between $^o\mathbf{u}(\mathbf{p}, t)$ and c_j. By extracting the rows of the manipulation points from the mode shape vectors, we can analytically construct the deformation Jacobian matrix mapping motions of the points onto changes of the modal coefficients.

Unfortunately, besides the reliance on object models, the modal decomposition techniques are also limited by the local nature of the linear modal analysis when analyzing object dynamics under large deformation. To avoid these problems, we do not compute the dynamic behaviors of the object or perform modal analysis on it. Instead, we propose a model-free framework to generate a deformation feature space using low-frequency modes of a base mesh.

B. Base Mesh and Object-Mesh Mapping

The base mesh used in our model-free framework is selected to be an ellipsoidal mesh. Consider an ellipsoid with the following surface vector [51]:

$$^c\boldsymbol{\gamma}(\zeta, \sigma) = {}^c\mathbf{T}_b \begin{bmatrix} a_x cos\zeta cos\sigma \\ a_y cos\zeta sin\sigma \\ a_z sin\zeta \end{bmatrix}^T \tag{4}$$

where $\zeta \in [-\pi/2, \pi/2]$ is the latitude while $\sigma \in [-\pi, \pi)$ is the longitude; $\{a_x, a_y, a_z\}$ are the size parameters in x, y, and z dimension, respectively; $^c\mathbf{T}_b \in \mathbb{R}^{4\times4}$ is its pose with respect to the camera frame. Discreting the ellipsoid, we obtain the base mesh with the domain Ω_b and the node set $\mathbf{n} \in \Omega_b$ of size N $(N \ll L)$. Using $^c\mathbf{T}_b$, stereo measurements of the object during deformation can be transformed to the base mesh frame. In the rest of our derivations, we set the base mesh frame $\{b\}$ as our default reference frame.

Following the shape modeling techniques in [39], [51], we treat the object shape as the deformed result from the base mesh. Then, as shown in Fig. 4(b), the object's global deformation can be reformulate with

$$\mathbf{x}(\mathbf{p},t) = \mathbf{x}(\boldsymbol{\eta}(\mathbf{p})) + \mathbf{u}(\mathbf{p},\boldsymbol{\eta}(\mathbf{p}),t) \tag{5}$$

where $\boldsymbol{\eta}(\mathbf{p}) \in \partial\Omega_b$ are the projections of object points $\mathbf{p}$ on the surface of the base mesh and $\mathbf{x}(\boldsymbol{\eta}(\mathbf{p})) \in \mathbb{R}^{3L}$ is their position vector. For a static base mesh with no deformation, the object-mesh displacements $\mathbf{u}(\mathbf{p},\boldsymbol{\eta}(\mathbf{p}),t) \in \mathbb{R}^{3L}$ can be used to quantify the object shape. Note that pose and size differences between the object and the base mesh can be absorbed in $\mathbf{u}(\mathbf{p},\boldsymbol{\eta}(\mathbf{p}),t)$. Thus, when generating the base mesh, we do not need to calibrate the object's pose and size to set (4). The mesh pose ${}^c\mathbf{T}_b$ and size parameters a_x and a_y can be roughly estimated using the stereo measurements of the object at the rest configuration (i.e., the configuration before the robot runs). Only a rough estimation of the object's thickness is required to set a_z.

Remark 1: The mesh pose ${}^c\mathbf{T}_b$ and sizes a_x and a_y can be computed using 3-D measurements of the object points, which are sampled from the image features (on the object surface) and manipulated by the robot. Besides the moment-based computation method in [44], we also propose a simple solution in Algorithm 1 (shown in Section VIII).

Then, we establish the object-mesh mapping to calculate $\mathbf{u}(\mathbf{p},\boldsymbol{\eta}(\mathbf{p}),t)$ in (5) and to further allocate it to mesh nodes. Inspired by the radially projecting virtual spring attachment method [44], [52], the mapping is computed as follows: First, for each object point $\mathbf{p}^w$ ($w = \{1,\ldots,L\}$) at the rest configuration, we project it to the mesh surface along its connection line with the mesh center and set the intersection surface point to be its mesh projection $\boldsymbol{\eta}(\mathbf{p}^w)$. Second, we attach a virtual spring between $\mathbf{p}^w$ and $\boldsymbol{\eta}(\mathbf{p}^w)$. Then, their object-mesh displacement $\mathbf{u}(\mathbf{p}^w,\boldsymbol{\eta}(\mathbf{p}^w),t) \in \mathbb{R}^3$ can be allocated among the nodes of the intersection element $L(\mathbf{p}^w)$ via

$$\mathbf{u}(\mathbf{p}^w,\boldsymbol{\eta}(\mathbf{p}^w),t) = \sum_{\mathbf{n}^i \in L(\mathbf{p}^w)} N_{w,i}(\boldsymbol{\eta}(\mathbf{p}^w),\mathbf{n}^i)\mathbf{u}(\mathbf{n}^i,t) \tag{6}$$

where $\mathbf{n}^i$ is the node whose global index in the base mesh is i; the scalar $N_{w,i}(\boldsymbol{\eta}(\mathbf{p}^w),\mathbf{n}^i)$ is the allocation weight of $\mathbf{n}^i$ with respect to $\boldsymbol{\eta}(\mathbf{p}^w)$, which is computed using the finite element (FE) shape function [53] of $L(\mathbf{p}^w)$; $\mathbf{u}(\mathbf{n}^i,t) \in \mathbb{R}^3$ is the allocated displacement vector of $\mathbf{n}^i$. Finally, by assembling the element object-mesh mappings in (6) for all object points $\mathbf{p}$, we can construct the global object-mesh mapping

$$\mathbf{u}(\mathbf{p},\boldsymbol{\eta}(\mathbf{p}),t) = \mathbf{N}(\boldsymbol{\eta}(\mathbf{p}),\mathbf{n})\mathbf{u}(\mathbf{n},t) \tag{7}$$

where $\mathbf{N}(\boldsymbol{\eta}(\mathbf{p}),\mathbf{n}) \in \mathbb{R}^{3L\times 3N}$ is the global object-mesh allocating matrix and $\mathbf{u}(\mathbf{n},t) \in \mathbb{R}^{3N}$ is the allocated nodal displacement field. However, this nodal displacement field still has high dimensions. The next step is to derive a reduced deformation basis by performing modal truncation on $\mathbf{u}(\mathbf{n},t)$.

C. Modal Truncation on Base Mesh

Given a set of arbitrarily assigned physical properties (including Young's modulus E, Poisson's ratio v, and the total mass M), we compute the general stiffness matrix $\mathbf{K} \in \mathbb{R}^{3N\times 3N}$ and mass matrix $\mathbf{M} \in \mathbb{R}^{3N\times 3N}$ of the base mesh using the finite element method (FEM). Following the approaches in [48], we compute the undamped free deformation (or vibration) modes of the base mesh by solving the following generalized eigenproblem:

$$\mathbf{K}\boldsymbol{\phi} = \omega^2\mathbf{M}\boldsymbol{\phi} \tag{8}$$

which has three N eigensolutions

$$\left\{(\omega_1^2,\boldsymbol{\phi}^1), \quad \ldots \quad (\omega_{3N}^2,\boldsymbol{\phi}^{3N})\right\}, \quad \omega_1^2 \le \ldots \le \omega_{3N}^2$$

where the eigenvector $\boldsymbol{\phi}^j \in \mathbb{R}^{3N}$ is the jth mode shape vector while ω_j is the corresponding natural frequency. Then, we form the mode shape matrix of the base mesh using these frequency-ordered eigenvectors

$$\boldsymbol{\Phi} = \begin{bmatrix}\boldsymbol{\phi}^1 & \boldsymbol{\phi}^2 & \ldots & \boldsymbol{\phi}^{3N}\end{bmatrix}.$$

Note that $\boldsymbol{\Phi} \in \mathbb{R}^{3N\times 3N}$ is $\mathbf{M}$-orthonormal and $\mathbf{K}$-orthogonal. We then discard high-frequency modes and normalize the retained m ($m \ll 3N$) modes. Afterward, we use the modal truncation to approximate the nodal displacements with

$$\mathbf{u}(\mathbf{n},t) = \sum_{j=1}^{m} \boldsymbol{\phi}_n^j s_j(t) \tag{9}$$

where $\boldsymbol{\phi}_n^j \in \mathbb{R}^{3N}$ is the jth normalized mode whose coefficient is s_j. Note that these modes are computed using the deformation model of the base mesh rather than the object. Even though different deformation models lead to different sets of mode shapes, each set satisfies similar orthogonal and frequency-ordered properties to span a low-dimensional deformation feature space. Thus, the truncated modal coefficients $\mathbf{s}(t) = \{s_1(t),\ldots,s_m(t)\}^T$ can form a unique and compact deformation representation.

D. Model-Free Deformation Basis

Choosing $\mathbf{s}(t)$ to be our deformation features, based on the above discussions, we derive the following modal-based deformation basis $\mathbf{B}(\mathbf{p},\mathbf{n}) \in \mathbb{R}^{3L\times m}$:

$$\begin{aligned}\mathbf{u}(\mathbf{p},\mathbf{n},t) &= \sum_{j=1}^{m}\underbrace{\sum_{i=1}^{N}\mathbf{N}_i(\boldsymbol{\eta}(\mathbf{p}),\mathbf{n}^i)[\boldsymbol{\phi}_n^j]_i}_{\mathbf{b}^j(\mathbf{p},\mathbf{n})} s_j(t)\\ &= \mathbf{N}(\boldsymbol{\eta}(\mathbf{p}),\mathbf{n})\boldsymbol{\Phi}_n\mathbf{s}(t) = \mathbf{B}(\mathbf{p},\mathbf{n})\mathbf{s}(t)\end{aligned} \tag{10}$$

where $\mathbf{N}_i(\boldsymbol{\eta}(\mathbf{p}),\mathbf{n}^i) \in \mathbb{R}^{3L\times 3}$ is the local object-mesh allocating matrix for $\mathbf{n}^i$; the subvector $[\boldsymbol{\phi}_n^j]_i \in \mathbb{R}^3$ consists of the variables from $\boldsymbol{\phi}_n^j$ that correspond to $\mathbf{n}^i$; the vector $\mathbf{b}^j(\mathbf{p},\mathbf{n}) \in \mathbb{R}^{3L}$ (the column vector of the basis $\mathbf{B}(\mathbf{p},\mathbf{n})$) can be regarded as the Ritz basis function in [40] (unlike [40], we formulate the *priori* vectors by combining the base mesh's low-frequency modes according to the object-mesh mapping); since the modes are generated in a model-free manner, we call $\boldsymbol{\Phi}_n \in \mathbb{R}^{3N\times m}$ the truncated reference mode shape matrix and the deformation feature space they span the truncated reference modal space. Note that our deformation features are designed by approximating

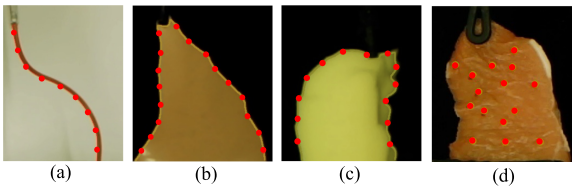


Fig. 5. Examples of obtaining surface samplings (the red dots) from different image features: (a) sampling points from the curve feature of a linear object by arc length; (b), (c) sampling points from the contour feature of a planar/volumetric object by arc length; (d) tracking point features on the surface of an animal (pork) tissue.

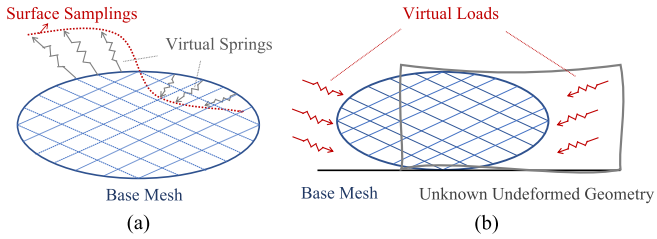


Fig. 6. Illustrations of (a) the virtual springs between surface samplings and their bash mesh projections; (b) the virtual loads to deform the base mesh to fit the undeformed object geometry.

the object's global deformation using a linear low-dimensional combination of the whole-body deformation modes. Thus, unlike the existing geometric deformation features that describe the (deformed) positions of some measurable parts of the object, our deformation features represent the global deformation of the object. By setting the feature dimension m to be equal or greater than the measurement dimension, overconstrained solutions of $\mathbf{s}(t)$ can be still computed even with partial stereo measurements. It should also be mentioned that our deformation basis can be formulated for linear, planar, and volumetric objects and can deal with stereo measurements of different image features.

IV. Deformation Feature Computation

This section presents the method to compute our modal-based deformation features in the truncated reference modal space from stereo measurements of the object.

A. Surface Samplings

As the proposed deformation basis directly transforms between object points and the deformation features $\mathbf{s}(t)$, we can compute $\mathbf{s}(t)$ by simply sampling points from the image features tracked on the object (with respect to Assumption 2). Given the tracked image features $\mathbf{y}(t) \in \mathbb{R}^{\varrho(t)\times 4}$ (i.e., a pixel vector in the pair of 2-D images obtained from the stereo cameras), we sample l points from them and compute the 3-D positions of the points

$$\mathbf{x}(\mathbf{p}_s, t) = f(\mathbf{y}(t)) \tag{11}$$

where the sampled points $\mathbf{p}_s$ are called the surface samplings, and $\mathbf{x}(\mathbf{p}_s, t) \in \mathbb{R}^{3l}$ is their position vector. $f : \mathbb{R}^{\varrho(t)\times 4} \to \mathbb{R}^{3l}$ is a nonlinear function of the point sampling and reconstruction process. Fig. 5 shows some examples of the point sampling from different image features.

B. Object-Mesh Mapping of Surface Samplings

To establish the relationship between the surface samplings $\mathbf{p}_s$ and our deformation features, we first formulate the object-mesh mapping of $\mathbf{p}_s$ such that their corresponding part in the deformation basis can be computed. Given $\mathbf{p}_s$, following the method in Section III-B, we can find their base mesh projections $\boldsymbol{\eta}(\mathbf{p}_s)$ and compute their object-mesh displacements $\mathbf{u}(\mathbf{p}_s, \boldsymbol{\eta}(\mathbf{p}_s), t) \in \mathbb{R}^{3l}$ with

$$\mathbf{u}(\mathbf{p}_s, \boldsymbol{\eta}(\mathbf{p}_s), t) = \mathbf{x}(\mathbf{p}_s, t) - \mathbf{x}(\boldsymbol{\eta}(\mathbf{p}_s)) \tag{12}$$

where $\mathbf{x}(\boldsymbol{\eta}(\mathbf{p}_s)) \in \mathbb{R}^{3l}$ is the position vector of $\boldsymbol{\eta}(\mathbf{p}_s)$. For each of $\boldsymbol{\eta}(\mathbf{p}_s)$, we compute its element object-mesh mapping in (6). Then, all the element mappings of $\boldsymbol{\eta}(\mathbf{p}_s)$ are assembled into the following matrix form:

$$\mathbf{u}(\mathbf{p}_s, \boldsymbol{\eta}(\mathbf{p}_s), t) = \mathbf{N}_s(\boldsymbol{\eta}(\mathbf{p}_s), \mathbf{n}_s)\mathbf{u}(\mathbf{n}_s, t) \tag{13}$$

where $\mathbf{n}_s$ (of size n) are the surface samplings' allocation nodes with nonzero allocation weights; $\mathbf{N}_s(\boldsymbol{\eta}(\mathbf{p}_s), \mathbf{n}_s) \in \mathbb{R}^{3l\times 3n}$ is the (local) object-mesh allocating matrix for surface samplings, which can be regarded as a submatrix consisting of the rows of $\boldsymbol{\eta}(\mathbf{p}_s)$ and the columns of $\mathbf{n}_s$ from the global object-mesh allocating matrix $\mathbf{N}(\boldsymbol{\eta}(\mathbf{p}), \mathbf{n})$; $\mathbf{u}(\mathbf{n}_s, t) \in \mathbb{R}^{3n}$ is the nodal displacement vector of $\mathbf{n}_s$. Afterward, we compute $[\boldsymbol{\Phi}_n]_s \in \mathbb{R}^{3n\times m}$ by extracting the rows of $\mathbf{n}_s$ from $\boldsymbol{\Phi}_n$, using which the nodal displacements $\mathbf{u}(\mathbf{n}_s, t)$ can be represented with our deformation features using (9).

As our deformation features describe the global deformation rather than the local point positions, we do not require the surface samplings to be the same object points. During the deformation, the surface samplings can slip on the object surface, whose influence can be absorbed in their object-mesh displacements. This benefits the implementation of our method for practical applications. By comparison, some existing works require artificial markers [7], [24] or textured objects [18] to track points.

C. Deformation Feature Computation Matrix

Computing our deformation features from 3-D position measurements of surface samplings requires the inverse process of (13). However, as the surface samplings $\mathbf{p}_s$ are usually sparser than their allocated nodes $\mathbf{n}_s$ ($l \leq n$), the direct inverse of (13) is underdetermined. To avoid solving the underdetermined problem, according to the virtual spring attachment method [44], [52], we specify displacement measurements as virtual forces that deform the object to fit the measured points through virtual springs [Fig. 6(a)]. In this way, given the principle of virtual work, the point deformation $\mathbf{u}(\mathbf{p}_s, \boldsymbol{\eta}(\mathbf{p}_s), t)$ can be allocated among the mesh nodes using the transpose (rather than the inverse) of $\mathbf{N}_s(\boldsymbol{\eta}(\mathbf{p}_s), \mathbf{n}_s)$:

$$\mathbf{r}(\mathbf{n}_s, t) = \mathbf{N}_s^T(\boldsymbol{\eta}(\mathbf{p}_s), \mathbf{n}_s)\mathbf{u}(\mathbf{p}_s, \boldsymbol{\eta}(\mathbf{p}_s), t) \tag{14}$$

where $\mathbf{r}(\mathbf{n}_s, t) \in \mathbb{R}^{3n}$ is the allocated virtual force vector of nodes $\mathbf{n}_s$. It can be further transformed to the modal forces

$\widetilde{\mathbf{r}}(t) \in \mathbb{R}^m$ with

$$\widetilde{\mathbf{r}}(t) = [\mathbf{\Phi}_n]_s^T \mathbf{r}(\mathbf{n}_s, t). \tag{15}$$

Nevertheless, in (14), displacement measurements are allocated as virtual forces. We need to add a rectified term between the force variables and the displacement variables in the truncated reference modal space. Following the relative derivations in [44], we perform the force-displacement rectification with

$$\mathbf{s}(t) = (\widetilde{\mathbf{K}} + \mathbf{I}_6)^{-1} \widetilde{\mathbf{r}}(t) \tag{16}$$

where $\mathbf{I}_6 \in \mathbb{R}^{m \times m}$ is a diagonal matrix whose first six diagonal elements are ones, while the other elements are zeros; the diagonal matrix $\widetilde{\mathbf{K}} \in \mathbb{R}^{m \times m}$ is the normalized modal stiffness matrix (unlike [44], we compute a normalized version with $\widetilde{\mathbf{K}} = \mathbf{\Phi}_n^T \mathbf{K} \mathbf{\Phi}_n$). The matrix $\mathbf{I}_6$ is added to $\widetilde{\mathbf{K}}$ because under the base mesh frame, eigenvalues of the first six modes (the rigid motion modes) are zeros.

Based on the above discussions, we can compute our 3-D deformation features via

$$\mathbf{s}(t) = \mathbf{D}_\Phi(\mathbf{n}_s) \mathbf{D}_N(\boldsymbol{\eta}(\mathbf{p}_s), \mathbf{n}_s) \mathbf{u}(\mathbf{p}_s, \boldsymbol{\eta}(\mathbf{p}_s), t) \tag{17}$$

where $\mathbf{D}_N(\boldsymbol{\eta}(\mathbf{p}_s), \mathbf{n}_s) \in \mathbb{R}^{3n \times 3l}$ is the mesh projection matrix that projects point measurements of the object to the nodal space of the base mesh with

$$\mathbf{D}_N(\boldsymbol{\eta}(\mathbf{p}_s), \mathbf{n}_s) = \mathbf{N}_s^T(\boldsymbol{\eta}(\mathbf{p}_s), \mathbf{n}_s) \tag{18}$$

and $\mathbf{D}_\Phi(\mathbf{n}_s) \in \mathbb{R}^{m \times 3n}$ is the modal projection matrix that transforms nodal variables of the base mesh to the deformation features in the truncated reference modal space with

$$\mathbf{D}_\Phi(\mathbf{n}_s) = (\widetilde{\mathbf{K}} + \mathbf{I}_6)^{-1} [\mathbf{\Phi}_n]_s^T. \tag{19}$$

Although the use of virtual forces is a well-established technique in some shape modeling works [44], [52], the definitions and computations of the forces are not physically accurate. Those works define the virtual forces to deform the object to fit sensor measurements, and then solve the equilibrium displacements (that constitute the recovered shape) numerically or through several iterations. By comparison, we only use the virtual forces to project stereo measurements to the deformation feature space such that the object deformation is uniquely represented (rather than accurately recovered). To ensure the unique representation, we formulate the deformation feature computation matrix whose rank satisfies

$$rank(\mathbf{D}_\Phi(\mathbf{n}_s) \mathbf{D}_N(\boldsymbol{\eta}(\mathbf{p}_s), \mathbf{n}_s)) = m \tag{20}$$

and also require the dimension of surface samplings satisfies $l \geq \frac{m}{3}$. In addition, we need to further deal with the modeling uncertainties introduced by the virtual forces when designing control laws.

Remark 2: The deformation feature computation matrices are formulated at the rest configuration. They do not need to be reformulated unless some one-to-one correspondences of the surface samplings are lost due to occlusions, in which case we can reassemble the matrices $\mathbf{N}_s(\boldsymbol{\eta}(\mathbf{p}_s), \mathbf{n}_s)$ and $[\mathbf{\Phi}_n]_s$ according to the changed configurations of $\mathbf{p}_s$ and $\mathbf{n}_s$.

V. Deformation Jacobian Matrix With Unknown Modal Parameters

To formulate robotic manipulation strategies, we need to analyze how robot manipulation affects our deformation features. In Euclidean space, the dynamics of a deformable object under external manipulation are described by partial differential equations [54] with infinite dimensions. To investigate the dynamics in our deformation feature space, we consider the following low-dimensional approximation of the object dynamics using the model reduction techniques in [55], [56]:

$$\widetilde{\mathbf{M}}_o \ddot{\mathbf{s}}(t) = \widetilde{\mathbf{F}}(\mathbf{s}(t), \dot{\mathbf{s}}(t)), \mathbf{x}(\mathbf{p}_r, t)) \tag{21}$$

where $\widetilde{\mathbf{M}}_o \in \mathbb{R}^{m \times m}$ is the mass matrix of the object in the truncated reference modal space; $\dot{\mathbf{s}} \in \mathbb{R}^m$ and $\ddot{\mathbf{s}} \in \mathbb{R}^m$ are the first and second order derivative of our deformation features $\mathbf{s}(t)$; $\mathbf{x}(\mathbf{p}_r, t) \in \mathbb{R}^{3k}$ $(3k \leq m)$ is the position vector of the manipulation points $\mathbf{p}_r$; the nonlinear term $\widetilde{\mathbf{F}}(\cdot) \in \mathbb{R}^m$ is the superposition of elastic forces, damping forces, and external forces of the object in the truncated reference modal space. However, (21) is hard to solve with unknown deformation models of the object. Focusing on kinematic control under the quasi-static assumption (Assumption 3), this article does not consider the object dynamics in our controller design. Instead, we formulate a deformation Jacobian matrix between $\dot{\mathbf{s}}(t)$ and $\dot{\mathbf{x}}(\mathbf{p}_r, t) \in \mathbb{R}^{3k}$, the first order derivative of $\mathbf{x}(\mathbf{p}_r, t)$, and treat the unknown object models as low-dimensional unknown parameters in the Jacobian matrix.

A. Quasi-Static Deformation in Truncated Reference Modal Space

Under the quasi-static assumption, the dominant effect of the object deformation is its potential energy. Thus, we investigate the potential energy of the object to formulate the relationship between $\dot{\mathbf{s}}(t)$ and $\dot{\mathbf{x}}(\mathbf{p}_r, t)$. Note that the quasi-static assumption is widely used in the existing works of deformation control (such as [7], [22], [24], [57]). Unlike these works studying the potential energy in Euclidean space, we formulate the potential energy in the truncated reference modal space using the subspace domain integral [38]

$$W(\mathbf{s}(t)) = \int_{\Omega_b} \psi(\mathbf{p}; \mathbf{s}(t)) d\Omega_{\mathbf{p}} \tag{22}$$

where $W(\mathbf{s}(t)) : \mathbb{R}^m \to \mathbb{R}$ is the subspace potential energy function and $\psi(\mathbf{p}; \mathbf{s}(t))$ is the nonnegative energy density at the object point $\mathbf{p}$ in the point domain $\Omega_{\mathbf{p}}$. Then, the equation of equilibrium in the modal space follows:

$$\frac{\partial W}{\partial \mathbf{s}}(\mathbf{s}(t)) + \widetilde{\mathbf{f}}(\mathbf{x}(\mathbf{p}_r, t)) = \mathbf{0}. \tag{23}$$

Solving this equation requires modeling knowledge of the object. To avoid that, we locally focus on a linear approximation of the relationship between $\delta \mathbf{s} \in \mathbb{R}^m$ (a small change of $\mathbf{s}(t)$) and $\delta \mathbf{x}(\mathbf{p}_r) \in \mathbb{R}^{3k}$ (a small change of $\mathbf{x}(\mathbf{p}_r, t)$). To construct the local approximation, inspired by [7], [58], we linearize (23)

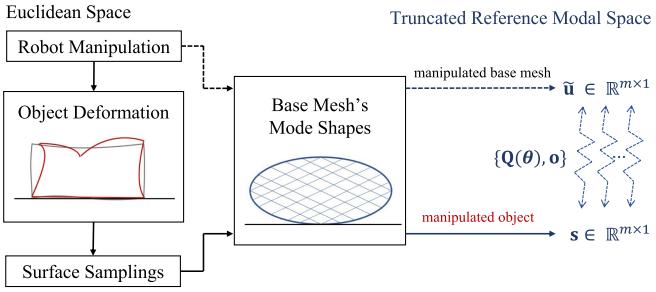


Fig. 7. Illustration of the deformation Jacobian derivation. The solid lines indicate the direct path from robot manipulation to our deformation features while the dotted lines indicate the path we derive the Jacobian matrix. The blue springs illustrate the local elastic relations between the modal displacements of the base mesh and the object in (27).

around the equilibrium into

$$\left(\frac{\partial^2 W}{\partial \mathbf{s}\partial \mathbf{s}}+\frac{\partial \widetilde{\mathbf{f}}}{\partial \mathbf{s}}\right)\delta\mathbf{s}+\left(\frac{\partial^2 W}{\partial \mathbf{s}\partial \mathbf{x}(\mathbf{p}_r,t)}+\frac{\partial \widetilde{\mathbf{f}}}{\partial \mathbf{x}(\mathbf{p}_r,t)}\right)\delta\mathbf{x}(\mathbf{p}_r)=\mathbf{0} \tag{24}$$

where $\partial W/\partial \mathbf{s}$ and $\widetilde{\mathbf{f}}(\mathbf{x}(\mathbf{p}_r,t))$ are regarded as internal and external forces in the modal space. We denote their counterparts in Euclidean space by $\mathbf{f}_{int}(t)$ and $\mathbf{f}(t)$, respectively. Then, (24) can be rewritten as

$$\underbrace{\left(\frac{\partial(\partial W/\partial \mathbf{s})}{\partial \mathbf{s}}+\frac{\partial \widetilde{\mathbf{f}}}{\partial \mathbf{s}}\right)}_{\widetilde{\mathbf{A}}}\delta\mathbf{s} + \underbrace{\left(\frac{\partial(\partial W/\partial \mathbf{s})}{\partial \mathbf{f}_{int}}\frac{\partial \mathbf{f}_{int}}{\partial \mathbf{x}(\mathbf{p}_r,t)}+\frac{\partial \widetilde{\mathbf{f}}}{\partial \mathbf{f}}\frac{\partial \mathbf{f}}{\partial \mathbf{x}(\mathbf{p}_r,t)}\right)}_{\widetilde{\mathbf{B}}}\delta\mathbf{x}(\mathbf{p}_r)=\mathbf{0} \tag{25}$$

where $\widetilde{\mathbf{A}} \in \mathbb{R}^{m\times m}$ and $\widetilde{\mathbf{B}} \in \mathbb{R}^{m\times 3k}$ are unknown constant relationship matrices computed with unknown stiffness matrices. Given Assumption 4 and (20), the relationship matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ have full column ranks. Then, for slow robot manipulation (where $\dot{\mathbf{s}}(t) \approx \delta\mathbf{s}/\delta t$ and $\dot{\mathbf{x}}(\mathbf{p}_r,t) \approx \delta\mathbf{x}(\mathbf{p}_r)/\delta t$ hold for δt, a small time interval), we can establish a Jacobian relationship between $\dot{\mathbf{s}}(t)$ and $\dot{\mathbf{x}}(\mathbf{p}_r,t)$.

However, formulating or estimating a deformation Jacobian matrix is a common technical challenge in the related works. To derive our deformation Jacobian matrix analytically, as shown in Fig. 7, we introduce an intermediate model-free variable $\tilde{\mathbf{u}} \in \mathbb{R}^m$, the modal displacements of the base mesh (rather than the object) under robot manipulation. To compute $\tilde{\mathbf{u}}$, we project the robot manipulation on the base mesh to the modal space. Based on Assumption 1, the manipulation points $\mathbf{p}_r$ are fixed object points whose positions can be obtained using robot measurements. Given $\mathbf{p}_r$, following the derivations in Sections III and IV, $\tilde{\mathbf{u}}$ can be computed with

$$\tilde{\mathbf{u}}(t)=(\widetilde{\mathbf{K}}+\mathbf{I}_6)^{-1}[\boldsymbol{\Phi}_n]_r^T\mathbf{N}_r^T(\boldsymbol{\eta}(\mathbf{p}_r),\mathbf{n}_r)(\mathbf{x}(\mathbf{p}_r,t)-\mathbf{x}(\boldsymbol{\eta}(\mathbf{p}_r))) \tag{26}$$

where $\boldsymbol{\eta}(\mathbf{p}_r)$ are the base mesh projections of $\mathbf{p}_r$, whose position vector is $\mathbf{x}(\boldsymbol{\eta}(\mathbf{p}_r)) \in \mathbb{R}^{3k}$; $\mathbf{n}_r$ is the manipulation points' allocation nodes of size h; $\mathbf{N}_r(\boldsymbol{\eta}(\mathbf{p}_r),\mathbf{n}_r) \in \mathbb{R}^{3k\times 3h}$ is the (local) object-mesh allocating matrix for the manipulation points; $[\boldsymbol{\Phi}_n]_r \in \mathbb{R}^{3h\times m}$ consists of the rows of $\mathbf{n}_r$ extracted from $\boldsymbol{\Phi}_n$. Nevertheless, $\tilde{\mathbf{u}}(t)$ only reflects deformation properties of the base mesh while our deformation features $\mathbf{s}(t)$ (computed from the object's stereo measurements) reflect the unknown deformation properties of the object. We must take the modeling uncertainties between the base mesh and the object into consideration.

B. Unknown Modal Parameters

Our method investigates all the modeling uncertainties in the truncated reference modal space. Given (10), the deformation features $\mathbf{s}(t)$ can be regarded as the generalized displacements of the object in the truncated reference modal space. As the modal space is generated using normalized modes, according to [45], [48], the influence of material and mass properties can be absorbed in the amplitudes of modal displacements. To further deal with the unknown geometric model of the object, we assume some unknown but constant virtual loads [see Fig. 6(b)] that deform the base mesh to fit the undeformed object geometry. In this way, according to [48], all the modeling uncertainties can be absorbed in the amplitudes of modal displacements. Then, inspired by [59], we approximate the local elastic relations between $\mathbf{s}(t)$ and $\tilde{\mathbf{u}}(t)$ with the following affine deformation model [60]:

$$\mathbf{s}(t)=\mathbf{Q}\tilde{\mathbf{u}}(t)+\mathbf{o} \tag{27}$$

where the unknown constant matrix $\mathbf{Q} \in \mathbb{R}^{m\times m}$ and the unknown constant vector $\mathbf{o} \in \mathbb{R}^m$ represent deformation properties of the object and the base mesh. Unlike the affine model used in [59] that approximates deformation in Euclidean space, this affine model defined in the modal space is decoupled and low-dimensional. Thus, $\mathbf{Q}$ is a diagonal matrix:

$$\mathbf{Q}(\boldsymbol{\theta})=\mathrm{diag}\begin{bmatrix}\theta_1, & \theta_2, & \dots & \theta_m\end{bmatrix} \tag{28}$$

consisting of a low-dimensional unknown modal parameter vector $\boldsymbol{\theta}=\begin{bmatrix}\theta_1, & \theta_2, & \dots & \theta_m\end{bmatrix}^T \in \mathbb{R}^m$.

Remark 3: Note that the linearization of (23) and the affine deformation model (27) are local quasi-static approximations of the object deformation. Similar techniques are widely used in the existing Jacobian-based deformation controllers (such as [7], [57], [59], [61]), whose aim is not developing an accurate deformation model but approximating how slow manipulation in a small time interval transforms into changes of the deformation features. Under Assumption 3, our method uses the approximations in a closed-loop controller to determine proper combinations of the manipulation velocities that would result in the desired changes of deformation features. We also design adaptive laws to online update the approximations such that their effective ranges can be continuously extended.

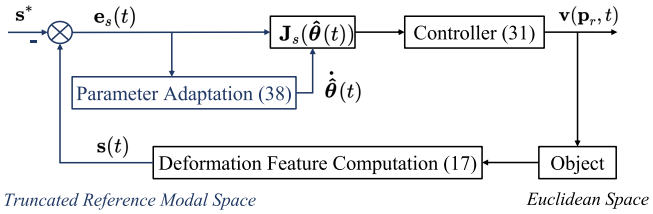


Fig. 8. Block diagram of our deformation controller. We use the dark blue color to present the parts in the truncated reference modal space.

C. Deformation Jacobian Matrix

Based on (26) and (27), we derive the following Jacobian relationship:

$$\begin{aligned}\dot{\mathbf{s}}(t) &= \mathbf{Q}(\boldsymbol{\theta})(\widetilde{\mathbf{K}} + \mathbf{I}_6)^{-1}[\boldsymbol{\Phi}_n]_r^T \mathbf{N}_r^T(\boldsymbol{\eta}(\mathbf{p}_r), \mathbf{n}_r)\dot{\mathbf{x}}(\mathbf{p}_r, t) \\ &= \mathbf{J}_s(\boldsymbol{\theta})\dot{\mathbf{x}}(\mathbf{p}_r, t) \end{aligned} \tag{29}$$

where $\mathbf{J}_s(\boldsymbol{\theta}) \in \mathbb{R}^{m\times 3k}$ is our deformation Jacobian matrix. Unlike some existing works [7], [9], [20] that treat the whole deformation Jacobian matrix as unknown, our Jacobian matrix is analytically formulated using mode shapes and with only low-dimensional unknown parameters.

VI. Adaptive Deformation Controller

This section designs an adaptive deformation controller with online estimation of the unknown Jacobian parameters. The controller's block diagram is shown in Fig. 8.

A. Kinematic Control Law

Given the deformation feature errors computed with

$$\mathbf{e}_s(t) = \mathbf{s}(t) - \mathbf{s}^* \tag{30}$$

we propose the following kinematic control law using the transpose of the online estimated deformation Jacobian matrix:

$$\mathbf{v}(\mathbf{p}_r, t) = -\mathbf{K}_s \mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t) \tag{31}$$

where the linear velocities $\mathbf{v}(\mathbf{p}_r, t)$ are the manipulation commands for $\mathbf{p}_r$; the transposed Jacobian matrix $\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t)) \in \mathbb{R}^{3k\times m}$ is computed using the online estimated parameter $\hat{\boldsymbol{\theta}}(t) \in \mathbb{R}^m$; $\mathbf{K}_s \in \mathbb{R}^{3k\times 3k}$ is a positive definite gain matrix called the feedback gain. Then, we can obtain the closed-loop error dynamics by combining (31), (30), and (29)

$$\begin{aligned}\dot{\mathbf{e}}_s(t) &= \mathbf{J}_s(\boldsymbol{\theta})(-\mathbf{K}_s \mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t)) \\ &= -\mathbf{J}_s(\boldsymbol{\theta})\mathbf{K}_s \mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t). \end{aligned} \tag{32}$$

We here discuss the physical interpretation of our transposed-Jacobian-based controller. Expanding the transpose of the deformation Jacobian matrix, we have

$$\begin{aligned}\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t)) &= \mathbf{N}_r(\boldsymbol{\eta}(\mathbf{p}_r), \mathbf{n}_r)[\boldsymbol{\Phi}_n]_r(\widetilde{\mathbf{K}} + \mathbf{I}_6)^{-T}\mathbf{Q}^T(\hat{\boldsymbol{\theta}}(t)) \\ &= \mathbf{N}_r(\boldsymbol{\eta}(\mathbf{p}_r), \mathbf{n}_r)[\boldsymbol{\Phi}_n]_r(\widetilde{\mathbf{K}} + \mathbf{I}_6)^{-1}\mathbf{Q}(\hat{\boldsymbol{\theta}}(t)) \end{aligned} \tag{33}$$

where $\mathbf{Q}(\hat{\boldsymbol{\theta}}(t))$ and $(\widetilde{\mathbf{K}} + \mathbf{I}_6)^{-1}$ are the estimated parameter matrix and the force-displacement rectified term in the truncated reference modal space; $\mathbf{N}_r(\boldsymbol{\eta}(\mathbf{p}_r), \mathbf{n}_r)$ represents the object-mesh mappings for the robot manipulation; as $det(\boldsymbol{\Phi}_n) \neq 0$, the submatrix $[\boldsymbol{\Phi}_n]_r$ relates a small change of the displacements of the manipulated nodes to a small change of the intermediate model-free variable $\widetilde{u}(t)$. Thus, the term

$$\mathbf{N}_r(\boldsymbol{\eta}(\mathbf{p}_r), \mathbf{n}_r)[\boldsymbol{\Phi}_n]_r \in \mathbb{R}^{3k\times m} \tag{34}$$

provides a deformation Jacobian relationship for the base mesh manipulation, whose direction is determined by the object-mesh mapping and the low-frequency modes of the base mesh. As discussed in [47], low-frequency modes are mainly determined by the low-order moments of inertia, thus compact bodies of similar sizes have similar low-frequency modes. In this way, if the selected base mesh is of similar size to the object, $[\boldsymbol{\Phi}_n]_r$ provides mass-scaled similar low-frequency modes of the object's low-frequency modes. We can thus regard $\mathbf{N}_r(\boldsymbol{\eta}(\mathbf{p}_r), \mathbf{n}_r)[\boldsymbol{\Phi}_n]_r$ as a reference direction of the object's deformation Jacobian matrix and simply initialize the unknown modal parameters to be ones (indicating no *a priori* knowledge). By comparison, some existing works [7], [9], [20] need an extra testing stage to initialize the whole deformation Jacobian matrix.

B. Online Parameter Estimation

Our parameter updating law is formulated by compensating for the estimation errors in the closed-loop system online. To do that, we first add a zero term to the closed-loop error dynamics (32) and rewrite the equation as

$$\begin{aligned}\dot{\mathbf{e}}_s(t) &= [-\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t)) + (\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t)) - \mathbf{J}_s(\boldsymbol{\theta}))]\mathbf{K}_s \mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t) \\ &= -\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t))\mathbf{K}_s \mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t) \\ &\quad + (\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t)) - \mathbf{J}_s(\boldsymbol{\theta}))\mathbf{K}_s \mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t). \end{aligned} \tag{35}$$

Then, we define the following parameter errors $\Delta\boldsymbol{\theta}(t) \in \mathbb{R}^m$ with

$$\Delta\boldsymbol{\theta}(t) = \hat{\boldsymbol{\theta}}(t) - \boldsymbol{\theta} \tag{36}$$

which can be used to linearize the estimation errors in the closed-loop error dynamics with

$$\begin{aligned}&(\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t)) - \mathbf{J}_s(\boldsymbol{\theta}))\mathbf{K}_s \mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t) = (\mathbf{Q}(\hat{\boldsymbol{\theta}}(t)) - \mathbf{Q}(\boldsymbol{\theta}))\cdot \\ &(\widetilde{\mathbf{K}} + \mathbf{I}_6)^{-1}[\boldsymbol{\Phi}_n]_r^T \mathbf{N}_r^T(\boldsymbol{\eta}(\mathbf{p}_r), \mathbf{n}_r)\mathbf{K}_s \mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t) \\ &= \mathbf{Y}(\mathbf{e}_s(t), \hat{\boldsymbol{\theta}}(t))\Delta\boldsymbol{\theta}(t) \end{aligned} \tag{37}$$

where the regression matrix $\mathbf{Y}(\mathbf{e}_s(t), \hat{\boldsymbol{\theta}}(t)) \in \mathbb{R}^{m\times m}$ is independent of the unknown parameters $\boldsymbol{\theta}$. Afterward, according to the Slotine-Li algorithm [62], we propose the following online parameter updating law:

$$\dot{\hat{\boldsymbol{\theta}}}(t) = -\Gamma^{-1}\mathbf{Y}^T(\mathbf{e}_s(t), \hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t) \tag{38}$$

where Γ is a positive definite scalar called the parameter updating gain.

Remark 4: The linear parametrization of the closed-loop error dynamics enables us to design the adaptive control laws [the (31) and (38)] with guaranteed stability (the detailed proof will be given in the next section). The objective of (38) is

updating the parameters to minimize the deformation feature errors $\mathbf{e}_s(t)$ rather than identifying the true parameter values that can fit some input–output measurements of the system. Thus, unlike some existing shape deformation controllers (such as [8], [9], [11], [20]) that need to learn or estimate the deformation models/parameters accurately, our controller is stable without requiring true parameter identification.

C. Stability Analysis

In this section, we analyze the stability of our deformation controller under the proposed adaptive laws.

Theorem 1: Under the control of the controller (31) and the adaptive algorithm (38) for parameter estimation, the deformation feature errors are convergent in the following way:

$$\lim_{t\to\infty} \mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t) = \mathbf{0}. \tag{39}$$

Proof: We introduce the following Lyapunov candidate function:

$$V(t) = \frac{1}{2}\mathbf{e}_s^T(t)\mathbf{e}_s(t) + \frac{\Gamma}{2}\Delta\boldsymbol{\theta}^T(t)\Delta\boldsymbol{\theta}(t). \tag{40}$$

Differentiating (40) with respect to time, we obtain

$$\begin{aligned}\dot{V}(t) &= \mathbf{e}_s^T(t)\dot{\mathbf{e}}_s(t) + \Gamma\Delta\dot{\boldsymbol{\theta}}^T(t)\Delta\boldsymbol{\theta}(t)\\ &= \mathbf{e}_s^T(t)\dot{\mathbf{e}}_s(t) + \Gamma\dot{\hat{\boldsymbol{\theta}}}^T(t)\Delta\boldsymbol{\theta}(t).\end{aligned} \tag{41}$$

Substituting the closed-loop error dynamics (35), we obtain

$$\begin{aligned}\dot{V}(t) = &-\mathbf{e}_s^T(t)\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t))\mathbf{K}_s\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t)\\ &+\mathbf{e}_s^T(t)(\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t)) - \mathbf{J}_s(\boldsymbol{\theta}))\mathbf{K}_s\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t)\\ &+\Gamma\dot{\hat{\boldsymbol{\theta}}}^T(t)\Delta\boldsymbol{\theta}(t).\end{aligned} \tag{42}$$

Then injecting (38) into it, we have

$$\begin{aligned}\dot{V}(t) = &-\mathbf{e}_s^T(t)\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t))\mathbf{K}_s\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t)\\ &+\mathbf{e}_s^T(t)(\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t)) - \mathbf{J}_s(\boldsymbol{\theta}))\mathbf{K}_s\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t)\\ &-\mathbf{e}_s^T(t)\mathbf{Y}(\mathbf{e}_s(t),\hat{\boldsymbol{\theta}}(t))\Delta\boldsymbol{\theta}(t).\end{aligned} \tag{43}$$

Further injecting (37), we obtain

$$\begin{aligned}\dot{V}(t) = &-\mathbf{e}_s^T(t)\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t))\mathbf{K}_s\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t)\\ &+\mathbf{e}_s^T(t)(\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t)) - \mathbf{J}_s(\boldsymbol{\theta}))\mathbf{K}_s\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t)\\ &-\mathbf{e}_s^T(t)(\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t)) - \mathbf{J}_s(\boldsymbol{\theta}))\mathbf{K}_s\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t).\end{aligned} \tag{44}$$

Thus

$$\dot{V}(t) = -\mathbf{e}_s^T(t)\mathbf{J}_s(\hat{\boldsymbol{\theta}}(t))\mathbf{K}_s\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t). \tag{45}$$

It is obvious that $\dot{V}(t) \le 0$, and hence $V(t) \le V(0)$ (i.e., $V(t)$ is upper bounded), which means that $\mathbf{e}_s(t)$ and $\hat{\boldsymbol{\theta}}(t)$ are bounded. Under our quasi-static deformation assumption, $\dot{\mathbf{e}}_s(t)$ and $\dot{\hat{\boldsymbol{\theta}}}(t)$ are also bounded, which further leads to that $\dot{V}(t)$ is uniformly continuous. Thus, according to barbalat's lemma [63], we can conclude that $\lim_{t\to\infty}\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))\mathbf{e}_s(t) = \mathbf{0}$.

Therefore, with the proposed controller, the deformation feature errors $\mathbf{e}_s(t)$ will be attracted to the invariant set in the null space $Ker(\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t)))$ as $t\to\infty$. If $m = 3\,k$, $\mathbf{e}_s(t)$ will converge to zeros. If $m > 3\,k$ (the case for most robotic tasks of shape deformation control), the control problem is overdetermined. Then, the nontrivial null space of $\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t))$ will give rise to local minima configurations, where the vector $\mathbf{e}_s(t)$ converges to a steady-state error. It is well known that the overdetermined controller may lead to local minima. More information and strategies about the overdetermined deformation controller and visual servoing controller can be found in [9] and [64], respectively. ■

Remark 5: Existence of the nontrivial $Ker(\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t)))$ does not mean that local minima always exist. According to [64], potential local minima are also due to the existence of unrealizable feature motion $\dot{\mathbf{s}}^{\perp}(t)$ that does not belong to the range space of $\mathbf{J}_s(\boldsymbol{\theta})$:

$$\dot{\mathbf{s}}^{\perp}(t) \in [Im(\mathbf{J}_s(\boldsymbol{\theta}))]^{\perp} = Ker(\mathbf{J}_s^T(\boldsymbol{\theta})). \tag{46}$$

On the other hand, potential local minima configurations

$$(\mathbf{s}(t) - \mathbf{s}^*) \in Ker(\mathbf{J}_s^T(\hat{\boldsymbol{\theta}}(t)) \cap Ker(\mathbf{J}_s^T(\boldsymbol{\theta})) \tag{47}$$

must be physically coherent, which means that the corresponding robot manipulation poses must exist under all the object's physical constraints including the manipulation constraints and the other unknown boundary conditions.

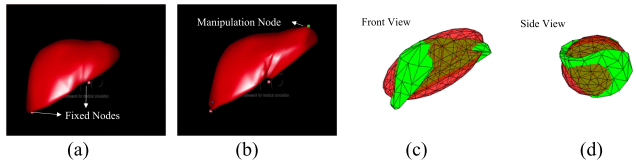


Fig. 9. Simulation setup: (a) undeformed configuration of the liver-shaped object, where the pink nodes denote the fixed nodes, (b) manipulating the object by changing positions of the manipulation node (the green node), (c), (d) comparisons of the base mesh (the red mesh) and the deformation model of the object (the green mesh).

VII. Simulation Analysis

A. Simulation Setup

Simulation validations are conducted based on the Simulation Open Framework Architecture (SOFA) [65] platform within a liver-shaped object scene [66]. In the SOFA scene, the object has a visual model [the red liver surface in Fig. 9(a) and (b)] and a deformation model [the green mesh in Fig. 9(c) and (d)]. We can modify the physical properties of the deformation model and directly read or change the positions of its nodes with specific indices. The object can be fixed in the simulation space by adding zero-displacement constraints to some fixed nodes (the pink nodes in Figs. 9 and 10) on the deformation model. We also set manipulation nodes (the green nodes in Figs. 9 and 10) and sampled nodes (the blue nodes in Fig. 10) on the deformation model to deform the object and to obtain surface samplings, respectively. The base mesh [the red mesh in Fig. 9(c) and (d)] is generated using the moment-based method proposed in [44] with the same unit system in the SOFA scene. Fig. 9(c) and

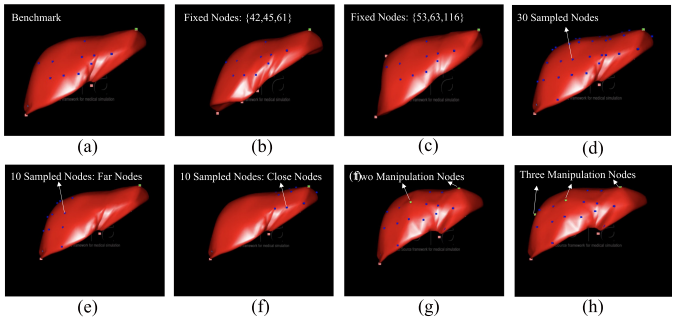


Fig. 10. Desired configurations of different cases: (a) the benchmark case, (b) the case with fixed node indices of $\{42, 45, 61\}$, (c) the case with fixed node indices of $\{53, 63, 116\}$, (d) the case with 30 sampled nodes (the blue nodes), (e) the far node case with the 10 sampled nodes far from the manipulation node, (f) the close node case with the 10 sampled nodes close to the manipulation node, (g) the case with two manipulation nodes, (h) the case with three manipulation nodes.

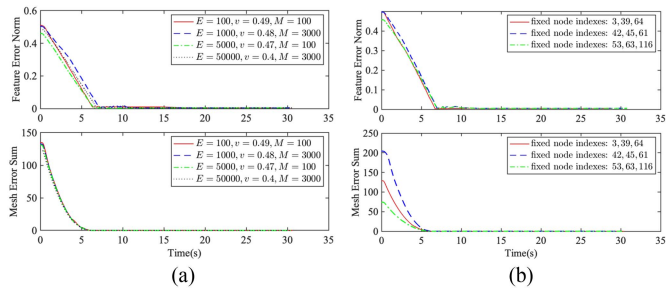


Fig. 11. Simulation results with different: (a) material and mass parameters; (b) boundary conditions.

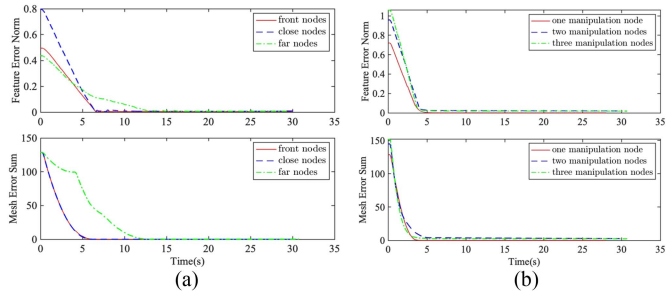


Fig. 12. Simulation results with: (a) different sampled node distributions; (b) multiple manipulation nodes.

(d) shows the comparisons between the deformation model of the object and the base mesh used for control. The material properties of the base mesh are set to be: $E = 50, v = 0.45$, and $M = 20$. The control gains are set to be: $\mathbf{K}_s = 80\mathbf{I}_{m\times m}$ and $\Gamma = 500$. The frequency of the simulation is 50 Hz.

In our simulations, we first set a benchmark case [see Fig. 10(a)] with the following conditions: the object's material and mass parameters are: $E = 100, v = 0.49, M = 100$; the fixed node indices are: $\{3, 39, 64\}$; desired deformations are generated by manipulating one node (of the index 85) with the displacement vector of $\mathbf{u}(\mathbf{p}_r)^* = (1, 1, 0.8)^T$ (unit: voxel of the simulation scene); the sampled nodes are selected on the front side of the object; the dimension of the deformation features is $m = 30$. Then, for each group of simulations, we separately changed the object's modeling parameters, the boundary conditions, the sampled node distributions, the manipulation node numbers, and the deformation feature dimensions for controlled validations. For different simulations, we developed deformation controllers using the same base mesh with fixed physical properties and pose parameters.

To analyze the convergence of our deformation features, we recorded the following feature error norm:

$$\|\mathbf{e}_s(t)\| = \sqrt{\mathbf{e}_s^T(t)\mathbf{e}_s(t)}. \tag{48}$$

Also, to analyze the convergence of the object's 3-D shape, we recorded the following total mesh error sum:

$$e_x(t) = (\mathbf{x}(\mathbf{n}_o, t) - \mathbf{x}^*(\mathbf{n}_o))^T(\mathbf{x}(\mathbf{n}_o, t) - \mathbf{x}^*(\mathbf{n}_o)) \tag{49}$$

where $\mathbf{x}(\mathbf{n}_o, t)$ is the position vector of the object's deformation model nodes $\mathbf{n}_o$ and $\mathbf{x}^*(\mathbf{n}_o)$ is the desired position vector.

B. Simulation Cases

1) Simulations With Different Material and Mass Parameters: We conducted a group of simulation cases with a large range of material and mass parameter changes of the object: $E = 100, v = 0.49, M = 100$ [the benchmark case in Fig. 10(a)]; $E = 1000, v = 0.48, M = 3000$; $E = 5000, v = 0.47, M = 100$; $E = 50\,000, v = 0.4, M = 3000$. Simulation results in Fig. 11(a) show the minimization of the magnitude errors $\|\mathbf{e}_s(t)\|$ and $e_x(t)$. The start points of the curves for $\|\mathbf{e}_s(t)\|$ and $e_x(t)$ are different because: the same manipulation on objects with different material and mass parameters produced different deformation feature values.

2) Simulations With Different Boundary Conditions: We conducted a group of simulation cases with different fixed node indices: $\{3, 39, 64\}$ [the benchmark case in Fig. 10(a)]; $\{42, 45, 61\}$ [see Fig. 10(b)]; $\{53, 63, 116\}$ [see Fig. 10(c)]. Simulation results in Fig. 11(b) show the minimization of the magnitude errors $\|\mathbf{e}_s(t)\|$ and $e_x(t)$. The differences of start points of the mesh error sum curves are much larger than the feature error norm curves. This is because the local deformations of the sampled nodes were less different than the object's global deformation. It further implies that our method can control the 3-D shape globally even with locally sampled nodes.

3) Simulations With Different Sampled Node Distributions: To validate our method with different distributions of the sampled nodes, we conducted three cases: the front node case [the benchmark case in Fig. 10(a)], the close node case [see Fig. 10(f)], and the far node case [see Fig. 10(e)]. The results in Fig. 12(a) show the minimization of the magnitude errors $\|\mathbf{e}_s(t)\|$ and $e_x(t)$. The start points of the feature error norm curves are different because: despite the same manipulation, different sampled nodes underwent different (local) deformations producing different deformation feature values.

In the far node case, the sampled nodes were distributed in a local region far from the manipulation nodes, but close to the constrained nodes. Their desired deformation was more complex and different than the desired manipulation, which

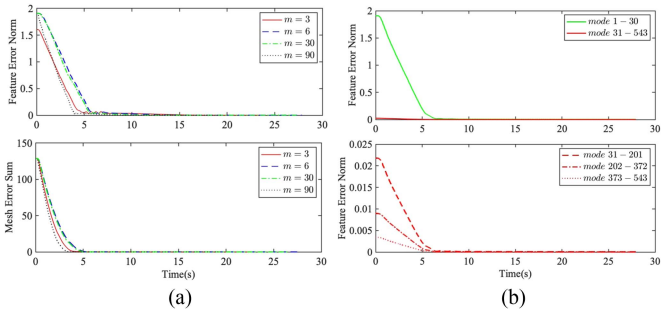


Fig. 13. Simulation results for: (a) different deformation feature dimensions; (b) the controlled and the nondirectly actuated modes.

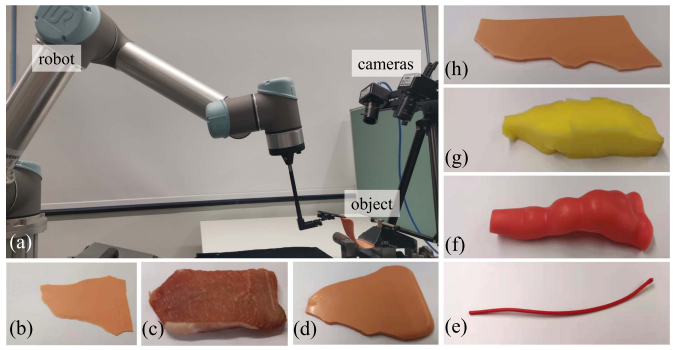


Fig. 14. Experiment setup and the experimental objects. (a): the robot-camera platform; (b) the deformable planar object (a piece of silicon skin); (c) the pork tissue block; (d) the short silicon block; (e) the deformable linear object (a piece of red electric wire); (f) the silicon colon model; (g) the sponge block; (h) the long silicon block.

affected the initial manipulation directions. However, our controller kept adjusting the manipulation command to minimize $\mathbf{e}_s(t)$. The turning points on the result curves happened when some of the dimensions of the manipulation velocity command changed between positive and negative. We further conclude that the sampled node distributions that are more global or close to the manipulation node produce better initial manipulation directions.

4) Simulations With Multiple Manipulation Nodes: To validate our method with multiple manipulation nodes, we conducted three simulation cases: the case with one manipulation node [the benchmark case in Fig. 10(a)]; the case with two manipulation nodes [see Fig. 10(g)]; the case with three manipulation nodes [Fig. 10(h)]. Results in Fig. 12(b) show the minimization of the magnitude errors $\|\mathbf{e}_s(t)\|$ and $e_x(t)$.

5) Simulations With Different Deformation Feature Dimensions: We conducted a group of simulation cases with different deformation feature dimensions: $m = 3$, $m = 6$, $m = 30$, and $m = 90$. As our method requires that $3n \geq m$, we increased the number of sampled nodes to 30 [see Fig. 10(d)] in these cases. Results in Fig. 13(a) show the minimization of the magnitude errors $\|\mathbf{e}_s(t)\|$ and $e_x(t)$ with different m.

6) Behaviors of Nondirectly Actuated Modes: We also analyze the behavior of the high-frequency modes that were not directly actuated in the simulation case with $m = 30$. As we can measure 181 nodes from the object's deformation model, the amplitudes of all the 543 modes were computed and recorded during the control process. We compare the behaviors of the modes used in our controller (the 1st–30th modes) and the nondirectly actuated modes (the 31th–543th modes) in Fig. 13(b). The results show that: 1) amplitudes of the nondirectly actuated modes are much smaller than the controlled modes; 2) the nondirectly actuated modes are similarly minimized as the controlled modes. The reasons are: modal amplitudes are inversely proportional to the square of their natural frequency [44]; even though not directly actuated, the high-frequency modes converge toward the desired values as the total mesh error sum $e_x(t)$ being minimized.

Algorithm 1: Controller($\mathbf{x}^*(\mathbf{p}_s)$, a_z, m, E, v, M, $\mathbf{K}_s$, Γ).

```
while receiving vision and robot measurements do
  measure x(p_s, t), x(p_r, t), ^c R_e(t)
  if t = t_0 then
    generate the base mesh ← ((50), (51), and (52)), a_z
    compute Φ_n and K̃ ← m, E, v, M
    establish the object-mesh mappings of p_s and p_r
    compute s* ← (12) and (17), x*(p_s)
    initialize θ(t_0) with ones
  end if
  if t > t_0 then
    compute s(t) ← (12) and (17), x(p_s, t)
    while robot is running do
      compute e_s(t) ← (30)
      update θ̂(t) ← (38), Γ
      compute J_s(θ̂(t)) ← (29)
      compute v(p_r, t) ← (31), K_s
      send v(p_r, t) to robot
    end while
  end if
end while
```

VIII. Experimental Validation

A. Experiment Setup

Our experiment platform consists of a Universal Robot 5 and two cameras. Fig. 14 shows the experiment setup, where one end of the object is fixed in the space while the other end is rigidly grasped by a slim clip mounted on the robot end-effector. Fig. 14(b)–(h) shows the deformable objects used in experiments. For different objects, their base meshes used for control are set with the same material and mass parameters ($E = 1$ GPa, $v = 0.4$, $M = 0.05$ kg). The control gains are set to be: $\mathbf{K}_s = 0.1\mathbf{I}_{m\times m}$ and $\Gamma = 0.1$. The frequency of the experiments is 30 Hz.

In advance of the control phase, we calibrated the stereo cameras, the hand-eye relationship (the transformation matrix ${}^r\mathbf{T}_c$), and the hand-grasper relationship (the position vector ${}^e\mathbf{x}(\mathbf{p}_r)$ of the grasped point under the end-effector frame $\{e\}$). Desired deformations in our experiments were generated by manually manipulating the object to target configurations. During the control phase, we tracked some image features of the object to obtain surface samplings. For base mesh generation, at t_0, and under some arbitrary configuration, we selected four points

among the surface samplings: $\mathbf{p}_0/\mathbf{p}_1$ with the smallest/largest x coordinate and the smallest y coordinate; $\mathbf{p}_2/\mathbf{p}_3$ with the smallest/largest x coordinate and the largest y coordinate. We computed the base mesh's size parameters $\{a_x, a_y\}$ with

$$a_x = \frac{\|\mathbf{x}(\mathbf{p}_1, t_0) - \mathbf{x}(\mathbf{p}_0, t_0)\| + \|\mathbf{x}(\mathbf{p}_3, t_0) - \mathbf{x}(\mathbf{p}_2, t_0)\|}{4}$$

$$a_y = \frac{1}{2}\left\| \frac{\mathbf{x}(\mathbf{p}_0, t_0) + \mathbf{x}(\mathbf{p}_1, t_0)}{2} - \frac{\mathbf{x}(\mathbf{p}_2, t_0) + \mathbf{x}(\mathbf{p}_3, t_0)}{2} \right\|. \tag{50}$$

We roughly estimated the object's thickness to set the size parameter a_z. Then, to determine the base mesh pose ${}^c\mathbf{T}_b$, we computed $\mathbf{c}(\mathbf{x}(\mathbf{p}_s, t_0)) \in \mathbb{R}^3$ (the mass center of the surface samplings) and measured ${}^c\mathbf{R}_e(t_0)$ (the orientation of the end-effector) to set

$${}^c\mathbf{T}_b = \begin{bmatrix} {}^c\mathbf{R}_e(t_0) & {}^c\mathbf{t}_b(t_0) \\ \mathbf{0} & 1 \end{bmatrix} \tag{51}$$

$${}^c\mathbf{t}_b(t_0) = \mathbf{c}(\mathbf{x}(\mathbf{p}_s, t_0)) - a_z \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T. \tag{52}$$

When the robot was running, we sent the deformation feature errors to the controller, which computed linear velocity commands for the robot manipulation. After transforming the velocity commands to the joint space of the robot, we directly set the joint velocities to each of the robot joints via the low-level interface provided by URScript [67]. Algorithm 1 shows the implementation of our deformation controller.

To analyze the convergence of our deformation features, we recorded the feature error norm $\|\mathbf{e}_s(t)\|$. In addition, to analyze the convergence of the object's 3-D shape, we recorded the following manipulation errors:

$$\mathbf{e}_d(t) = \mathbf{x}(\mathbf{p}_r, t) - \mathbf{x}^*(\mathbf{p}_r) \tag{53}$$

where $\mathbf{x}(\mathbf{p}_r, t)$ is the measured position vector of the manipulation point $\mathbf{p}_r$ and $\mathbf{x}^*(\mathbf{p}_r)$ is the target position vector of $\mathbf{p}_r$. It should be mentioned that we only recorded $\mathbf{x}^*(\mathbf{p}_r)$ for result analysis. We did not use it for robot control. We also gave 2-D image comparisons between the object's resulting and target shape by adding the desired image as a translucent mask on the resulting image.

B. Validation Cases

1) Validations With Different Kinds of Objects: To validate that our method can deal with different types of objects and the tracked image features, we conducted experimental cases using linear, planar, and volumetric objects under stereo measurements of points, curves, and contours. According to [44], [47], using 30 lowest frequency deformation modes, a wide range of nonrigid shapes can be described. Aiming at the global 3-D deformation control, we set the deformation feature dimensions to be: $m = 30 \sim 50$.

The linear object case [see Fig. 15(a)] used a piece of red wire. The desired deformation was generated by pushing the wire forward while lifting it. We obtained surface samplings from the wire's curve tracked by color segmentation. We set the deformation feature dimension to be: $m = 30$, and estimated the wire's thickness to be: $a_z = 0.001(m)$. The planar object case [see Fig. 15(b)] used a piece of silicon skin. The desired deformation was generated by pulling the silicon skin to left. We obtained surface samplings from the silicon skin's contour tracked by color segmentation. We selected the deformation feature dimension to be: $m = 40$, and estimated the silicon skin's thickness to be: $a_z = 0.0005(m)$. The volumetric object case [see Fig. 15(c)] used a pork tissue block. The desired deformation was generated by lifting the tissue to right. We obtained surface samplings by tracking points on the tissue surface. We selected the deformation feature dimension to be: $m = 30$, and estimated the tissue's thickness to be: $a_z = 0.0125(m)$.

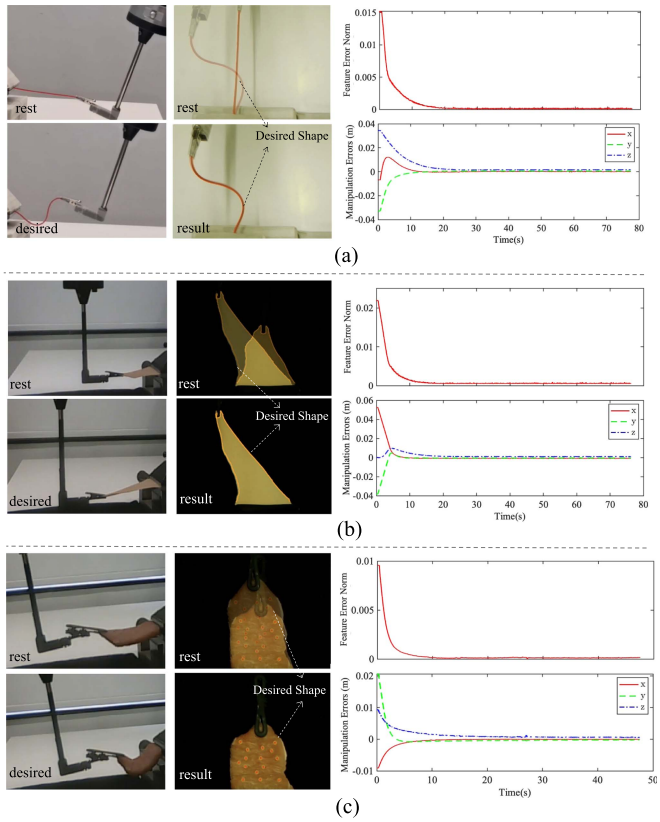


Fig. 15. Cases with different kinds of objects: the first/second column shows the third/left camera views; the third column shows the result curves of the feature error norm ($\|\mathbf{e}_s(t)\|$) and the manipulation errors ($\mathbf{e}_d(t)$). (a) for the linear object; (b) for the planar object; (c) for the volumetric tissue.

Result curves in Fig. 15 show the minimization of both $\|\mathbf{e}_s(t)\|$ and $\mathbf{e}_d(t)$ for different cases. Manipulation error curves present few steady-state errors (less than 3 mm) because modal-based dimension reductions neglect some details of the object's shape deformation. In addition, from the resulting images of the left camera views, we can see that the objects are manipulated to fit the desired shapes in a general sense.

2) Validations With Base Mesh Size Errors: To validate our method with base mesh size errors, we conducted experimental cases using the same object [the short silicon block in Fig. 14(d)]

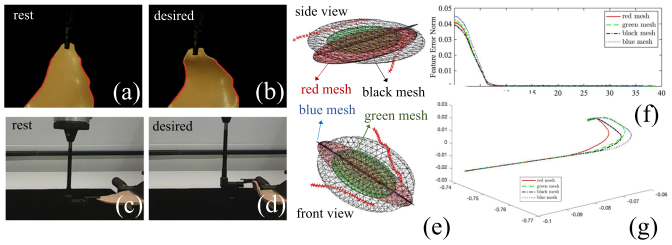


Fig. 16. Cases with different base mesh size errors. (a)/(b) the left camera views of the rest/desired configuration, where the red dots denote the surface samplings; (c)/(d) the third camera views of the rest/desired configuration; (e) the illustrations of the different-sized meshes with respect to the surface samplings (the red cross symbol) under the rest configuration; (f) the result curves of $\|\mathbf{e}_s(t)\|$; (e) the manipulation trajectories (of $\mathbf{x}(\mathbf{p}_r, t)$), where the green star symbol denotes the target manipulation position ($\mathbf{x}^*(\mathbf{p}_r)$).

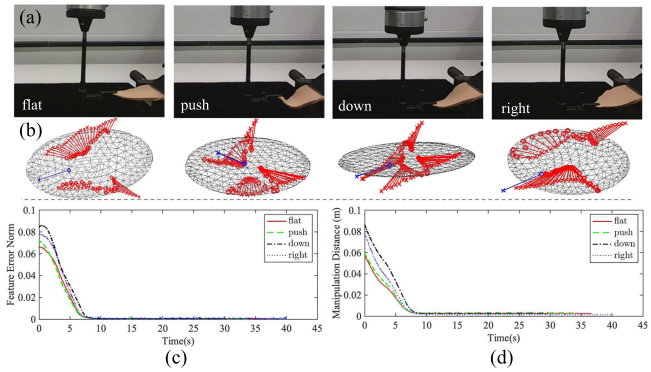


Fig. 17. Cases with different rest configurations. (a) the third camera views; (b) surface samplings/manipulation points (the red/blue cross symbol) connecting (through red/blue lines) with their base mesh projection (the red/blue circles); (c) the result curves of $\|\mathbf{e}_s(t)\|$; (d) the result curves of the manipulation distance ($\|\mathbf{e}_d(t)\|$).

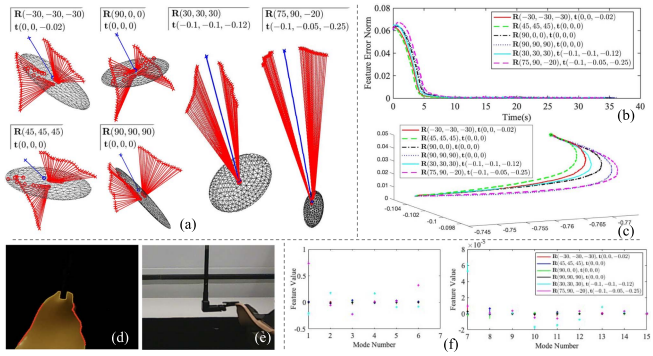


Fig. 18. Cases with base mesh pose errors. (a) Surface samplings and manipulation points connecting with their base mesh projections under the rest configuration. (b) The result curves of $\|\mathbf{e}_s(t)\|$; (c) The manipulation trajectories (of $\mathbf{x}(\mathbf{p}_r, t)$), where the green star symbol denotes the target manipulation position ($\mathbf{x}^*(\mathbf{p}_r)$). (d), (e) Left/third camera views of the desired configuration. (f) Deformation feature values of 1st–6th/7th–15th mode. The star symbols denote the desired feature values, which are connected by solid lines from the corresponding rest feature values.

with different-sized base meshes. We set the same desired deformation from the rest configuration in Fig. 16(c) to the desired configuration in Fig. 16(d). We obtained surface samplings [the red dots in Fig. 16(a) and (b)] from the object's contour tracked by color segmentation. We selected four base meshes [see Fig. 16(e)] with different size parameters (unit:m): the red mesh with $\{a_x, a_y, a_z\} = \{0.015, 0.05, 0.0075\}$; the green mesh with $\{a_x, a_y, a_z\} = \{0.0175, 0.025, 0.001\}$; the black mesh with $\{a_x, a_y, a_z\} = \{0.03, 0.045, 0.0025\}$; the blue mesh with $\{a_x, a_y, a_z\} = \{0.001, 0.055, 0.001\}$. We set the same base mesh pose according to [(51) and (52)]. The deformation feature dimension was set to be: $m = 15$.

Results in Fig. 16(f) show the minimization of $\|\mathbf{e}_s(t)\|$ for all cases. Also, as we can see from Fig. 16(g), different base mesh sizes do influence the deformation processes (illustrated by the manipulation trajectories), but the target manipulation position can always be reached. The more similar the size of the base mesh is to the object, the better the manipulation trajectory. On the other hand, the initial directions of these manipulation trajectories are similar. This is because: even projected to different base meshes, the primary directions of the manipulation point to fit the same desired deformation were similar.

3) Validations With Different Rest Configurations: As we compute the base mesh pose using rest measurements of surface samplings, we further validate our method with different rest configurations. We selected the same object [the short silicon block in Fig. 14(d)] but set four different rest configurations [see Fig. 17(a)]: the flat case (with the object being in a flat shape); the push case (with the object being pushed forward); the down case (with the object being put down); the right case (with the object being pulled to right). These different rest configurations led to different base mesh poses [see Fig. 17(b)]. Desired deformations in these cases were generated from their rest configurations to the same desired configuration [see Fig. 18(e)]. For different cases, we selected the same base mesh (the black mesh in Fig. 16). The deformation feature dimension was set to be: $m = 15$. Results in Fig. 17(c) and (d) show the minimization of both $\|\mathbf{e}_s(t)\|$ and $\|\mathbf{e}_d(t)\|$.

4) Validations With Base Mesh Pose Errors: To validate our method with large pose errors between the base mesh and the object, we added an extra transformation to the roughly estimated base mesh pose

$$^c\mathbf{T}_b = \begin{bmatrix} ^c\mathbf{R}_e(t_0) & ^c\mathbf{t}_b(t_0) \\ \mathbf{0} & 1 \end{bmatrix} \cdot \mathbf{T}_{add}. \tag{54}$$

The extra transformation was formulated as

$$\mathbf{T}_{add} = \begin{bmatrix} \mathbf{R}(\alpha_x, \alpha_y, \alpha_z) & \mathbf{t}(\xi_x, \xi_y, \xi_z) \\ \mathbf{0} & 1 \end{bmatrix} \tag{55}$$

where $\mathbf{R}(\alpha_x, \alpha_y, \alpha_z) \in \mathbb{R}^{3\times3}$ is a rotation matrix with $\{\alpha_i, i = x, y, z\}$ denoting the rotation angle (unit: degree) about the ith axis; $\mathbf{t}(\xi_x, \xi_y, \xi_z) \in \mathbb{R}^3$ is a translation vector with $\{\xi_i, i = x, y, z\}$ denoting the displacement component (unit: m) along the ith axis. We set the same object [the short silicon block in Fig. 14(d)] and the same base mesh (the black mesh in Fig. 16). As shown in Fig. 18(a), we set six cases: the case p1 with $\mathbf{R}(-30, -30, -30)$ and $\mathbf{t}(0, 0, -0.02)$; the case p2 with $\mathbf{R}(45, 45, 45)$ and $\mathbf{t}(0, 0, 0)$; the case p3 with $\mathbf{R}(90, 0, 0)$ and $\mathbf{t}(0, 0, 0)$; the case p4 with $\mathbf{R}(90, 90, 90)$ and $\mathbf{t}(0, 0, 0)$; the case

p5 with $\mathbf{R}(30, 30, 30)$ and $\mathbf{t}(-0.1, -0.1, -0.12)$; the case p6 with $\mathbf{R}(75, 90, -20)$ and $\mathbf{t}(-0.1, -0.05, -0.25)$. For different cases, we generated the desired deformation from the flat case in Fig. 17(a) to the desired configuration in Fig. 18(e). The deformation feature dimension was set to be: $m = 15$.

Results in Fig. 18(b) show the minimization of $\|\mathbf{e}_s(t)\|$ for all cases. Also, as we can see from Fig. 18(c), different base mesh poses do influence the deformation processes (illustrated by the manipulation trajectories), but the target manipulation position can always be reached. In Fig. 18(b), the feature error norm curve of the case p6 presents an increasing trend at the beginning stage, which implies unsatisfactory initial manipulation commands. The reason is: the pose errors were so large that the base mesh projection of the manipulation point was among the projections of the surface samplings. Nevertheless, our controller still can achieve the minimization of the deformation feature errors. It should be noted that the initial values of the deformation feature errors [the curves' start points in Fig. 18(b)] of different cases are similar even with the large differences of their base mesh poses. This is because: the object-mesh pose errors, which can be regarded as rigid motions absorbed in the first six modes, mainly influence the (rest and desired) deformation feature values of the first six modes rather than the deformation feature errors. It can be seen from Fig. 18(f) that: feature values of the first six modes are much larger than the other modes; feature error values (the length of solid lines) are much smaller than the desired feature values (the star symbols' positions) in the first six modes.

5) Validations With Complex-Shaped Objects of Similar Sizes: As discussed in Section VI, different-shaped objects with similar sizes have similar low-frequency modes, which leads to different behaviors in Euclidean space but similar behaviors in the truncated modal space. To validate our method for objects with different complex shapes but similar sizes, we conducted experiments using controllers with the same base mesh. We selected an irregular-shaped sponge block [see Fig. 14(g)] and a silicon colon model [see Fig. 14(f)]. We set the black mesh in Fig. 16 as the base mesh, and obtained surface samplings from the objects' contours tracked by color segmentation. We set the same kind of desired deformation [see Fig. 19]: lifting the object to the left while pushing it forward. The deformation feature dimension was set to be: $m = 15$.

Results in Fig. 19 show the minimization of both $\|\mathbf{e}_s(t)\|$ and $\mathbf{e}_d(t)$ for different cases. Manipulation error curves present few steady-state errors (less than 2 mm) because modal-based dimension reductions neglect some details of the object's shape deformation. In addition, from the resulting images of the left camera views, we can see that the objects are manipulated to fit the desired shapes in a general sense.

6) Validation With Large Deformation: We also validated our method with large deformation. It should be mentioned that the large desired deformation causes large initial deformation feature errors and fast initial manipulation commands, thus we need to slow down the parameter updating speed to avoid oscillations at the beginning of the control phase. In the large deformation case, we adjusted the parameter updating gain to be: $\Gamma = 100$. We selected the long silicon block [see Fig. 14(h)] as the object and the black mesh in Fig. 16 as the base mesh. We obtained surface samplings from the objects' contours tracked by color segmentation. The deformation feature dimension was set to be: $m = 50$. To further analyze the control performance with large deformation, besides the deformation feature errors [see Fig. 20(c)] and the manipulation trajectory [see Fig. 20(d)], we also plotted the velocity commands [see Fig. 20(e)] and the computational speed measurements [see Fig. 20(f)] in the joint space. The results show that: the deformation feature errors are minimized; the target manipulation position is reached; the manipulation is smooth in the joint space.

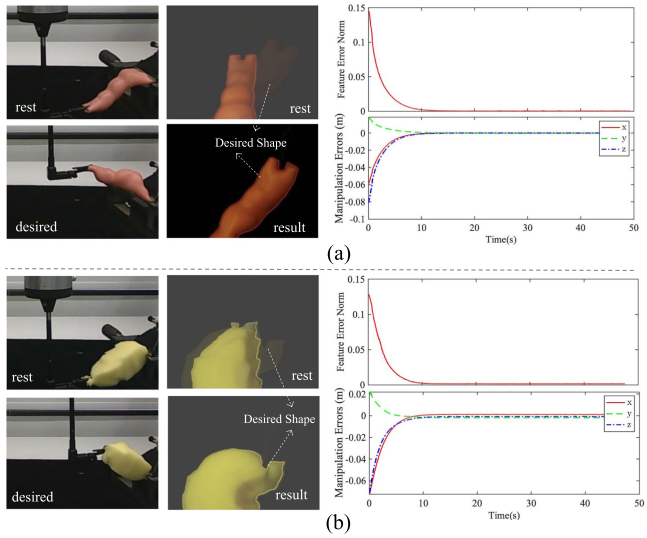


Fig. 19. Cases with complex-shaped objects of similar sizes: the first/second column shows the third/left camera views; the third column shows the result curves of $\|\mathbf{e}_s(t)\|$ and $\mathbf{e}_d(t)$. (a) For the sponge block. (b) For the silicon colon model.

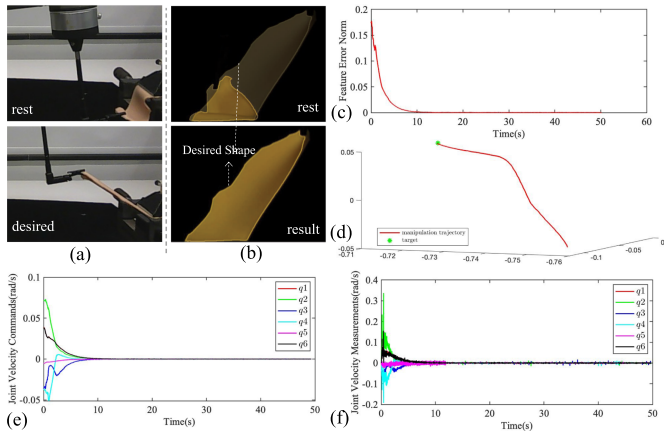


Fig. 20. Case with large deformation. (a/b) Third/left camera view of the rest and desired configurations. (c) Result curve of $\|\mathbf{e}_s(t)\|$, where the small bulges are due to contour detection noises. (d) Manipulation trajectory (of $\mathbf{x}(\mathbf{p}_r, t)$), where the green star symbol denotes the target manipulation position ($\mathbf{x}^*(\mathbf{p}_r)$). (e) Curves of joint velocity commands. (f) Curves of computational joint velocity measurements.

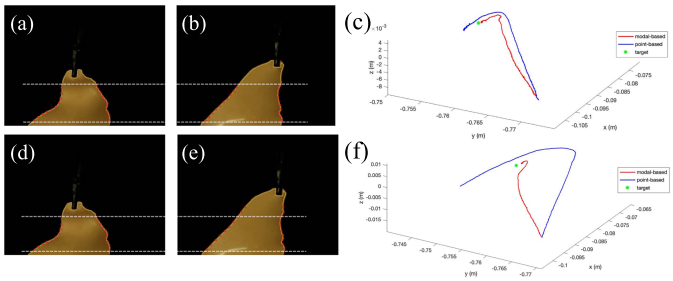


Fig. 21. Sampling illustrations and the manipulation trajectories of the comparative study: the first/second column shows the left camera views of the different sampling methods at the rest/desired configurations; the third column shows manipulation trajectories ($\mathbf{x}(\mathbf{p}_r, t)$), where the green star symbol denotes the target manipulation position ($\mathbf{x}^*(\mathbf{p}_r)$). (a)–(c) for the "good" sampling case. (d)–(f) for the "bad" sampling case.

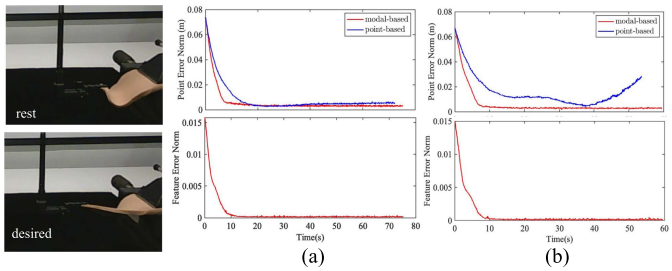


Fig. 22. Setup and error curves of the comparative study: the first column shows the third camera views of the rest and desired configurations; the second/third column shows the curves of the point error norm ($\|\mathbf{x}(\mathbf{p}_s, t) - \mathbf{x}^*(\mathbf{p}_s)\|$) and the feature error norm ($\|\mathbf{e}_s(t)\|$) of the "good"/"bad" sampling case.

C. Comparative Study

As our method belongs to the deformation-Jacobian-based approaches, we compared our method (the 3-D deformation controller using modal-based global deformation features and modal-based deformation Jacobian matrix) with the baseline method (the 3-D deformation controller using point-based local geometric features and point-based deformation Jacobian matrix) proposed by Navarro-Alarcon et al. [7].

We selected the same object [the short silicon block in Fig. 14(d)] and the same desired deformation from the rest configuration to the desired configuration in Fig. 22. We sampled 20 points from the object's contour tracked by color segmentation. To unify the two methods to the same setup, we set the sampled points $\mathbf{p}_s$ as the control points of [7] and formulated a point-based controller where the object's deformation is directly described by $\mathbf{x}(\mathbf{p}_s, t)$. According to the baseline method, we considered the following point-based deformation Jacobian matrix between a small position change of the sampled points $\mathbf{p}_s$ and a small position change of the manipulation point $\mathbf{p}_r$:

$$\delta\mathbf{x}(\mathbf{p}_s) = \mathbf{J}_p \delta\mathbf{x}(\mathbf{p}_r).$$

Based on the online estimation of the (whole) deformation Jacobian matrix $\hat{\mathbf{J}}_p(t)$, we computed the manipulation commands of the point-based controller with

$$\mathbf{v}(\mathbf{p}_r, t) = -\mathbf{K}_p \hat{\mathbf{J}}_p^+(t)(\mathbf{x}(\mathbf{p}_s, t) - \mathbf{x}^*(\mathbf{p}_s))$$

where $\hat{\mathbf{J}}_p^+(t)$ is the pseudo-inverse matrix of $\hat{\mathbf{J}}_p(t)$. We initialized $\hat{\mathbf{J}}_p$ by conducting test motions around the starting point. As to our modal-based controller, we selected the black mesh in Fig. 16 as the base mesh, and used $\mathbf{x}(\mathbf{p}_s, t)$ to compute the modal-based deformation features $\mathbf{s}(t)$. The dimension of our modal-based deformation features was set to be: $m = 20$. During the robot manipulation, we recorded the manipulation trajectories of $\mathbf{x}(\mathbf{p}_r, t)$ and the point error norm [(i.e., $\|\mathbf{x}(\mathbf{p}_s, t) - \mathbf{x}^*(\mathbf{p}_s)\|$] for both controllers. We also recorded $\|\mathbf{e}_s(t)\|$ for our modal-based controller. We adjusted $\mathbf{K}_p$ and $\mathbf{K}_s$ such that the initial velocity amplitudes of the two controllers were similar [which can be seen from the similar initial slopes of the point error norm curves in Fig. 22(a) and (b)].

We set two cases with different point sampling methods: the "good" sampling case [Fig. 21(a) and (b)], where we sampled points by dividing the contour's arc length equally during the whole manipulation process; the "bad" sampling case [Fig. 21(d) and (e)], where we only sampled points by dividing the arc length equally at t_0, and then sampled points with fixed 2-D intervals (of the y-coordinate in the image plane of the left camera). In this way, the "good" sampling case adjusted the point sampling along with the object deformation, and thus was able to sample the 3-D object points from a small neighborhood of the initial sampled points. On the other hand, the "bad" sampling case sampled the 3-D object points that kept changing and getting farther away from the manipulation point.

Figs. 21(c) and 22(a) show the results of the "good" sampling case. We can conclude that compared to the point-based controller, our modal-based controller achieves: 1) smaller steady-state errors of both $\mathbf{e}_d(t)$ and $\|\mathbf{x}(\mathbf{p}_s, t) - \mathbf{x}^*(\mathbf{p}_s)\|$; 2) faster minimization of the point errors; 3) better manipulation trajectories. Figs. 21(f) and 22(b) show the results of the "bad" sampling case. Our modal-based controller successfully reaches the target manipulation position (with only a small steady-state error) while the point-based controller fails. The reasons are: our modal-based features describe the global object deformation rather than the local 3-D positions of the sampled points; our modal-based features enable us to formulate the deformation Jacobian matrix with physically based reference directions.

D. Cases With Unsatisfactory Situations

1) Cases With Unreachable Desired Deformation: To validate our method with unreachable desired deformation, we selected one manipulation point to generate the desired deformation [see Fig. 23(a)] while using a different manipulation point for control [see Fig. 23(b)]. We set the object to be the short silicon block [see Fig. 14(d)] and the base mesh to be the black mesh in Fig. 16. The deformation feature dimension was set to be: $m = 15$. During the control phase, we recorded the point error norms [(i.e., $\|\mathbf{x}(\mathbf{p}_s, t) - \mathbf{x}^*(\mathbf{p}_s)\|$] and $\|\mathbf{e}_s(t)\|$. The results in Fig. 23(c) show that the controller reaches a local minimum with a small steady-state error which means that the resulting deformation is close to the target deformation.

2) Cases With Occlusions: To validate our method with occlusions, we set two kinds of cases: the occlusion case [see Fig. 24(a)], where we manually placed black occlusion above the

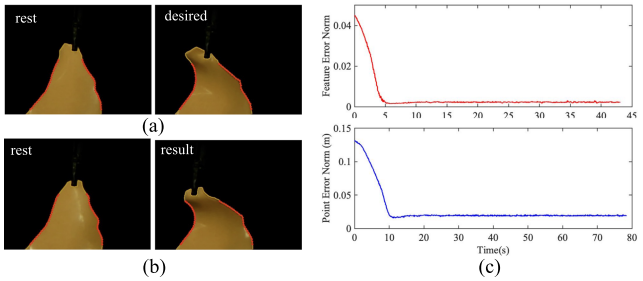


Fig. 23. Results of the case with unreachable desired deformation: (a) the left camera views of the rest/desired configuration when generating the desired deformation; (b) the left camera views of the rest/result configuration for the control task; (c) the result curves of the feature error norm ($\|\mathbf{e}_s(t)\|$) and the point error norm ($\|\mathbf{x}(\mathbf{p}_s,t)-\mathbf{x}^*(\mathbf{p}_s)\|$).

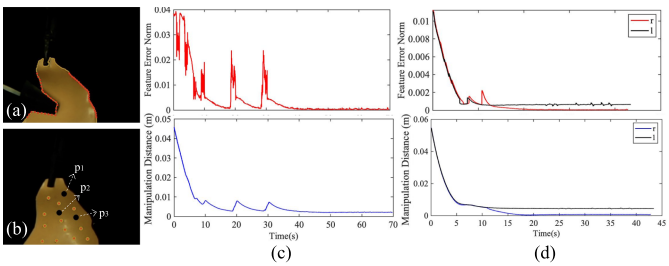


Fig. 24. Cases with occlusions and sampling loss: (a/b) snapshots of the occlusion/sampling loss case; (c/d) the feature error norm curve and the manipulation distance (i.e., $\|\mathbf{e}_d(t)\|$) curve of the occlusion/sampling loss case, where "l" is for the permanent loss case and "r" is for the temporary loss case. Bumps in the feature error curves are caused by surface sampling changes. However, the corresponding bumps in the manipulation distance curves are much smaller since the modal truncation discards the high-frequency deformation modes.

object during deformation and after convergence; the sampling loss cases ([see Fig. 24(b)], where we simulated the situations of temporary and permanent loss of some sampled points. We set the same desired deformation from the rest configuration in Fig. 16(c) to the desired configuration in Fig. 16(d). We set the object to be the short silicon block [see Fig. 14(d)] and the base mesh to be the black mesh in Fig. 16. The deformation feature dimension was set to be: $m = 15$.

For the occlusion case, we obtained surface samplings from the object's contour tracked by color segmentation. When occlusions occurred (Fig. 24(a)), we sampled points on the (partially) obscured contour, which changed the values of our deformation features (causing the large bumps of the feature error curves in Fig. 24(c)). Results in Fig. 24(c) show the minimization of both $\|\mathbf{e}_s(t)\|$ and $\|\mathbf{e}_d(t)\|$ even with dynamic occlusions. For the sampling loss cases, we obtained surface samplings by tracking points on the object. We set two sub-cases: the temporary loss case, where we simulated the process of losing point $\mathbf{p}_1 \to \mathbf{p}_2 \to \mathbf{p}_3$, and then recovering point $\mathbf{p}_3 \to \mathbf{p}_2 \to \mathbf{p}_1$; the permanent loss case, where we simulated the process of losing point $\mathbf{p}_1 \to \mathbf{p}_2 \to \mathbf{p}_3$, and then only recovering point $\mathbf{p}_3 \to \mathbf{p}_2$. As point loss/recovery caused changes of the samplings' one-to-one correspondences, we reformulated the deformation feature computation matrices in (17) by re-assembling the matrices

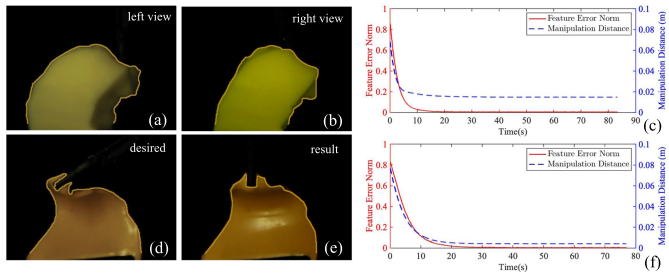


Fig. 25. Example situations where our controller cannot deform the object to the desired shape. The case with inaccurate reconstruction: (a), (b) are the camera views during manipulation, and (c) shows the result curves of $\|\mathbf{e}_s(t)\|$ and $\|\mathbf{e}_d(t)\|$. The case with torsional deformation: (d), (e) are the left camera views of the desired and resulting shapes, and (f) shows the result curves of $\|\mathbf{e}_s(t)\|$ and $\|\mathbf{e}_d(t)\|$.

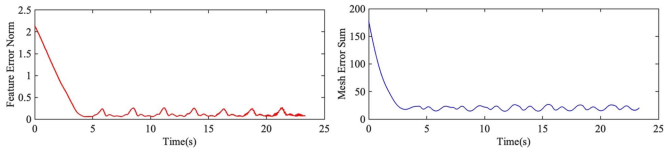
$\mathbf{N}_s(\boldsymbol{\eta}(\mathbf{p}_s),\mathbf{n}_s)$ and $[\boldsymbol{\Phi}_n]_s$ according to the changed configurations of $\mathbf{p}_s$ and $\mathbf{n}_s$. Results in Fig. 24(d) show the minimization of both $\|\mathbf{e}_s(t)\|$ and $\|\mathbf{e}_d(t)\|$. Permanent loss of the sampled point only causes a small steady-state error which means that the object was manipulated to a position close to the target position.

IX. Discussion and Conclusion

A. Limitations

There are some situations where the proposed controller cannot deform the object to (or close to) the desired shape. To start with, our control performance is limited by the accuracy of object perception and reconstruction, which is challenging for texture-less volumetric objects with complex shapes. For example, as shown in Fig. 25(a) and (b), if the cross-sectional contours detected by the two cameras correspond to different parts of the object (where the inherent self-occlusion problem in stereo matching leads to inaccurate reconstruction), the object cannot be deformed to the desired shape [implied by the steady-state errors of the manipulation distance in Fig. 25(c)]. In addition, we cannot tackle the desired shape with torsional deformation. This is because, at this stage, we design the deformation basis using linear modal analysis. Due to its linear strain assumption, we can only control the linear velocity of manipulation points. For example, given the desired shape [see Fig. 25(d)] whose generation includes twisting the object, our controller can minimize the errors of our deformation features [see Fig. 25(f)] but the resulting shape is quite different from the desired one [see Fig. 25(e)]. To solve this kind of problem, further efforts should be made to investigate rotational deformation under the framework of modal analysis.

We also discuss some methodological limitations of the proposed controller. First, our method is developed under the quasi-static assumption and we conducted validation cases with slow manipulation. Object dynamics are not considered in our controller design. To show the influence of the object dynamics on the closed-loop behavior, we simulated a case with accelerated manipulation and lowered Young's modulus of the object,



(a) (b)

Fig. 26. Simulation analysis of the vibrating object. (a) Result curve of $\|\mathbf{e}_s(t)\|$. (b) Result curve of $\mathbf{e}_x(t)$.

causing the dynamic deformation to be non-negligible. The results in Fig. 26 show that the deformation feature and shape errors only converge to some bounded areas and oscillate with the object's vibrations. To minimize the errors and compress the vibrations, we need to further study object dynamics in our controller design, which will be a future direction of our research. Second, we need to track image features on the object (such as points, curves, and contours) to obtain surface samplings. Point detection or object segmentation is required. We also need to find a start point of the segmented contour or curve to track the spatial order of the sampled points. In the experiments, we selected the lowest left corner of the object's fixed end as the start point. Another limitation is the local nature of the deformation Jacobian matrix and the linear modal analysis. Further efforts can be made to improve the method with global control or planning strategies.

X. Conclusion

This article proposed an adaptive 3-D shape deformation controller using novel global deformation features based on modal analysis. Instead of performing modal analysis on the object, we proposed a model-free framework using truncated modes of the base mesh to span a low-dimensional deformation feature space with unique object representations. All the modeling uncertainties introduced by the unknown geometry and physical properties of the object were treated as low-dimensional modal parameters in the deformation Jacobian matrix. Based on the Jacobian matrix, we formulated an adaptive controller where the modal parameters can be linearized from the closed-loop error dynamics. In this way, we can design the online parameter estimation laws with guaranteed control stability proved using the Lyapunov-based method. We conducted simulations to validate our controller with different modeling, sampling, and manipulating conditions. Extensive experiments were conducted with different linear, planar, and volumetric objects under different stereo measurements of points, curves, and contours. The comparative study with the baseline method showed the advantages of our controller. We also validated the effectiveness of our controller with large pose and size errors of the base mesh, with large deformation, with unreachable desired deformation, and with occlusions.

To further generalize the modal-based deformation control approach, our future works will develop methods without requiring tracking image features on the object. Global and dynamic deformation control will also be investigated.

References

[1] C. Shin, P. W. Ferguson, S. A. Pedram, J. Ma, E. P. Dutson, and J. Rosen, "Autonomous tissue manipulation via surgical robot using learning based model predictive control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 3875–3881.

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

[3] X. Li, X. Su, and Y.-H. Liu, "Vision-based robotic manipulation of flexible PCBs," *IEEE/ASME Trans. Mechatron.*, vol. 23, no. 6, pp. 2739–2749, Dec. 2018.

[4] T. Wada, S. Hirai, S. Kawamura, and N. Kamiji, "Robust manipulation of deformable objects by a simple pid feedback," in *Proc. ICRA. IEEE Int. Conf. Robot. Autom.*, 2001, vol. 1, pp. 85–90.

[5] J. Das and N. Sarkar, "Autonomous shape control of a deformable object by multiple manipulators," *J. Intell. Robot. Syst.*, vol. 62, no. 1, pp. 3–27, 2011.

[6] S. Tokumoto and S. Hirai, "Deformation control of rheological food dough using a forming process model," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, vol. 2, pp. 1457–1464.

[7] D. Navarro-Alarcon et al., "Automatic 3-D manipulation of soft objects by robotic arms with an adaptive deformation model," *IEEE Trans. Robot.*, vol. 32, no. 2, pp. 429–441, Apr. 2016.

[8] 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.

[9] 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 Trans. Robot.*, vol. 34, no. 1, pp. 272–279, Feb. 2018.

[10] J. Qi et al., "Contour moments 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.

[11] 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.

[12] M. Abayazid, P. Moreira, N. Shahriari, S. Patil, R. Alterovitz, and S. Misra, "Ultrasound-guided three-dimensional needle steering in biological tissue with curved surfaces," *Med. Eng. Phys.*, vol. 37, no. 1, pp. 145–150, 2015.

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

[14] V. E. Arriola-Rios, P. Guler, F. Ficuciello, D. Kragic, B. Siciliano, and J. L. Wyatt, "Modeling of deformable objects for robotic manipulation: A tutorial and review," *Front. Robot. AI*, vol. 7, 2020, Art. no. 82.

[15] F. Nadon, A. J. Valencia, and P. Payeur, "Multi-modal sensing and robotic manipulation of non-rigid objects: A survey," *Robotics*, vol. 7, no. 4, 2018, Art. no. 74.

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

[17] D. Navarro-Alarcon, Y.-H. Liu, J. G. Romero, and P. Li, "Model-free visually servoed deformation control of elastic objects by robot manipulators," *IEEE Trans. Robot.*, vol. 29, no. 6, pp. 1457–1468, Dec. 2013.

[18] 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.

[19] J. Qi, W. Ma, D. Navarro-Alarcon, H. Gao, and G. Ma, "Adaptive shape servoing of elastic rods using parameterized regression features and auto-tuning motion controls," 2020, *arXiv:2008.06896*.

[20] R. Lagneau, A. Krupa, and M. Marchal, "Active deformation through visual servoing of soft objects," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 8978–8984.

[21] A. Cherubini, V. Ortenzi, A. Cosgun, R. Lee, and P. Corke, "Model-free vision-based shaping of deformable plastic materials," *Int. J. Robot. Res.*, vol. 39, no. 14, pp. 1739–1759, 2020.

[22] 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.

[23] A.-M. Cretu, P. Payeur, and E. M. Petriu, "Soft object deformation monitoring and learning for model-based robotic hand manipulation," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 42, no. 3, pp. 740–753, Jun. 2012.

[24] M. Yu, K. Lv, H. Zhong, S. Song, and X. Li, "Global model learning for large deformation control of elastic deformable linear objects: An efficient and adaptive approach," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 417–436, Feb. 2023.
[25] J. Montagnat, H. Delingette, and N. Ayache, "A review of deformable surfaces: Topology, geometry and deformation," *Image Vis. Comput.*, vol. 19, no. 14, pp. 1023–1040, 2001.
[26] K. Madi, E. Paquet, and H. Kheddouci, "New graph distance for deformable 3D objects recognition based on triangle-stars decomposition," *Pattern Recognit.*, vol. 90, pp. 297–307, 2019.
[27] J. Fayad, L. Agapito, and A. Del Bue, "Piecewise quadratic reconstruction of non-rigid surfaces from monocular sequences," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 297–310.
[28] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1498–1505.
[29] M.-P. Gascuel, "An implicit formulation for precise contact modeling between flexible solids," in *Proc. 20th Annu. Conf. Comput. Graph. Interactive Techn.*, 1993, pp. 313–320.
[30] M. Prasad, A. Fitzgibbon, A. Zisserman, and L. Van Gool, "Finding NEMO: Deformable object class modelling using curve matching," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1720–1727.
[31] C. B. Kelemen and G. Gerig, "Segmentation of 2-D and 3-D objects from MRI volume data using constrained elastic deformations of flexible Fourier contour and surface models," *Med. Image Anal.*, vol. 1, no. 1, pp. 19–34, 1996.
[32] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *Proc. 13th Annu. Conf. Comput. Graph. Interactive Techn.*, 1986, pp. 151–160.
[33] A. R. Fugl, A. Jordt, H. G. Petersen, M. Willatzen, and R. Koch, "Simultaneous estimation of material properties and pose for deformable objects from depth and color images," in *Proc. Joint DAGM (German Assoc. Pattern Recognit.) OAGM Symp.*, 2012, pp. 165–174.
[34] L. Zaidi, J. A. Corrales, B. C. Bouzgarrou, Y. Mezouar, and L. Sabourin, "Model-based strategy for grasping 3D deformable objects using a multi-fingered robotic hand," *Robot. Auton. Syst.*, vol. 95, pp. 196–206, 2017.
[35] A. H. Barr, "Global and local deformations of solid primitives," in *Readings in Computer Vision*. Amsterdam, The Netherlands:Elsevier, 1987, pp. 661–670.
[36] A. P. Pentland, "Perceptual organization and the representation of natural form," in *Readings in Computer Vision*. Amsterdam, The Netherlands:Elsevier, 1987, pp. 680–699.
[37] J. Barbič and D. L. James, "Real-time subspace integration for st. Venant-Kirchhoff deformable models," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 982–990, 2005.
[38] S. S. An, T. Kim, and D. L. James, "Optimizing cubature for efficient integration of subspace deformations," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–10, 2008.
[39] D. Metaxas and D. Terzopoulos, "Dynamic deformation of solid primitives with constraints," in *Proc. 19th Annu. Conf. Comput. Graph. Interactive Techn.*, 1992, pp. 309–312.
[40] P. Krysl, S. Lall, and J. E. Marsden, "Dimensional model reduction in non-linear finite element dynamics of solids and structures," *Int. J. Numer. Methods Eng.*, vol. 51, no. 4, pp. 479–504, 2001.
[41] A. Pentland and J. Williams, "Good vibrations: Modal dynamics for graphics and animation," in *Proc. 16th Annu. Conf. Comput. Graph. Interactive Techn.*, 1989, pp. 215–222.
[42] D. Baraff and A. Witkin, "Dynamic simulation of non-penetrating flexible bodies," *ACM SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 303–308, 1992.
[43] M. G. Choi and H.-S. Ko, "Modal warping: Real-time simulation of large rotational deformation and manipulation," *IEEE Trans. Vis. Comput. Graph.*, vol. 11, no. 1, pp. 91–101, Jan./Feb. 2005.
[44] A. Pentland and S. Sclaroff, "Closed-form solutions for physically based shape modeling and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 7, pp. 715–729, Jul. 1991.
[45] A. Agudo, L. Agapito, B. Calvo, and J. M. Montiel, "Good vibrations: A modal analysis approach for sequential non-rigid structure from motion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1558–1565.
[46] L. Ljung, "Perspectives on system identification," *Annu. Rev. Control*, vol. 34, no. 1, pp. 1–12, 2010.
[47] A. Pentland and B. Horowitz, "Recovery of nonrigid motion and structure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 7, pp. 730–742, Jul. 1991.
[48] K.-J. Bathe, *Finite Element Procedures*. Berlin, Germany:Klaus-Jurgen Bathe, 2006.
[49] A. Witkin and W. Welch, "Fast animation and control of nonrigid structures," in *Proc. 17th Annu. Conf. Comput. Graph. Interactive Techn.*, 1990, pp. 243–252.
[50] A. Pentland and J. Williams, "Perception of non-rigid motion: Inference of shape, material and force," in *Proc. 11th Int. Joint Conf. Artif. Intell.*, 1989, vol. 2, pp. 1565–1570.
[51] F. Solina and R. Bajcsy, "Recovery of parametric models from range images: The case for superquadrics with global deformations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 2, pp. 131–147, Feb. 1990.
[52] D. Terzopoulos, A. Witkin, and M. Kass, "Symmetry-seeking models and 3D object reconstruction," *Int. J. Comput. Vis.*, vol. 1, no. 3, pp. 211–221, 1988.
[53] P. I. Kattan, *MATLAB Guide to Finite Elements: An Interactive Approach*. Berlin, Germany:Springer, 2010.
[54] J. L. Junkins, *Introduction to Dynamics and Control of Flexible Structures*. Washington, DC, USA: Amer. Inst. Aeronaut. Astronaut., 1993.
[55] J. Barbič, "FEM simulation of 3D deformable solids: A practitioner's guide to theory, discretization and model reduction. Part 2: Model reduction," in *Proc. ACM SIGGRAPH Course Notes*, 2012, Art. no. 20.
[56] C. Von Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt, "An efficient construction of reduced deformable objects," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–10, 2013.
[57] S. Hirai and T. Wada, "Indirect simultaneous positioning of deformable objects with multi-pinching fingers based on an uncertain model," *Robotica*, vol. 18, no. 1, pp. 3–11, 2000.
[58] F. Zhong, Y. Wang, Z. Wang, and Y.-H. Liu, "Dual-arm robotic needle insertion with active tissue deformation for autonomous suturing," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2669–2676, Jul. 2019.
[59] D. Navarro-Alarcon, Y.-h. Liu, J. G. Romero, and P. Li, "On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments," *Int. J. Robot. Res.*, vol. 33, no. 11, pp. 1462–1480, 2014.
[60] R. W. Ogden, *Non-Linear Elastic Deformations*. Chelmsford, MA, USA:Courier Corporation, 1997.
[61] 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. Autom.*, 2022, pp. 1337–1343.
[62] J.-J. E. Slotine and W. Li, "On the adaptive control of robot manipulators," *Int. J. Robot. Res.*, vol. 6, no. 3, pp. 49–59, 1987.
[63] J.-J. E. Slotine et al., *Applied Nonlinear Control*, vol. 199, no. 1. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.
[64] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control*. Berlin, Germany:Springer, 1998, pp. 66–78.
[65] J. Allard et al., "SOFA-An open source framework for medical simulation," in *Proc. Conf. Med. Meets Virtual Reality*, 2007, vol. 125, pp. 13–18.
[66] 2023. [Online]. Available: https://github.com/sofa-framework/sofa/commits/master/examples/Tutorials/ForceFields/TutorialForceFieldLiverFEM.scn
[67] 2018. [Online]. Available: https://s3-eu-west-1.amazonaws.com/ur-support-site/32554/scriptManual-3.5.4.pdf

Bohan Yang received the B.Eng. degree in automation from Hunan University, Changsha, China, in 2014, the M.S. degree in control science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 2017, and the Ph.D. degree in mechanical and automation engineering from the Chinese University of Hong Kong, HKSAR, China, in 2023.

She is currently a Postdoctoral Fellow with the T Stone Robotics Institute, The Chinese University of Hong Kong. Her research interests include visual serving and medical robotics.

Bo Lu (Member, IEEE) received the B.Eng. degree in ship and offshore engineering from the Department of Ship and Offshore Engineering, Dalian University of Technology, Liaoning Province, China, in 2013. He received the M.S. degree (first Hons.) and the Ph.D. degree in mechanical engineering from the Department of Mechanical Engineering, The Hong Kong Polytechnic University, HKSAR, China, in 2015 and 2019, respectively.

Afterward, he worked as a Postdoctoral Research Fellow at the T-stone Robotics Institute, The Chinese University of Hong Kong, HKSAR, China. He is currently an Associate Professor with the Robotics and Microsystems Center, School of Mechanical and Electric Engineering, Soochow University. His current research interests include medical robotics, computer vision, and vision-based and learning-driven automation and intervention.

Wei Chen received the B.E. degree in computer science and technology from Zhengzhou University, Zhengzhou, China, in 2012, and the M.S. degree in mechanical and automation engineering from The Chinese University of Hong Kong, Hong Kong in 2021. He is currently working toward the Ph.D. degree in mechanical and automation engineering with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, HKSAR, China.

Fangxun Zhong received the B.Eng. degree in automation from Beijing Institute of Technology, Beijing, China, in 2014 and the Ph.D. degree in mechanical and automation engineering from The Chinese University of Hong Kong, China, in 2021.

He is currently a Postdoctoral Fellow with the T Stone Robotics Institute, The Chinese University of Hong Kong. His research interests include surgery autonomy, medical robotics, dexterous robot planning and control.

Yun-Hui Liu (Fellow, IEEE) received the Ph.D. degree in applied mathematics and information physics from the University of Tokyo, Tokyo, Japan, in 1992.

After working at the Electrotechnical Laboratory of Japan as a Research Scientist, he joined The Chinese University of Hong Kong (CUHK) in 1995 and is currently Choh-Ming Li Professor of Mechanical and Automation Engineering and the Director of the T Stone Robotics Institute. He also serves as the Director/CEO of Hong Kong Centre for Logistics Robotics sponsored by the InnoHK programme of the HKSAR Government. He is an Adjunct Professor with the State Key Lab of Robotics Technology and System, Harbin Institute of Technology, China. He has published more than 500 papers in refereed journals and refereed conference proceedings and was listed in the Highly Cited Authors (Engineering) by Thomson Reuters in 2013. His research interests include visual servoing, logistics robotics, medical robotics, multifingered grasping, mobile robots, and machine intelligence.

Dr. Liu has received numerous research awards from international journals and international conferences in robotics and automation and government agencies. He was the Editor-in-Chief of Robotics and Biomimetics and served as an Associate Editor of the IEEETRANSACTION ON ROBOTICS AND AUTOMATION and General Chair of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems.