Adaptive Shape Servoing of Elastic Rods using Parameterized Regression Features and Auto-Tuning Motion Controls

Jiaming Qi*, Wanyu Ma*, David Navarro-Alarcon, *Senior Member, IEEE*, Han Gao and Guangfu Ma

Abstract—In this paper, we present a new vision-based method to control the shape of elastic rods with robot manipulators. Our new method computes parameterized regression features from online sensor measurements that enable to automatically quantify the object's configuration and establish an explicit shape servo-loop. To automatically deform the rod into a desired shape, our adaptive controller iteratively estimates the differential transformation between the robot's motion and the relative shape changes; This valuable capability allows to effectively manipulate objects with unknown mechanical models. An auto-tuning algorithm is introduced to adjust the robot's shaping motion in real-time based on optimal performance criteria. To validate the proposed theory, we present a detailed numerical and experimental study with vision-guided robotic manipulators.

Index Terms—Robotics, visual servoing, deformable objects, sensor-based control, model estimation.

I. INTRODUCTION

THE manipulation of deformable objects is currently an open (and hot) research problem in robotics [1] that has attracted many researchers due to its great applicability in many areas, e.g. manipulating fabrics [2], shaping of food materials [3], assembling soft components [4], manipulating cables [5], interacting with tissues [6], etc. Note that physical interactions between a robot and a deformable object will inevitably alter the object's shape. The feedback control of these additional object degrees-of-freedom (DOF) is referred to in the literature as shape servoing [7], a frontier problem that presents three main challenges: (i) The efficient feedback characterization of the object's shape (which has infinite number of DOF to be controlled by a robot with limited manipulation directions); (ii) The computation of a motion-deformation model for control (which depends on the—typically unknown—mechanical properties of the object); (iii) The online adjustment of the robot's motion during the soft object manipulation task (note that these objects might be delicate and easy to damage).

J. Qi and G. Ma are with the Harbin Institute of Technology, China.

W. Ma and D. Navarro-Alarcon are with The Hong Kong Polytechnic University, KLN, Hong Kong. Corresponding author: dna@ieee.org

H. Gao is with the Beijing Institute of Technology, China.

For shape servoing tasks, an excellent feature extraction algorithm can describe the soft object to the greatest extent with the least number of feature coordinates. The most widely used image features are feature points, centroid points, distance/angle/curvature features, and other artificially marked points [8]. However, as the above are local features, and hard to describe the overall geometric information of soft objects. Thus, the development of global features present an advantage in this problem. In [9], image moments were used to characterize the object's contour, but its real-time performance was limited due to a large amount of calculations. In [10], Principal Component Analysis was used to project raw Fast Point Feature Histograms into a new space with higher variance and a lower dimension. A catenary-based feature descriptor was developed for tethered wheeled robots and underwater vehicles in [11], [12]; However, this approach is only applicable to very specific shapes, hence, cannot be used for complex contours. Recently, a method based on Fourier coefficients was developed in [13], [14]; This approach can effectively compute a low-dimension feature vector to represent the shape of complex objects. Bézier curves and Non-Uniform Rational Basis Splines (NURBS) are an interesting option to describe complex contours, however, they have not been thoroughly exploited in the literature to establish an explicit shape servo-loop. Other types of shape feature representations rely on machine learning, e.g. Nair combined learning and visual feedback to manipulate ropes in [15]. Although learning-based approaches have good adaptability, they require large and "rich enough" data sets to generalize to diffvnerent situations (which is difficult to guarantee in many applications). Designing a computationally efficient feature extraction algorithm that provides a reliable representation for controller design is an important problem in soft object manipulation.

To execute shape servoing tasks, a controller requires a model (e.g. a Jacobian matrix) that describes the relationship between robot motions and feedback deformations. In [16], the authors used parameter linearization and least-squares methods to compute the deformation Jacobian matrix. Then, to improve real-time performance, an online estimation technique and its convergence proof were developed in [17]. However, both methods depend on the selection of the regression matrix and need a prior-known structure. The Broyden method was used to online estimate the Jacobian matrix in [18] without a known structure. In [19], the authors combined dynamic recursive least square with online estimation; Although these

methods have a small amount of calculation and are easy to implement, they are prone to enter local minima. Kalman Filter (KF) has an excellent performance in estimating unknown variables using a series of measurements observed over time in the presence of noise and uncertainty. Thus, KF is a good option for online estimating the sensorimotor relations in our soft object manipulation problem. For example, in [20] a state-space model was established using the Jacobian matrix elements as the system state and used KF to observe its unknown values. With standard Kalman filters, the noise covariance matrix is often a constant matrix, for solving this limitation, the authors in [21] proposed a fuzzy adaptive KF to deal with a time-varying covariance matrix. In our soft object manipulation problem, a good Jacobian matrix estimation algorithm is necessary to properly "steer" the object towards a desired target shape.

Although model-free shape control (as formulated in [16]) is known for its simple structure and robustness to uncertainties, these controllers generally use a constant feedback gain (which limits the types of dynamic responses they can achieve). When the performance requirements change (e.g. deforming materials with different stiffness), a fixed gain may produce oscillations, cause the system to lose stability or even damage the manipulated object. Therefore, it is important to design a dynamic parameter tuning algorithm for shape servoing tasks, such that it can adapt to various performance requirements.

Based on the limitations of the above mentioned works, in this paper, we propose a new solution with the following original contributions:

1) We present new feedback feature vectors (constructed with the coefficients of Bézier and NURBS) to efficiently represent elastic rods.
2) We propose a new adaptive deformation controller with KF-based estimators and online parameter optimization.
3) We report detailed simulations and experimental results to validate the proposed method.

To the best of authors knowledge, this is the first time that a shape servo-controller uses Bézier/NURBS to establish an explicit shape servo-loop. These features were used in [22] but only to project the object's contour into the image plane, which largely differs from our feedback representation approach.

The rest of this paper is organized as follows: Section II presents the preliminaries; Section III describes the methods; Section IV and Section V presents the results; Section VI gives final conclusions.

II. PRELIMINARIES

Notation. Throughout this paper we use very standard notation. Column vectors are denoted with bold small letters $\mathbf{v}$ and matrices with bold capital letters $\mathbf{M}$. Time evolving variables are represented as $\mathbf{m}_k$, where the subscript $*_k$ denotes the discrete time instant. $\mathbf{E}_n$ is an $n \times n$ identity matrix.

Our aim in this work is to solve the automatic shape control of elastic rods with visual servoing. We denote the vector of robot motion by $\mathbf{r}_k \in \mathbb{R}^q$. To derive our new method, let us first consider the following conditions:
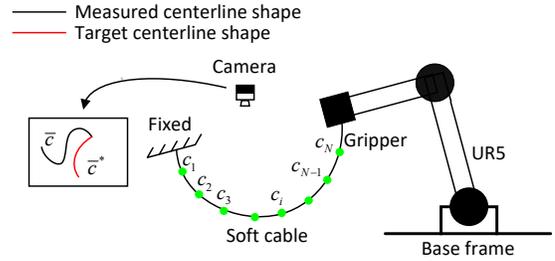

Fig. 1. Representation of the cable manipulation case of study, where a vision sensor continuously measures the feature vector $\mathbf{s}$ of the cable, which should be accurately deformed into target feature $\mathbf{s}^*$ within the controller.

$\bar{\mathbf{c}}$ centerline data $\hat{\mathbf{J}}$ Jacobian estimation
λ control coefficient $1/z$ first-order delay
$\mathbf{s}$ measured feature vector $\mathbf{s}^*$ target feature vector
$\Delta\mathbf{s}$ differentiation of $\mathbf{s}$ $\Delta\mathbf{r}$ velocity command signal
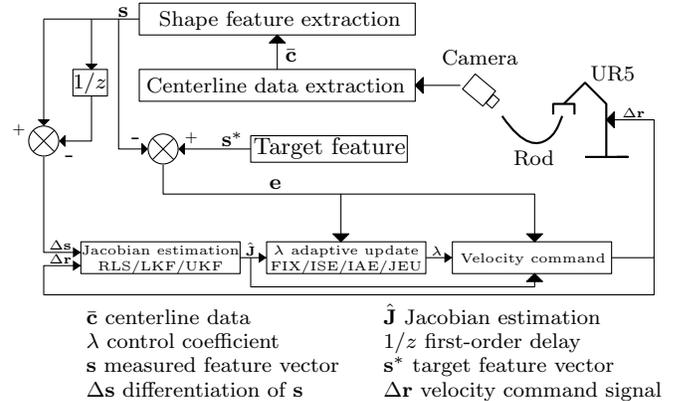
Fig. 2. The block diagram showing the overall workflow of the system.

- The shape of the rod is measured with a fixed camera in an eye-to-hand configuration (depicted in Fig. 1). We denote the 2D image contour as:

$$\bar{\mathbf{c}} = [\mathbf{c}_1^\mathsf{T}, \ldots, \mathbf{c}_N^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{2N} \quad \mathbf{c}_i = [u_i, v_i]^\mathsf{T} \in \mathbb{R}^2 \quad (1)$$

where, N is the number of the contour points, $\mathbf{c}_i$ denotes the ith $(i = 1, \cdots, N)$ point of the contour in the image, u_i and v_i are the coordinates under the image frame.
- During the manipulation task, the rod is rigidly grasped by the robot and remains all the time within the observable range of the camera, and no occlusion occurs.
- The robot's motion is commanded with classical kinematic controls $\Delta\mathbf{r}_k$ [23] that render stiff behaviours and satisfy the incremental position motions $\mathbf{r}_k = \mathbf{r}_{k-1} + \Delta\mathbf{r}_k$.
- The rod is manipulated at low speed such that its shape is determined by the equilibrium of its potential/elastic energy terms only.

Problem Statement. Design a vision-based adaptive control scheme to automatically deform an elastic rod into a desired 2D image shape, without requiring any knowledge of the object's mechanical model.

III. METHODS

Fig. 2 shows the block diagram of the overall manipulation task. The centerline data extraction represents the image processing pipeline, which will be given in Section V-A.
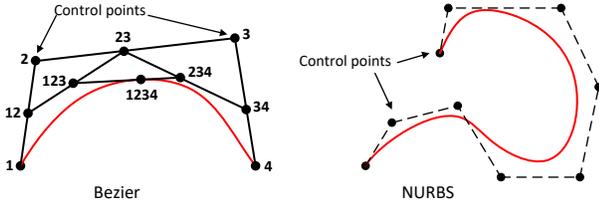
Fig. 3. The schematic diagrams of the Bézier and NURBS curve.

A. Feedback Shape Parameters

The *naïve approach* to shape servoing is to synthesise a regulator that attempts to drive the full coordinates of $\bar{\mathbf{c}}$ into a desired 2D contour. The main problem with this approach is that the raw vector $\bar{\mathbf{c}}$ is not an efficient signal for real-time control as its dimension very large. Therefore, it is necessary to design an algorithm that computes a reduced dimension feature vector, which can be used for feedback control. In this paper, we fit the feedback signal to a continuous curve $\boldsymbol{f}(\rho)$ that describes the 2D contour, where ρ is a variable constructed with sensor data which represents the normalized arc-length $(0 \leq \rho \leq 1)$. Then, $\mathbf{c}_i = \boldsymbol{f}(\rho_i)$, where ρ_i is the arc-length between the start point $\mathbf{c}_1$ and the point $\mathbf{c}_i$ (see Fig. 1), and $\rho_1 = 0, \rho_N = 1$.

The idea behind the proposed method proposed is to compute the coefficients of a linearly parametrized regression model of the 2D contour, and use it as a quasi-measurement of the object's shape. Such model has the general form:

$$\bar{\mathbf{c}} = \bar{\mathbf{G}}(\rho) \cdot \mathbf{s} \tag{2}$$

where $\bar{\mathbf{G}}$ is a regression-like matrix with a known structure, and $\mathbf{s}$ is a vector of *unknown* parameters (which in our formulation represents the shape feature vector). We illustrate this principle with four representative examples: polynomial, Bézier, NURBS, and Fourier. The schematic diagrams of the Bézier and NURBS curves are shown in Fig. 3.

- *Polynomial Parameterization.* Given a variable parameter ρ, the general formula is to approximate a curve is:

$$\boldsymbol{f}(\rho) = \sum_{j=0}^{n} \rho^j \mathbf{p}_j \tag{3}$$

where the positive number $n \in \mathbb{N}$ is the order of the polynomial, and $\mathbf{p}_j = [a_j, b_j]^\mathsf{T}$ are the shape parameters of the 2D contour. Polynomial regression can easily represent smooth regular shapes, however, if the order n is selected too large, it may produce overfitting of the curve.

- *Bézier Parameterization.* As shown in Fig.3, Bézier curve approximates the curve with a polynomial expression using control points. $n+1$ control points can determine a n-degree Bézier curve which is described as follows:

$$\boldsymbol{f}(\rho) = \sum_{j=0}^{n} B_{j,n}(\rho) \mathbf{p}_j,$$

$$B_{j,n}(\rho) = \frac{n!}{j!(n-j)!} (1-\rho)^{n-j} \rho^j, \tag{4}$$

where $\mathbf{p}_j = [a_j, b_j]^\mathsf{T}$ are control points as well as the shape parameters of the 2D contour. Bézier curve has a first-order derivability. It guarantees that the fitting will advance smoothly with the control points without fluctuations, thus, it can represent complex shapes. Yet, note that when many control points are used, it will increase the degree of Bézier curve and the computational burden.

- *NURBS Parameterization.* It has local shape description properties, thus, the number of control points is independent from the degree of curves. This model can describe complex curves more accurately and efficiently than polynomial and Bézier. The definition of a mth-degree NURBS curve [24] is:

$$\boldsymbol{f}(\rho) = \frac{\sum_{j=0}^{n} N_{j,m}(\rho) \omega_j \mathbf{p}_j}{\sum_{j=0}^{n} N_{j,m}(\rho) \omega_j} \tag{5}$$

where n is the approximation degree of the NURBS, $\mathbf{p}_j = [a_j, b_j]^\mathsf{T}$ are control points, ω_j are the weights, $N_{j,m}$ are m-degree B-spline basis functions. We set

$$R_{j,m}(\rho) = \frac{N_{j,m}(\rho) \omega_j}{\sum_{l=0}^{n} N_{l,m}(\rho) \omega_l} \tag{6}$$

to rewrite Eq. (5) in the form

$$\boldsymbol{f}(\rho) = \sum_{j=0}^{n} R_{j,m}(\rho) \mathbf{p}_j \tag{7}$$

Along this work, we set $m = n$ such that the NURBS reduces to a rational Bézier curve and $N_{j,m}(\rho)$ reduces to $B_{j,m}(\rho)$ in (4).

- *Fourier Parametrization* Besides the above three cases, Fourier series (as presented in [13]) are also a type of regression equation. A Fourier curve of degree n is described as follows:

$$\boldsymbol{f}(\rho) = \begin{bmatrix} a_0 \\ c_0 \end{bmatrix} + \sum_{j=1}^{n} \begin{bmatrix} a_j & b_j \\ c_j & d_j \end{bmatrix} \begin{bmatrix} \cos(j\rho) \\ \sin(j\rho) \end{bmatrix} \tag{8}$$

where $[a_0, c_0]^\mathsf{T}$ and $\mathbf{p}_j = [a_j, b_j, c_j, d_j]^\mathsf{T}$ are components of frequencies as well as the shape parameters of the 2D contour.

Parameterized Regression. We can rewrite (3), (4), (7) and (8) into the linear parameterization form $\boldsymbol{f} = \mathbf{G}(\rho)\mathbf{s}$, with the regression matrix $\mathbf{G}$ and feature vector $\mathbf{s}$ defined by:

- For polynomial:

$$\mathbf{F}_j = \mathrm{diag}(\rho^j, \rho^j) \in \mathbb{R}^{2 \times 2}$$
$$\mathbf{G} = [\mathbf{F}_0, \cdots, \mathbf{F}_n] \in \mathbb{R}^{2 \times 2(n+1)}$$
$$\mathbf{s} = [\mathbf{p}_0^\mathsf{T}, \cdots, \mathbf{p}_n^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{2(n+1)} \tag{9}$$

- For Bézier curve:

$$\mathbf{F}_j = \mathrm{diag}(B_{j,n}(\rho), B_{j,n}(\rho)) \in \mathbb{R}^{2 \times 2}$$
$$\mathbf{G} = [\mathbf{F}_0, \cdots, \mathbf{F}_n] \in \mathbb{R}^{2 \times 2(n+1)}$$
$$\mathbf{s} = [\mathbf{p}_0^\mathsf{T}, \cdots, \mathbf{p}_n^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{2(n+1)} \tag{10}$$

- For NURBS:

$$\mathbf{F}_j = \mathrm{diag}(R_{j,n}\left(\rho\right), R_{j,n}\left(\rho\right)) \in \mathbb{R}^{2\times 2}$$
$$\mathbf{G} = [\mathbf{F}_0, \cdots, \mathbf{F}_n] \in \mathbb{R}^{2\times 2(n+1)}$$
$$\mathbf{s} = [\mathbf{p}_0^\intercal, \cdots, \mathbf{p}_n^\intercal]^\intercal \in \mathbb{R}^{2(n+1)} \tag{11}$$

- For Fourier:

$$\mathbf{F}_j = \begin{bmatrix} \cos\left(j\rho\right) & \sin\left(j\rho\right) & 0 & 0 \\ 0 & 0 & \cos\left(j\rho\right) & \sin\left(j\rho\right) \end{bmatrix} \in \mathbb{R}^{2\times 4}$$
$$\mathbf{G} = [\mathbf{E}_2, \mathbf{F}_1, \cdots, \mathbf{F}_n] \in \mathbb{R}^{2\times(4n+2)}$$
$$\mathbf{s} = [a_0, c_0, \mathbf{p}_1^\intercal, \cdots, \mathbf{p}_n^\intercal]^\intercal \in \mathbb{R}^{(4n+2)} \tag{12}$$

By using N sample points combined with the different regression equations (9), (10), (11), and (12), we can get the same long structures presented in (2):

$$\bar{\mathbf{c}} = [\mathbf{c}(\rho_1)^\intercal, \cdots, \mathbf{c}(\rho_N)^\intercal]^\intercal$$
$$\bar{\mathbf{G}} = [\mathbf{G}(\rho_1)^\intercal, \ldots, \mathbf{G}(\rho_N)^\intercal]^\intercal$$

Thus, the feature vector $\mathbf{s}$ is computed from sensor feedback at every iteration as:

$$\mathbf{s} = \mathcal{G}\cdot\bar{\mathbf{c}} \qquad \text{for} \quad \mathcal{G} = \left(\bar{\mathbf{G}}^\intercal\bar{\mathbf{G}}\right)^{-1}\bar{\mathbf{G}}^\intercal \tag{13}$$

To invert $\bar{\mathbf{G}}^\intercal\bar{\mathbf{G}}$, a sufficient number N of data points must be used such that $2N > 2(n+1)$ (or $2N > 4(n+2)$ for the Fourier case).

Remark 1. Although this paper only gives four forms of curve parametrization, there are many other geometric expressions (e.g. B-spline and rational approximation) that can be linearly parameterized as (2).

B. Approximation of the Local Deformation Model

Since we consider regular (i.e. mechanically well-behaved) elastic objects, it is reasonable to assume that small robot motions $\Delta\mathbf{r}_k = \mathbf{r}_k - \mathbf{r}_{k-1} \in \mathbb{R}^q$ will produce small changes $\Delta\bar{\mathbf{c}}$ in the observed 2D contour. We locally model this situation (around the current operating point) with the following expression:

$$\Delta\bar{\mathbf{c}}_k = \mathbf{D}_k \cdot \Delta\mathbf{r}_k \tag{14}$$

where we introduce the matrix $\mathbf{D}_k$ to model the local deformation properties of the elastic object undergoing quasi-static manipulation by the robot. By combining (14) with (13) we obtain the motion model:

$$\Delta\mathbf{s}_k = \mathcal{G}\mathbf{D}_k \cdot \Delta\mathbf{r}_k = \mathbf{J}_k \cdot \Delta\mathbf{r}_k \tag{15}$$

where $\Delta\mathbf{s}_k = \mathbf{s}_k - \mathbf{s}_{k-1} \in \mathbb{R}^p$ denotes the features' changes, and the matrix $\mathbf{J}_k = \mathcal{G}\mathbf{D}_k \in \mathbb{R}^{p\times q}$ represents a Jacobian-like matrix that transforms robot motions into shape changes, and which cannot be analytically computed as the deformation properties of the object are *unknown*. Instead of identifying the full mechanical model, in this paper we design an algorithm that computes local approximations of $\mathbf{J}_k$ in real-time. For estimating the Jacobian-like matrix, we use two KFs, LKF and UKF, which excellent real-time performance and are robust to external disturbances [25]. Throughout this note, we assume that $\mathbf{J}_k$ is full column rank during the manipulation task (a condition that is easy to satisfy in practice as the dimension of $\mathbf{s}$ is much larger than $\mathbf{r}$). Consider the discrete form of (15)

$$\mathbf{s}_k = \mathbf{s}_{k-1} + \mathbf{J}_k \cdot \Delta\mathbf{r}_k \tag{16}$$

with state $\mathbf{x}_k = [\partial s_1/\partial\mathbf{r}, \ldots, \partial s_p/\partial\mathbf{r}]^\intercal \in \mathbb{R}^{pq}$, and $\partial s_i/\partial\mathbf{r} = [\partial s_i/\partial r_1, \ldots, \partial s_i/\partial r_q] \in \mathbb{R}^{1\times q}$ is the ith row of the Jacobian $\mathbf{J}_k$, the discrete system (16) can be transformed into the linear stochastic system with no control input:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \boldsymbol{\eta}_k$$
$$\mathbf{y}_k = \mathbf{M}_k \cdot \mathbf{x}_k + \boldsymbol{\nu}_k \tag{17}$$

The system output is defined as $\mathbf{y}_k = \Delta\mathbf{s}_k$, and let the noise sequences be zero-mean Gaussian white noise, namely, process noise $\boldsymbol{\eta}_k \sim N\left(\mathbf{0}, \mathbf{A}_k\right)$ and measurement noise $\boldsymbol{\nu}_k \sim N\left(\mathbf{0}, \mathbf{B}_k\right)$, with variances $\mathbf{A}_k$ and $\mathbf{B}_k$ respectively. $\mathbf{M}_k$ is the measurement matrix defined by:

$$\mathbf{M}_k = \mathrm{diag}\left(\underbrace{\Delta\mathbf{r}_k^\intercal, \cdots, \Delta\mathbf{r}_k^\intercal}_{p}\right) \in \mathbb{R}^{p\times pq} \tag{18}$$

1) Linear Kalman Filter: LKF is an algorithm that estimates unknown states of a system by observing measurements and inaccuracies. The algorithm works in a two-step process: predict and update. With the prediction of the current states and their uncertainties, once the next measurement is observed, these prediction are updated using a weighted average. Firstly, denote the prediction of the state and its variance using $\hat{\mathbf{x}}_k^-$ and $\mathbf{P}_k^-$. Denote the update value of the state and its variance using $\hat{\mathbf{x}}_k$ and $\mathbf{P}_k$ with initialization of $\hat{\mathbf{x}}_0 = \mathrm{E}(\mathbf{x}_0)$ and $\mathbf{P}_0 = \mathrm{Var}(\mathbf{x}_0)$ respectively. For system (17), present LKF as follows:

$$
\begin{aligned}
\text{Predict:} \quad & \mathbf{P}_k^- = \mathbf{P}_{k-1} + \mathbf{A}_{k-1} \\
& \mathbf{H}_k = \mathbf{P}_k^-\mathbf{M}_k^\intercal\left(\mathbf{M}_k\mathbf{P}_k^-\mathbf{M}_k^\intercal + \mathbf{B}_k\right)^{-1} \\
& \hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1} \\
\text{Update:} \quad & \mathbf{P}_k = (\mathbf{E}_{pq} - \mathbf{H}_k\mathbf{M}_k)\mathbf{P}_k^- \\
& \hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{H}_k\left(\mathbf{y}_k - \mathbf{M}_k\hat{\mathbf{x}}_k^-\right) \\
& \hat{\mathbf{y}}_k = \mathbf{M}_k\hat{\mathbf{x}}_k
\end{aligned}
\tag{19}
$$

Once we get $\hat{\mathbf{x}}_k$ at each step, we update the Jacobian $\hat{\mathbf{J}}_k$. But, LKF is only suitable for linear systems, when estimating the nonlinear time-varying Jacobian matrix, it may cause estimation errors.

2) Unscented Kalman Filter: KF preforms the propagation of a Gaussian Random Variable (GRV) through the system dynamics while LKF is not good enough to propagate a GRV for a nonlinear system [26]. UKF selects a minimal set of sample points (so called sigma points) of system state $\mathbf{x}$ and guarantees the state distribution is again approximated by a GRV, using the Unscented Transformation (UT) [27], [28] which is an approach that maps system state $\mathbf{x}$ to sigma points $\boldsymbol{\chi}$, to fix the large error introduced by the first-order linearization of the nonlinear system.

Firstly, we rewrite (17) into the nonlinear form:

$$\mathbf{x}_k = \boldsymbol{g}(\mathbf{x}_{k-1}) + \boldsymbol{\eta}_k$$
$$\mathbf{y}_k = \boldsymbol{h}(\mathbf{x}_k) + \boldsymbol{\nu}_k \tag{20}$$

Initialize the mean and covariance of the state with $\hat{\mathbf{x}}_0 = \mathrm{E}(\mathbf{x}_0)$ and $\mathbf{P}_{x,0} = \mathrm{Var}(\mathbf{x}_0)$ respectively. Define a matrix $\boldsymbol{\chi} \in \mathbb{R}^{pq \times (2pq+1)}$ of $2pq + 1$ sigma vectors $\boldsymbol{\chi}_i \in \mathbb{R}^{pq}$:

$$\boldsymbol{\chi}_k^0 = \hat{\mathbf{x}}_k$$
$$\boldsymbol{\chi}_k^i = \hat{\mathbf{x}}_k + \left(\sqrt{(pq+\kappa)\,\mathbf{P}_{x,k}}\right)_i$$
$$\boldsymbol{\chi}_k^{i+pq} = \hat{\mathbf{x}}_k - \left(\sqrt{(pq+\kappa)\,\mathbf{P}_{x,k}}\right)_i \quad (21)$$

where $i = 1, \cdots, pq$ and $(\bullet)_i$ is the ith column of the matrix which should be decomposed using numerical method such as SVD and Cholesky decomposition. The coresponding first-order weights $W^{(m)}$ and second-order weights $W^{(c)}$ are

$$W_0^{(m)} = \frac{\kappa}{pq+\kappa}$$
$$W_0^{(c)} = \frac{\kappa}{pq+\kappa} + \left(1 - \alpha^2 + \beta\right)$$
$$W_i^{(m)} = W_i^{(c)} = \frac{\kappa}{2\,(pq+\kappa)}, \quad i = 1, \cdots, 2pq. \quad (22)$$

where $\kappa = \alpha^2\,(pq + \lambda) - pq$ is a scaling parameter. α determines the spread of the sigma points around $\hat{\mathbf{x}}_{k-1}$ which is usually set $1e^{-1} \leq \alpha \leq 1$. λ is a secondeary scaling parameter, usually set to 0. β is state distribution parameters, for Guassian distribution, usually set to 2.

The above sigma points in (21) are transformed through the state equation $\boldsymbol{g}(\bullet)$ and observation equation $\boldsymbol{h}(\bullet)$ of the system to obtain new sigma points $\boldsymbol{\gamma}$:

$$\boldsymbol{\chi}_k = \boldsymbol{g}\left(\boldsymbol{\chi}_{k-1}\right) = \boldsymbol{\chi}_{k-1}$$
$$\boldsymbol{\gamma}_k = \boldsymbol{h}\left(\boldsymbol{\chi}_k\right) = \mathbf{M}_k \boldsymbol{\chi}_k \quad (23)$$
$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2pq} W_i^{(m)} \boldsymbol{\gamma}_k^i$$

Then, present classical two-step KF frame to the new sigma points:

$$\text{Predict:} \quad \hat{\mathbf{x}}_k^- = \sum_{i=0}^{2pq} W_i^{(m)} \boldsymbol{\chi}_k^i$$
$$\mathbf{P}_{x,k}^- = \sum_{i=0}^{2pq} W_i^{(c)} \left(\boldsymbol{\chi}_k^i - \hat{\mathbf{x}}_k^-\right)\left(\boldsymbol{\chi}_k^i - \hat{\mathbf{x}}_k^-\right)^{\mathsf{T}} + \mathbf{A}_k$$
$$\text{Update:} \quad \mathbf{P}_{y,k} = \sum_{i=0}^{2pq} W_i^{(c)} \left(\boldsymbol{\gamma}_k^i - \hat{\mathbf{y}}_k^-\right)\left(\boldsymbol{\gamma}_k^i - \hat{\mathbf{y}}_k^-\right)^{\mathsf{T}} + \mathbf{B}_k$$
$$\mathbf{P}_{xy,k} = \sum_{i=0}^{2pq} W_i^{(c)} \left(\boldsymbol{\chi}_k^i - \hat{\mathbf{x}}_k^-\right)\left(\boldsymbol{\gamma}_k^i - \hat{\mathbf{y}}_k^-\right)^{\mathsf{T}}$$
$$\mathbf{K} = \mathbf{P}_{xy,k} \mathbf{P}_{y,k}^{-1}$$
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}\left(\mathbf{y}_k - \hat{\mathbf{y}}_k^-\right)$$
$$\mathbf{P}_{x,k} = \mathbf{P}_{x,k-1} - \mathbf{K}\mathbf{P}_{y,k}\mathbf{K}^{\mathsf{T}} \quad (24)$$

Once we get $\hat{\mathbf{x}}_k$ at each step, we update the Jacobian $\hat{\mathbf{J}}_k$.

C. Shape Servoing Controller

Let us assume that at the time instant k, the transformation matrix $\hat{\mathbf{J}}_k$ has been exactly estimated by the online estimator,

so that the shape-motion difference model satisfies:

$$\mathbf{s}_k = \mathbf{s}_{k-1} + \hat{\mathbf{J}}_k \cdot \Delta \mathbf{r}_k \quad (25)$$

In this section, we design the necessary motion command $\Delta \mathbf{r}_k$ to minimize the shape error between the measured feature $\mathbf{s}_k$ and a constant target feature $\mathbf{s}^*$. Define the deformation error as $\mathbf{e}_k = \mathbf{s}^* - \mathbf{s}_k$. Then, according to (25), we have

$$\mathbf{e}_k - \mathbf{e}_{k-1} = -\hat{\mathbf{J}}_k \Delta \mathbf{r}_k \quad (26)$$
$$\mathbf{e}_k + \mathbf{e}_{k-1} = 2\mathbf{e}_{k-1} - \hat{\mathbf{J}}_k \Delta \mathbf{r}_k \quad (27)$$

Our new shape control method also has the capability to limit the magnitude of $\Delta \mathbf{r}_k$, which helps to avoid drastic instantaneous movements during the shaping motions (this valuable property is particularly useful during the manipulation of soft/delicate objects). To this end, consider first the following performance index:

$$Q = \mathbf{e}_k^{\mathsf{T}} \mathbf{e}_k + \lambda \Delta \mathbf{r}_k^{\mathsf{T}} \Delta \mathbf{r}_k \quad (28)$$

where λ is the weight that controls the magnitude of $\Delta \mathbf{r}_k$. It can also guarantee the smoothness of $\mathbf{r}_k$. Note that if λ is too small, the system may oscillate or even lose stability.

By substituting (26) into (28), we have:

$$Q = \left(\mathbf{e}_{k-1} - \hat{\mathbf{J}}_k \Delta \mathbf{r}_k\right)^{\mathsf{T}} \left(\mathbf{e}_{k-1} - \hat{\mathbf{J}}_k \Delta \mathbf{r}_k\right) + \lambda \Delta \mathbf{r}_k^{\mathsf{T}} \Delta \mathbf{r}_k \quad (29)$$

The partial derivative of (29) along $\Delta \mathbf{r}_k$ yields:

$$\frac{\partial Q}{\partial \Delta \mathbf{r}_k} = -2\hat{\mathbf{J}}_k^{\mathsf{T}} \left(\mathbf{e}_{k-1} - \hat{\mathbf{J}}_k \Delta \mathbf{r}_k\right) + 2\lambda \Delta \mathbf{r}_k \quad (30)$$

By equating (30) to zero, we can compute the velocity command $\Delta \mathbf{r}_k$ as:

$$\Delta \mathbf{r}_k = \left(\lambda \mathbf{E}_q + \hat{\mathbf{J}}_k^{\mathsf{T}} \hat{\mathbf{J}}_k\right)^{-1} \hat{\mathbf{J}}_k^{\mathsf{T}} \mathbf{e}_{k-1} \quad (31)$$

Thus, at each time step, the incremental position command is calculated as follows:

$$\mathbf{r}_k = \mathbf{r}_{k-1} + \left(\lambda \mathbf{E}_q + \hat{\mathbf{J}}_k^{\mathsf{T}} \hat{\mathbf{J}}_k\right)^{-1} \hat{\mathbf{J}}_k^{\mathsf{T}} \mathbf{e}_{k-1} \quad (32)$$

Proposition. Consider that the model estimation algorithm (either LKF or UKF) exactly approximates the deformation Jacobian matrix such that $\mathbf{J}_k = \hat{\mathbf{J}}_k$. For this situation, the shape servo-controller (31) enforces a stable (passive) closed-loop system that locally minimises the error $\mathbf{e}_k$ to a steady-state area whose magnitude depends on the feasibility of the target feature.

Proof: From (31), we can have the expression:

$$\mathbf{e}_{k-1}^{\mathsf{T}} \hat{\mathbf{J}}_k = \Delta \mathbf{r}_k^{\mathsf{T}} \left(\lambda \mathbf{E}_q + \hat{\mathbf{J}}_k^{\mathsf{T}} \hat{\mathbf{J}}_k\right) \quad (33)$$

Let us define the discrete Lyapunov function and its finite difference as [29]:

$$V_k = \frac{1}{2} \mathbf{e}_k^{\mathsf{T}} \mathbf{e}_k \quad (34)$$

$$\Delta V_k = V_k - V_{k-1} = \frac{1}{2} \mathbf{e}_k^{\mathsf{T}} \mathbf{e}_k - \frac{1}{2} \mathbf{e}_{k-1}^{\mathsf{T}} \mathbf{e}_{k-1} \quad (35)$$

By substituting (26) and (27) into (35) we obtain:

$$\Delta V_k = \frac{1}{2}(\mathbf{e}_k + \mathbf{e}_{k-1})^\mathsf{T}(\mathbf{e}_k - \mathbf{e}_{k-1})$$

$$= -\left(\mathbf{e}_{k-1} - \frac{1}{2}\hat{\mathbf{J}}_k\Delta\mathbf{r}_k\right)^\mathsf{T}\hat{\mathbf{J}}_k\Delta\mathbf{r}_k$$

$$= -\mathbf{e}_{k-1}^\mathsf{T}\hat{\mathbf{J}}_k\Delta\mathbf{r}_k + \frac{1}{2}\Delta\mathbf{r}_k^\mathsf{T}\hat{\mathbf{J}}_k^\mathsf{T}\hat{\mathbf{J}}_k\Delta\mathbf{r}_k \qquad (36)$$

Then, substituting (33) into (36) yields:

$$\Delta V_k = -\Delta\mathbf{r}_k^\mathsf{T}\left(\lambda\mathbf{E}_q + \hat{\mathbf{J}}_k^\mathsf{T}\hat{\mathbf{J}}_k\right)\Delta\mathbf{r}_k + \frac{1}{2}\Delta\mathbf{r}_k^\mathsf{T}\hat{\mathbf{J}}_k^\mathsf{T}\hat{\mathbf{J}}_k\Delta\mathbf{r}_k$$

$$= -\Delta\mathbf{r}_k^\mathsf{T}\mathbf{L}_1\Delta\mathbf{r}_k = -\mathbf{e}_{k-1}^\mathsf{T}\mathbf{L}_2\mathbf{e}_{k-1} \leq 0$$

$$\mathbf{L}_1 = \lambda\mathbf{E}_q + \frac{1}{2}\hat{\mathbf{J}}_k^\mathsf{T}\hat{\mathbf{J}}_k$$

$$\mathbf{L}_2 = \hat{\mathbf{J}}_k\left(\lambda\mathbf{E}_q + \hat{\mathbf{J}}_k^\mathsf{T}\hat{\mathbf{J}}_k\right)^{-1}\mathbf{L}_1\left(\lambda\mathbf{E}_q + \hat{\mathbf{J}}_k^\mathsf{T}\hat{\mathbf{J}}_k\right)^{-1}\hat{\mathbf{J}}_k^\mathsf{T} \quad (37)$$

The stability (passivity) of the system is guaranted since (by definition) the Jacobian matrix $\hat{\mathbf{J}}_k$ has a full column rank, therefore, the matrix $\mathbf{L}_1$ is symmetric and positive definite. Now, to analyze the stability properties of the feedback shape error $\mathbf{e}_k$, let us note that the symmetric matrix $\mathbf{L}_2$ is only positive semi-definite. This implies that the magnitude of the shape feedback error $\|\mathbf{e}_k\|$ can only be minimised to a local region around the origin. For these types of overdetermined visual servoing control schemes, global asymptotic convergence of the error cannot be guaranteed [30]. ∎

Remark 2. The proposed velocity command (31) and Jacobian estimation algorithms (LKF, UKF) are entirely determined from the input and output data collected by the sensor feedback and have no relationship with the mathematical model and order of the controlled process.

D. Online Parameter Tuning

In general, the weight λ is constant, which makes the controller not adaptable to different control performance requirements. Meanwhile, elastic rods of different materials have different physical properties for manipulation. Some can be manipulated at high speed, while some are only suitable for chronic uniform deformation. Introducing parameter optimization criterion, the controller can be adaptively extended to various application scenarios.

Since we define $\mathbf{e}_k = \mathbf{s}^* - \mathbf{s}_k$ and by referring to (26) and (31), the closed-loop system can be obtained as follows:

$$\mathbf{e}_k = \mathbf{e}_{k-1} - \hat{\mathbf{J}}_k\left(\lambda\mathbf{E}_q + \hat{\mathbf{J}}_k^\mathsf{T}\hat{\mathbf{J}}_k\right)^{-1}\hat{\mathbf{J}}_k^\mathsf{T}\mathbf{e}_{k-1} \qquad (38)$$

There are several performance optimization criteria corresponding to diverse performance requirements. In this paper, three of them are introduced:

1) ISE criterion. Integral of the squared error (ISE) is one of the most well known criteria for obtaining optimal controller parameters [31], which penalizes both positive and negative errors.

$$H = \frac{1}{2}\sum_{k=1}^{\infty}\mathbf{e}_k^\mathsf{T}\mathbf{e}_k \qquad (39)$$

where, $\mathbf{e}_k$ defined in (38). According to (31) and (39), we can obtain the ISE-based parameter optimization criterion as:

$$\frac{\partial H}{\partial \lambda} = \sum_{k=1}^{\infty}\mathbf{e}_k^\mathsf{T}\hat{\mathbf{J}}_k\mathbf{Z}\hat{\mathbf{J}}_k^\mathsf{T}\mathbf{e}_{k-1} \qquad (40)$$

$$\mathbf{Z} = \left(\lambda\mathbf{E}_q + \hat{\mathbf{J}}_k^\mathsf{T}\hat{\mathbf{J}}_k\right)^{-2} \qquad (41)$$

2) IAE criterion. Another frequently used criteria, the integral of the absolute error (IAE), is shown as follows:

$$H = \sum_{k=1}^{\infty}\|\mathbf{e}_k\| \qquad (42)$$

Similar to the previous derivation, we can obtain the IAE-based parameter optimization criterion as:

$$\frac{\partial H}{\partial \lambda} = \sum_{k=1}^{\infty}\frac{\mathbf{e}_k^\mathsf{T}}{\sqrt{\mathbf{e}_k^\mathsf{T}\mathbf{e}_k}}\hat{\mathbf{J}}_k\mathbf{Z}\hat{\mathbf{J}}_k^\mathsf{T}\mathbf{e}_{k-1} \qquad (43)$$

where $\mathbf{Z}$ is defined by (41). Comparison of (40) and (43) shows that IAE is sensitive with small errors (smaller than one) while ISE works well on large errors.

3) JEU criterion. ISE and IAE only consider the performance requirements on errors but not meet higher standards of the system, such as the minimum overshoot, the peak time, the rising time, etc.. Therefore, in order to more comprehensively adjust the system transition performance and dynamic properties, JEU criterion which is a weighted sum of ISE and Integral Squared Controller Output [32], [33] is used here. In our case, the velocity command constraint is added:

$$H = \frac{1}{2}\sum_{k=1}^{\infty}\omega_1\mathbf{e}_k^\mathsf{T}\mathbf{e}_k + \omega_2\Delta\mathbf{r}_k^\mathsf{T}\Delta\mathbf{r}_k \qquad (44)$$

where ω_1 and ω_2 are positive scalar weight that satisfy $\omega_1 + \omega_2 = 1$. The parameter optimization criterion is:

$$\frac{\partial H}{\partial \lambda} = \sum_{k=1}^{\infty}\left(\omega_1\mathbf{e}_k^\mathsf{T}\hat{\mathbf{J}}_k - \omega_2\Delta\mathbf{r}_k^\mathsf{T}\right)\mathbf{Z}\hat{\mathbf{J}}_k^\mathsf{T}\mathbf{e}_{k-1} \qquad (45)$$

where $\mathbf{Z}$ is defined by (41). Besides, ISE can be seen as a simplified version of JEU.

For any of the above methods, the parameter λ is updated with the gradient descent rule:

$$\lambda_k = \lambda_{k-1} + \Delta\lambda, \qquad \Delta\lambda = -d \cdot \frac{\partial H}{\partial \lambda} \qquad (46)$$

where H is defined in (39), (42), (44) and d is a small positive weight which controls the update rate of λ.

IV. SIMULATION RESULTS

We consider a planar robot that rigidly grasps one end of an elastic rod, whose other end is static. A monocular vision sensor observes the manipulated cable and measures its 2D contour in real-time. The cable simulation is simulated as in [34] by using the minimum energy principle [35]. This simulator is publicly available at https://github.com/q546163199/shape_deformation/. All numerical simulations are implemented in MATLAB. Since the real feedback data point is in pixels, the simulated cable is designed to move within the range of 600px×600px.
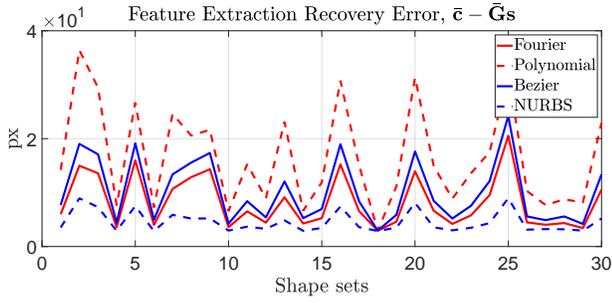
Fig. 4. Feature extraction comparison among polynomial, Bézier, NURBS and Fourier among 30 shape sets in the simulation.
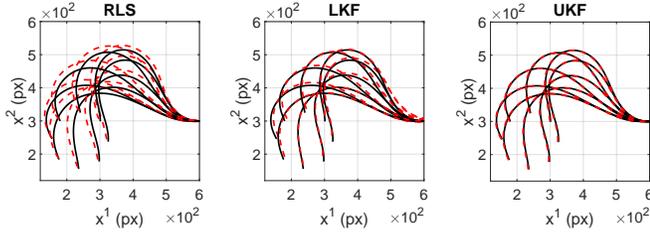

Fig. 5. Comparison between the visually measured cable profile (black solid line) and its approximation with NURBS series (red dashed line)

A. Feature Extraction Comparison

To verify the accuracy of the proposed shape feature, we compare three regression equations (9), (10), (11) with the Fourier method (12). Fig. 4 describes the error $(\bar{\mathbf{c}} - \bar{\mathbf{G}}\mathbf{s})$ between the original shape $\bar{\mathbf{c}}$ and the reconstruction shape $\bar{\mathbf{G}}\mathbf{s}$, which depicts the NURBS has the highest accuracy while the polynomial is the worst. The higher the polynomial order is, the worse the regression becomes. Table I shows that Fourier is the fastest. Although the NURBS is the slowest, it can be handled as long as the robot motion is slow enough. In the following sections, we uniformly use the NURBS with approximation order of 8.

TABLE I
COMPARISON RESULTS AMONG FOURIER, POLYNOMIAL, BÉZIER AND NURBS AMONG 30 SHAPE SETS

	G	**s**	**Order**	**Average time**	**Average error**
Fourier	200x18	18	4	0.002s	8.1965px
Polynomial	200x18	18	8	0.004s	16.5667px
Bézier	200x18	18	8	0.007s	10.1931px
NURBS	200x18	18	8	0.011s	4.5628px

B. Validation of the Jacobian Estimation

In this section, we compare the accuracy of three methods in estimating the Jacobian matrix, namely, LKF, UKF, and Recursive Least Square (RLS) given in (47). The cable is manipulated by the robot which moves along a circular trajectory whose center is $(0.4, 0.4)$ in the anticlockwise direction. At the very beginning of the movement, the robot moves the grasped cable in an initial sampling area to initialize the Jacobian matrix, based on which Jacobian matrix update with new input
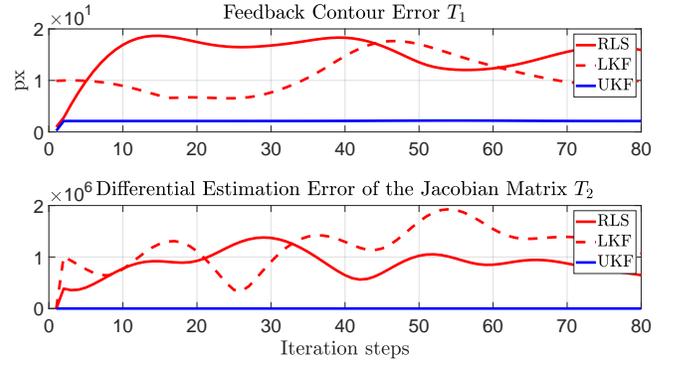

Fig. 6. Profiles of the cost functions T_1 and T_2 that are computed along the circular trajectory around the center $(0.4, 0.4)$.
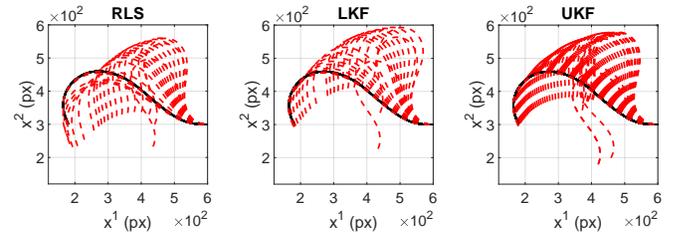

Fig. 7. Initial (red dashed line) and target (black solid line) configurations of the shape deformation simulation among RLS, LKF, and UKF.
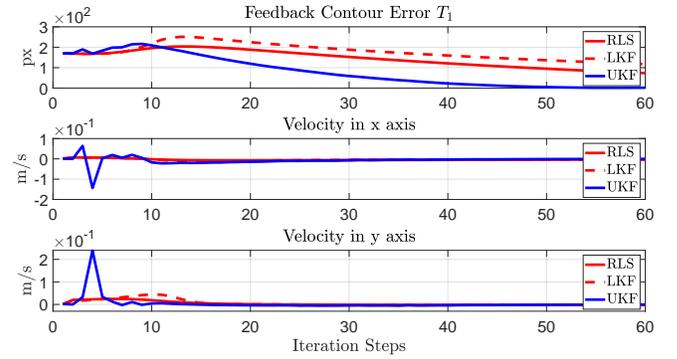

Fig. 8. Profiles of the cost function T_1 and velocity command $\Delta \mathbf{r}_k$ among RLS, LKF and UKF within manipulation task.
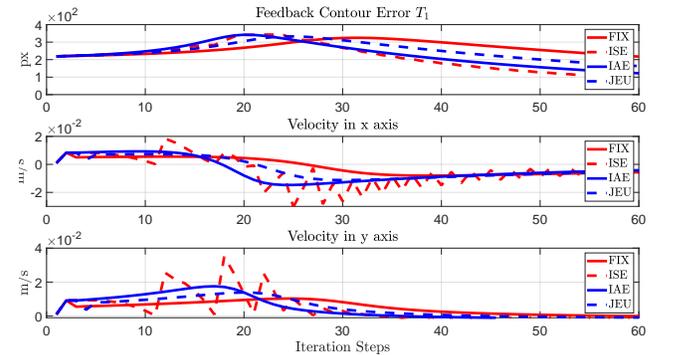

Fig. 9. Profiles of the cost function T_1 and velocity command $\Delta \mathbf{r}_k$ among FIX, ISE, IAE and JEU ($\omega_1 = 0.5, \omega_2 = 0.5$) tested in the LKF case.
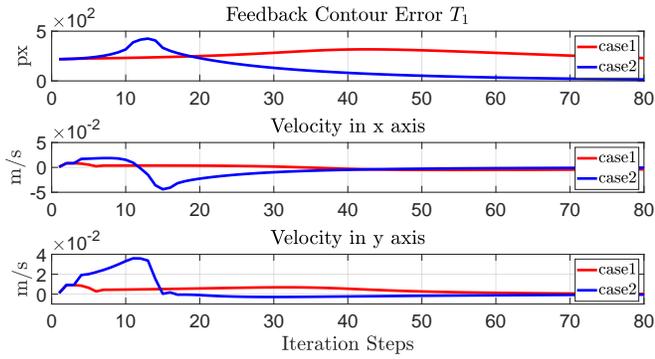
Fig. 10. Profiles of the cost function T_1 and velocity command $\Delta \mathbf{r}_k$ between two different weight values (case 1: $\omega_1 = 0.1, \omega_2 = 0.9$ and case 2: $\omega_1 = 0.9, \omega_2 = 0.1$) of JEU (44) tested in the LKF case.

and output data

$$\hat{\mathbf{J}}_k = \hat{\mathbf{J}}_{k-1} + \frac{\left(\mathbf{y}_k - \hat{\mathbf{J}}_{k-1}\Delta\mathbf{r}_k\right)\Delta\mathbf{r}_k^\mathsf{T}\mathbf{U}_{k-1}}{\lambda + \Delta\mathbf{r}_k^\mathsf{T}\mathbf{U}_{k-1}\Delta\mathbf{r}_k}$$

$$\mathbf{U}_k = \frac{1}{\lambda}\left(\mathbf{U}_{k-1} - \frac{\mathbf{U}_{k-1}\Delta\mathbf{r}_k\Delta\mathbf{r}_k^\mathsf{T}\mathbf{U}_{k-1}}{\lambda + \Delta\mathbf{r}_k^\mathsf{T}\mathbf{U}_{k-1}\Delta\mathbf{r}_k}\right) \qquad (47)$$

Fig. 5 demonstrates three measured shapes (black solid line) of the cable in circular motion, and the corresponding approximated shapes (red dashed line) based on the feedback feature vector $\mathbf{s}$. To assess the accuracy of the proposed algorithms, two errors are computed throughout the process:

$$T_1 = \|\bar{\mathbf{c}}_k^* - \bar{\mathbf{c}}_k\| \quad T_2 = \left\|\Delta\mathbf{s}_k - \hat{\mathbf{J}}_k\Delta\mathbf{r}_k\right\| \qquad (48)$$

Fig. 6 demonstrates the plots of T_1 and T_2 during the circular motion. Except for the UKF method, both errors increase when the robot is initializing the Jacobian matrix and decrease when the Jacobian matrix estimitor starts to work. Compared with RLS and LKF, the UKF method has a stronger ability to estimate the Jacobian matrix for both errors quickly converge to a smaller steady error under the UKF method.

C. Manipulation of Elastic Rods

In this section, we use adaptive controller (31) with fixed λ to allow the robot to manipulate the simulated cable into the desired shapes. Note that, to design feasible target cable shapes, a number of attempting computation is conducted. Fig. 7 depicts the progress of the cable deformation under the controllers based on RLS, LKF, and UKF. The red dashed curves represent the initial and transitional trajectories, and the black solid curve represents the target shape with parameters vector $\mathbf{s}^*$. Fig. 8 depicts the error T_1 and the velocity command $\Delta\mathbf{r}_k$. Table II clearly shows that UKF is the best method with the shortest convergence time and smallest deformation error.

TABLE II
RESULTS AMONG RLS, LKF AND UKF

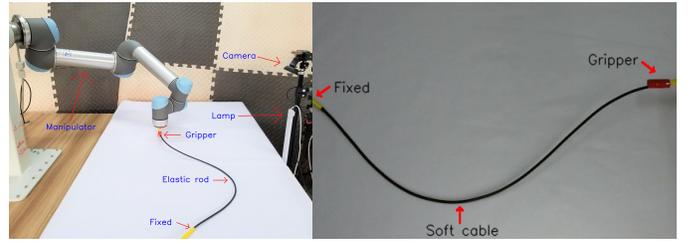|	RLS	LKF	UKF
Steps	534	488	158
Time(second)	55.54	48.83	17.91

Fig. 11. Experimental setup.

D. Comparison of Parameter Optimization

In this section, we compare the performance of various parameter optimization criteria, including FIX (fixed λ), ISE (39), IAE (42) and JEU (44)), based on LKF. Fig. 9 depicts the profiles of the error T_1 and the velocity command $\Delta\mathbf{r}_k$. Table III shows that ISE has shortest convergence time, while the FIX has the longest one, for ISE focuses on compensating errors so that the control command based on ISE heavily fluctuates in order to make the error converge as soon as possible. Meanwhile, ISE has a better performance than IAE for it is sensitive of large errors.

TABLE III
COMPARISON AMONG FIXED, ISE, IAE AND JEU TESTED IN THE LKF

	Fixed	ISE	IAE	JEU
Steps	386	186	233	291
Time(second)	36.42	17.18	21.92	31.17

To further illustrate the impact of different weight factors for JEU (44), we give two cases:

- case 1: $\omega_1 = 0.1, \omega_2 = 0.9$, namely, we pay more attention to the smoothness of the velocity command.
- case 2: $\omega_1 = 0.9, \omega_2 = 0.1$, namely, we pay more attention to the convergence speed of the error.

Fig. 10 shows that the error in case 1 converges slower with smoother velocity commands than case 2. This further verifies the effectiveness and feasibility of the parameter optimization criterion (44).

Remark 3. In the real applications, it is necessary to set the weight coefficients ω_1, ω_2 reasonably to prevent damage of the robotic arm caused by high-frequency chattering of the command signal.

V. EXPERIMENTAL RESULTS

In this section, we conduct various experiments using a UR5 robot constrained to xy-plane ($q = 2$) motion $\Delta\mathbf{r} = (\Delta r_x, \Delta r_y)$ defined in the base frame. An experimental video demonstrating our method can be downloaded here https://github.com/q546163199/experiment_video/raw/master/paper1/video.mp4. Fig. 11 shows our experiment setup. The rod's images are captured by a Logitech C270 camera and processed with a Linux-based PC at 30 fps and OpenCV. The results are displayed by MATLAB. We assess each algorithm's convergence speed by comparing the deformation of the moving cable every two frames.
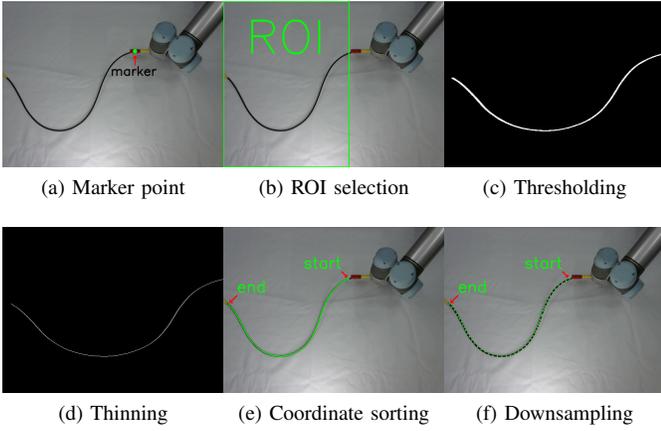
(a) Marker point   (b) ROI selection   (c) Thresholding

(d) Thinning   (e) Coordinate sorting   (f) Downsampling

Fig. 12.   Image processing steps.

Algorithm 1 Centerline of the object coordinate sorting

Input: Unordered centerline $\mathbf{p}_{in} = \left[p_{in}^0, \ldots, p_{in}^N\right]$; starting point p_{out}^0;

Output: Ordered centerline $\mathbf{p}_{out} = \left[p_{out}^0, \ldots, p_{out}^N\right]$;

1: $k = 0$, $j = 0$
2: $box\_size = 10$
3: **while** $j <= N - 1$ **do**
4:     **for** $i = 0$ to $column(\mathbf{p}_{in}) - 1$ **do**
5:         **if** $\left|p_{in}^i - p_{out}^j\right|_x \leq box\_size$ and $\left|p_{in}^i - p_{out}^j\right|_y \leq box\_size$ **then**
6:             Save $p_{box}^k = p_{in}^i$ with index i
7:             $k = k + 1$
8:         **end if**
9:     **end for**
10:     $index = \left\{i \mid \min \left\|p_{box}^k - p_{out}^j\right\|_2\right\}$
11:     $j = j + 1$
12:     $p_{out}^j = p_{in}^{index}$
13:     Delete p_{in}^{index} from $\mathbf{p}_{in}$
14:     $k = 0$
15:     Reset $\mathbf{p}_{box}$
16: **end while**

A. Image Processing

This section presents the relevant image processing for feature extraction and data sampling. The overall process (shown in Fig. 12) is as follows:

1) Segment the red area nearby the gripper based on HSV color space and abstract it as a green marker point (see Fig. 12a).
2) Segment the region of the interest (ROI) containing the rod according to the green marker point (see Fig. 12b).
3) Identify the rod in ROI, remove the noise, and obtain a binary image with the skeleton of the rod using OpenCV morphological opening algorithm (see Fig. 12c).
4) Get an unordered centerline by applying OpenCV GUO-HALL thinning algorithm to the rod skeleton (see Fig. 12d).
5) Design the **Algorithm** 1 to get the ordered centerline. Note that starting point is the closest point to the marker point on the centerline. (see Fig. 12e).
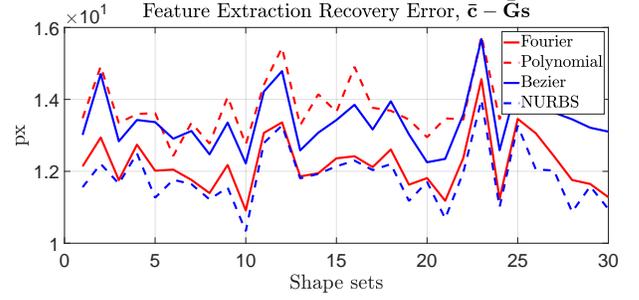

Fig. 13.   Feature extraction comparison among polynomial, Bézier, NURBS and Fourier among on 30 shape sets in the experiment.
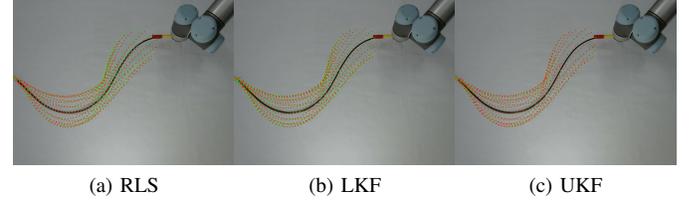

(a) RLS   (b) LKF   (c) UKF

Fig. 14.   Comparison between the visually measured cable profile (green dashed line) and its approximation with NURBS series (red dashed line)
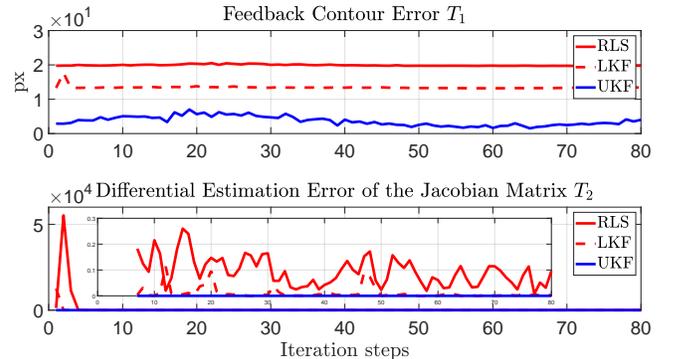
6) Equidistantly sample the ordered centerline to get a fixed number of data points (see Fig. 12f).

B. Feature Extraction Comparison

Similar to Section IV-A, we control the random movement of the UR5 to get 30 sets of centerline data; Fig. 13 shows these comparison results. We can see that NURBS's fitting accuracy is still the best, while the polynomial is the worst; this is consistent with the simulation results. It further verifies the effectiveness of the feature extraction algorithms designed in this paper. In the following sections, we use NURBS with approximation order of 8.

C. Validation of the Jacobian Estimation

Similar to Section IV-B, we test the accuracy of three methods (RLS, LKF, and UKF) for estimating the Jacobian matrix. In Fig. 14, the green dashed line is obtained by real feedback measurement, while the red dashed line represents the cable's estimation centerline with the NURBS series. From Fig. 15, we can see that UKF estimates are the best.

Fig. 15.   Profiles of the cost functions T_1 and T_2 that are computed along the circular trajectory.

(a) experiment1-RLS   (b) experiment2-RLS   (c) experiment3-RLS   (d) experiment4-RLS   (e) experiment5-RLS   (f) experiment6-RLS

(g) experiment1-LKF   (h) experiment2-LKF   (i) experiment3-LKF   (j) experiment4-LKF   (k) experiment5-LKF   (l) experiment6-LKF

(m) experiment1-UKF   (n) experiment2-UKF   (o) experiment3-UKF   (p) experiment4-UKF   (q) experiment5-UKF   (r) experiment6-UKF
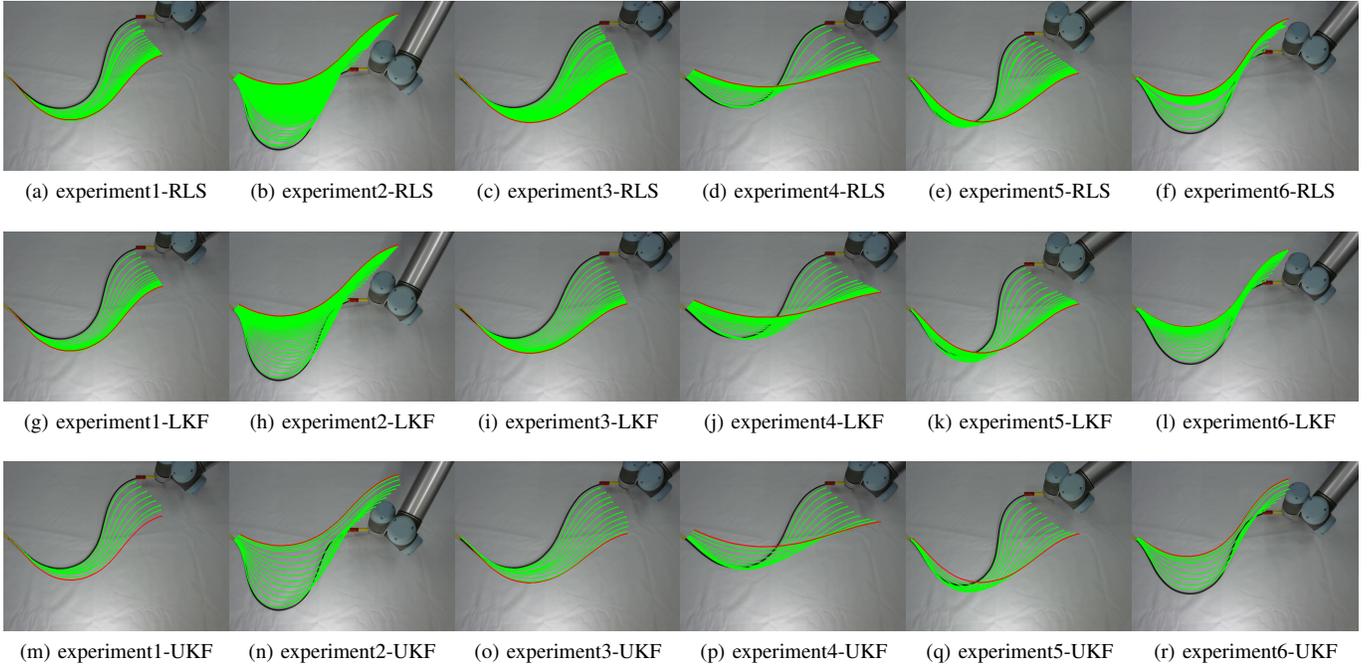
Fig. 16. Initial (black solid line), transition (green solid line) and target (red solid line) configurations in the six shape deformation experiments which have a variety of different initial and target shape with a single robot among RLS, LKF and UKF.

(a) experiment1 result

(b) experiment2 result

(c) experiment3 result

(d) experiment4 result

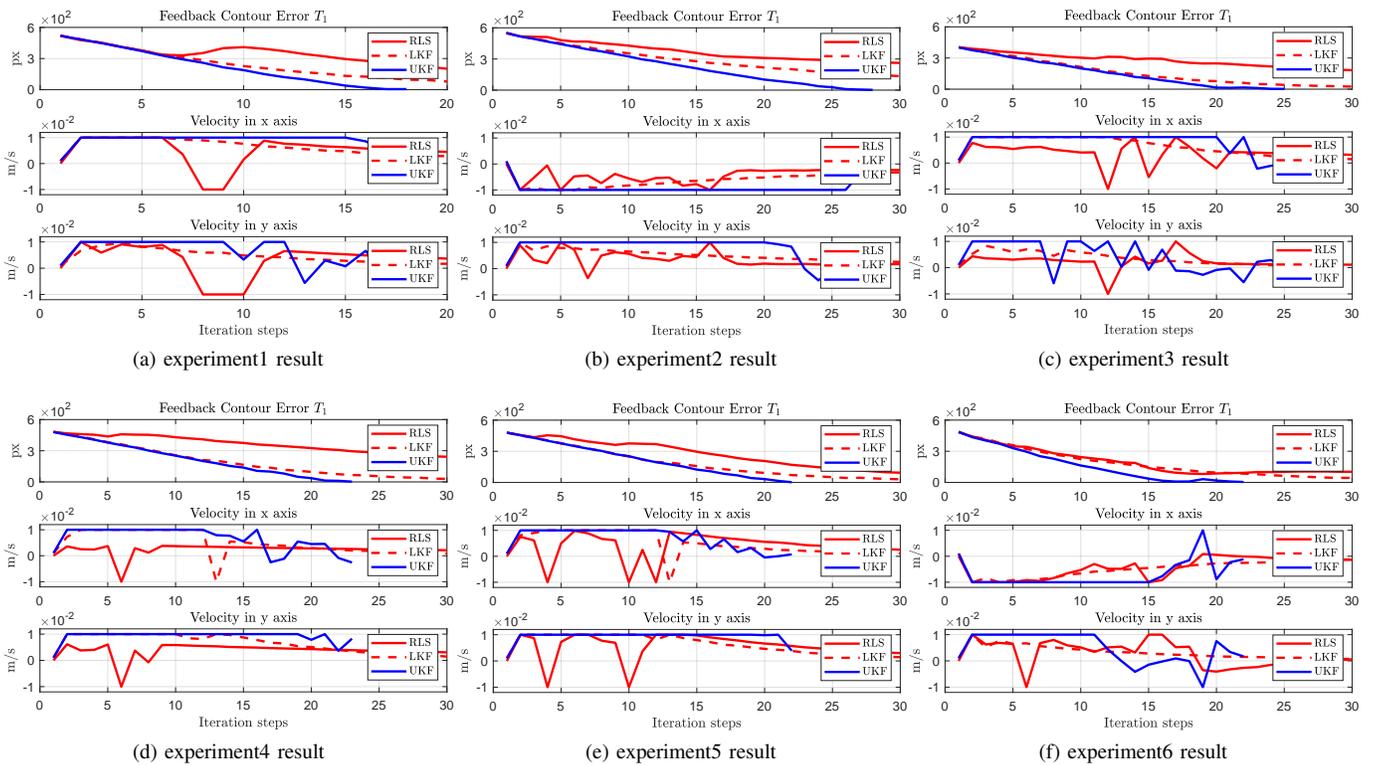(e) experiment5 result

(f) experiment6 result

Fig. 17. Profiles of the cost function T_1 and velocity command $\Delta \mathbf{r}_k$ among RLS, LKF and UKF within six shape deformation experiments.
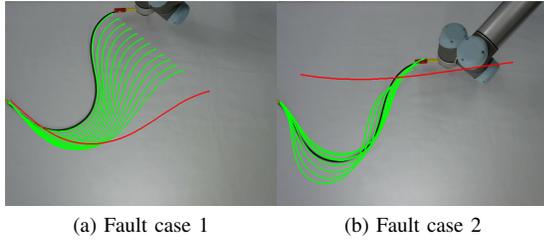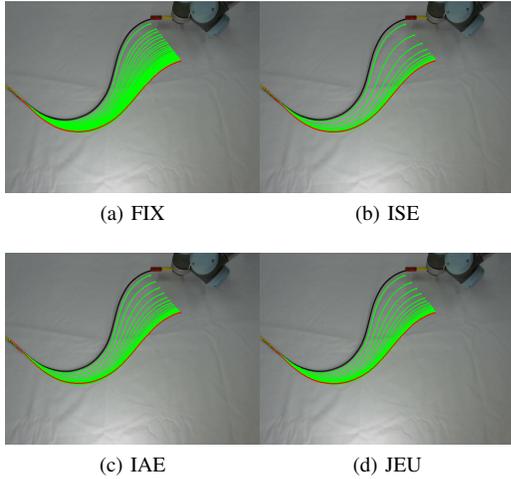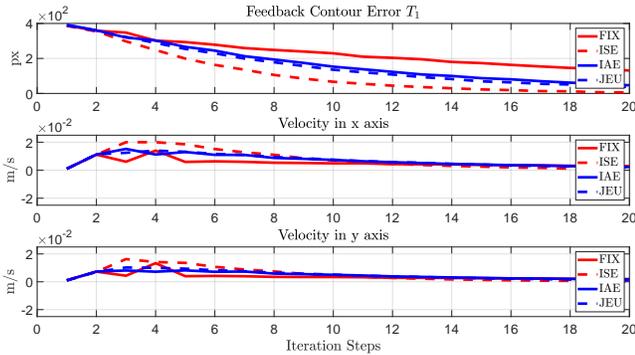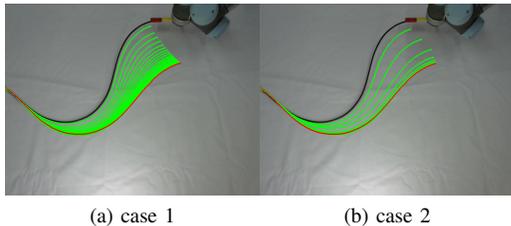
(a) Fault case 1   (b) Fault case 2

Fig. 18. Fault shape deformation experiments display.

(a) FIX   (b) ISE

(c) IAE   (d) JEU

Fig. 19. Initial (black solid line), transition (green solid line) and target (red solid line) configurations among FIX, ISE, IAE and JEU ($\omega_1 = 0.5, \omega_2 = 0.5$) tested in the LKF.

Fig. 20. Profiles of the cost function T_1 and velocity command $\Delta \mathbf{r}_k$ among FIX, ISE, IAE and JEU ($\omega_1 = 0.5, \omega_2 = 0.5$) tested in the LKF.

(a) case 1   (b) case 2

Fig. 21. Initial (black solid line), transition (green solid line) and target (red solid line) configurations between two conditions (case 1: $\omega_1 = 0.01, \omega_2 = 0.99$ and case 2: $\omega_1 = 0.99, \omega_2 = 0.01$) of JEU (44) tested in the LKF.

Fig. 22. Profiles of the cost function T_1 and velocity command $\Delta \mathbf{r}_k$ between two different weight values (case 1: $\omega_1 = 0.01, \omega_2 = 0.99$ and case 2: $\omega_1 = 0.99, \omega_2 = 0.01$) of JEU (44) tested in the LKF.

Meanwhile, no noticeable fluctuation of UKF plots reflects the strong adaptability of the UKF algorithm to effectively estimate the Jacobian matrix in different local regions.

D. Manipulation of Elastic Rods

Similar to Section IV-C, we use the proposed adaptive controller (31) with fixed λ to allow the robot to manipulate the simulated cable into the desired shapes. Note that, to design feasible target cable shapes, the robot grasping the cable moves to a predefined position and the target shape is computed using image processing and shape feature extraction algorithm. Then, the robot automatically goes back to the initial position and starts the shape deformation experiment. Considering safety, the upper limit of the saturation of the velocity command $\Delta \mathbf{r}_k$ is set to $0.01 m/s$.

We carry out six experiments with a variety of different initial and desired shapes, shown in Fig. 16, while Fig. 17 shows the profiles of T_1 and the velocity command $\Delta \mathbf{r}_k$. No apparent fluctuations in Fig. 16 shows that the proposed algorithms have excellent adaptability to the different starting conditions. Corresponding to the simulation results, Fig. 17 shows that UKF converges the fastest while FIX is the slowest. UKF can quickly respond to sudden deviations of the system and control the robot back to the normal state, showing the strong adaptability and robustness.

During the experiment, we found the following fault conditions, shown in Fig. 18.

1) If the initial shape is too different from the target shape or the initial sampling area, it is easy to cause singular control problems, seen in Fig. 18a.
2) If the target shape is similar to a straight line, as the proposed feature extraction algorithm is based on the least square method, so it is easy to cause singularity in the feature extractor, which makes the velocity command $\Delta \mathbf{r}_k$ too large and the system out of control, seen in Fig. 18b.

Therefore, in the practical applications, it is necessary to guarantee the initial shape around the initial sampling area and avoid singular shapes such as straight lines.

E. Parameter Optimization Comparison

Similar to Section IV-D, we validate the effectiveness of the proposed parameter optimization criteria based on LKF, shown in Fig. 19. Set the upper limit of the velocity command $\Delta\mathbf{r}_k$ as $0.02m/s$. Fig. 20 demonstrates that three adaptive methods (ISE, IAE, and JEU) converge to smaller errors and provide smoother velocity commands than FIX. Meanwhile, ISE converges the fastest and FIX converges the slowest. Since we set $\omega_1 = 0.5, \omega_2 = 0.5$ for JEU method, to some extent, the effect of JEU is similar to IAE. At the same time, the velocity command $\Delta\mathbf{r}_k$ fluctuates heavily under the ISE and keeps stable under the JEU. The correspondence between experimental results and simulation results demonstrates that the proposed parameter optimization criteria can effectively adjust the dynamic value of λ according to the control requirements.

Consistent with the simulation part, we also verify the effect of the JEU with different weight coefficients, shown in Fig. 21. The Difference is that, to highlight the impact of different weight coefficients, we design, case 1: $\omega_1 = 0.01, \omega_2 = 0.99$ and case 2: $\omega_1 = 0.99, \omega_2 = 0.01$. Fig. 22 shows that the error in case 1 converges slower with smoother velocity commands. In contrast, the velocity command of case 2 is relatively larger and close to the upper limit of saturation. From Fig. 22, we know different weight coefficients of JEU can bring significantly different control performance, thus, in the practical applications, they should be thoughtfully designed according to the system performance requirements.

VI. CONCLUSION

This paper presents a new visual servoing framework for automatically manipulating elastic rods to a desired configuration; It includes shape feature design, Jacobian matrix estimation, and online parameter optimization. First, new shape features (Bézier/NURBS) based on a regressive identification process are presented to characterize the object's contour. Second, we compare the performance of two KFs (LKF, UKF) in estimating the nonlinear time-varying Jacobian matrix. Then, the velocity command and its implementation are derived. To optimize control parameters, we utilize various performance criteria and an adaptive update law combined with a gradient descent rule. Finally, numerical and experimental results validate the effectiveness and feasibility of the proposed control method.

The proposed shape features (Bézier/NURBS) flexibly represent the high-dimension contour information with a low-dimension feature vector, in which NURBS has the highest accuracy. The identification method needs no artificial markers which makes it suitable for real-world applications. When estimating the shape Jacobian matrix, UKF performs better than LKF at nonlinear time-varying scenarios. The introduced parameter optimization criteria are able to meet different performance requirements and achieve online adaptive parameter adjustment. The sensorimotor model is estimated from visual feedback data without prior knowledge of the object deformation properties and camera calibration.

The proposed method also has many limitations. For instance, the manipulated object is limited to elastic materials (i.e. with a self-recovery ability), hence, the proposed approach might not be suitable for inelastic or non-homogeneous objects. The method is not able to judge if the desired configuration is reachable, which might lead to the failure during the task. Besides, the proposed centerline coordinate sorting algorithm has an inevitable delay; Thus, the algorithm is mainly applied to the low-speed movements. For the KF-based estimators, it should be assumed that both process noise and measurement noise are Gaussian noise with known distribution (in the simulations and experiments, we set both as constant matrices), which is hard to guarantee in practice. We also assume that the object is always within the camera's visible range. However, occlusions caused by the robot's movement or the object itself may sometimes happen.

As future work, we plan to manipulate objects into a more complex shapes, such as spatial 3D shapes or by using multiple robots. An algorithm with a complete feature detection will be designed to improve the system's robustness in the case of occlusion. We may combine neural networks (due to its strong nonlinear function approximation ability) with the manipulation tasks to obtain more accurate features and to determine the reachability of the desired shape.

REFERENCES

[1] A. Cherubini, *Manipulation of flexible objects*, ser. Encyclopedia of Robotics, M. H. Ang, O. Khatib, and B. Siciliano, Eds. Springer-Verlag, 2020.

[2] M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. Abbeel, "Bringing clothing into desired configurations with limited perception," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3893–3900.

[3] S. Tokumoto and S. Hirai, "Deformation control of rheological food dough using a forming process model," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2. IEEE, 2002, pp. 1457–1464.

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

[5] H. Wakamatsu, E. Arai, and S. Hirai, "Knotting/unknotting manipulation of deformable linear objects," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 371–395, 2006.

[6] H. M. Yip, Z. Wang, D. Navarro-Alarcon, P. Li, C. Greiffenhagen, T. H. Cheung, and Y.-H. Liu, "A collaborative robotic uterine positioning system for laparoscopic hysterectomy: Design and experiments," *Int. J. Med. Rob. Comp. Assist. Surg.*, vol. 16, no. 4, pp. 1–15, 2020.

[7] 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.

[8] 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," *The International Journal of Robotics Research*, vol. 33, no. 11, pp. 1462–1480, 2014.

[9] F. Chaumette, "Image moments: a general and useful set of features for visual servoing," *IEEE Transactions on Robotics*, vol. 20, no. 4, pp. 713–723, 2004.

[10] 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, 2019.

[11] M. Laranjeira, C. Dune, and V. Hugel, "Catenary-based visual servoing for tethered robots," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 732–738.

[12] ——, "Catenary-based visual servoing for tether shape control between underwater vehicles," *Ocean Engineering*, vol. 200, p. 107018, 2020.

[13] D. Navarro-Alarcon and Y.-H. Liu, "Fourier-based shape servoing: a new feedback method to actively deform soft objects into desired 2-d image contours," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 272–279, 2018.

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

[15] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," 2017.

[16] 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 Transactions on Robotics*, vol. 29, no. 6, pp. 1457–1468, 2013.

[17] D. Navarro-Alarcon and Y.-h. Liu, "A dynamic and uncalibrated method to visually servo-control elastic deformations by fully-constrained robotic grippers," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4457–4462.

[18] K. Hosoda and M. Asada, "Versatile visual servoing without knowledge of true jacobian," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, vol. 1. IEEE, 1994, pp. 186–193.

[19] J. A. Piepmeier, G. V. McMurray, and H. Lipkin, "Uncalibrated dynamic visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 143–147, 2004.

[20] J. Qian and J. Su, "Online estimation of image jacobian matrix by kalman-bucy filter for uncalibrated stereo vision feedback," in *IEEE International Conference on Robotics Automation*, 2002.

[21] X. Lv and X. Huang, "Fuzzy adaptive kalman filtering based estimation of image jacobian for uncalibrated visual servoing," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 2167–2172.

[22] H. Wang, B. Yang, J. Wang, X. Liang, W. Chen, and Y. Liu, "Adaptive visual servoing of contour features," *IEEE/ASME Trans. on Mechatronics*, vol. 23, no. 2, pp. 811–822, 2018.

[23] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of Intelligent and Robotic Systems*, vol. 3, no. 3, pp. 201–212, 1990.

[24] L. Piegl and W. Tiller, *The NURBS book*. Springer Science & Business Media, 2012.

[25] C. K. Chui, *Kalman filtering with real-time application*, 2009.

[26] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee, 2000, pp. 153–158.

[27] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *American Control Conference*, 1995.

[28] F. Gustafsson and G. Hendeby, "Some relations between extended and unscented kalman filters," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 545–555, 2011.

[29] S. Sarpturk, Y. Istefanopulos, and O. Kaynak, "On the stability of discrete-time sliding mode control systems," *IEEE Transactions on Automatic Control*, vol. 32, no. 10, pp. 930–932, 1987.

[30] J.-J. Slotine and W. Li, *Applied Nonlinear Control*, 1st ed. Upper Saddle River, NJ: Prentice Hall, 1991.

[31] M. Zhuang and D. P. Atherton, "Automatic tuning of optimum pid controllers," *Control Theory Applications Iee Proceedings D*, vol. 140, no. 3, pp. 216–224, 1993.

[32] W. Xiong, B. Xu, and Q. Zhou, "Study on optimization of pid parameter based on improved pso," *Computer Engineering*, vol. 31, no. 24, pp. 41–43, 2005.

[33] S. Das, I. Pan, K. Halder, S. Das, and A. Gupta, "Lqr based improved discrete pid controller design via optimum selection of weighting matrices using fractional order integral performance index," *Applied Mathematical Modelling*, vol. 37, no. 6, pp. 4253–4268, 2013.

[34] H. Wakamatsu, "Modeling of linear objects considering bend, twist, and extensional deformation," *Proc.of IEEE Int.conf.robotics Automation*, 1995.

[35] D. Fleisch and L. Kinnaman, "A student's guide to lagrangians and hamiltonians," 2015.