This is the Pre-Published Version.

This is an Accepted Manuscript of an article published by Taylor & Francis in Advanced Robotics on 23 Nov 2021 (Published online), available at http://www.tandfonline.com/10.1080/01691864.2021.2004222

Towards Latent Space Based Manipulation of Elastic Rods using Autoencoder Models and Robust Centerline Extraction

Jiaming Qi, Guangfu Ma, David Navarro-Alarcon, Haibo Zhang and Yueyong Lyu

***Abstract*— The automatic shape control of deformable objects is an open (and currently hot) manipulation problem that is challenging due to the object's high-dimensional shape information and its complex physical properties. As a feasible solution to these issues, in this paper, we propose a new methodology to automatically deform elastic rods into 2D desired shapes. For that, we present an efficient method that uses the Deep Autoencoder Network (DAE) to compute a compact representation of the object's infinite dimensional shape. To deal with the (typically unknown) properties, we use an online algorithm that approximates the sensorimotor mapping between the robot's configuration and the object's shape features. Our new approach has the capability to compute the rod's centerline from raw visual data and in real-time. This is done by introducing an innovative algorithm based on the self-organizing network (SOM). The effectiveness of the proposed method is thoroughly validated with numerical simulations and experiments.**

I. Introduction

The capability to automatically control the shape of soft objects with robot manipulators is highly valuable in many applications, such as food processing [1], robotic surgery [2], cable assembly [3], and household works [4]. Although has been achieved in recent years, shape control is still an open research problem [5]. One of the most crucial issues that hamper the implementation of these type of tasks is difficulty to obtain a meaningful feedback representation of the object's configuration in real-time. However, due to the intrinsic high-dimensional nature of deformable objects, standard vision-based control algorithms (e.g. based on simple point features) cannot be used as these cannot properly capture the objects' state. In this work, we aim to provide an efficient solution to this problem.

Different from operations on rigid objects, as soft objects have infinite dimensional geometric information, it is necessary to design a simple and effective feature extractor for shape deformation [6]. At present, the traditional methods are roughly divided into two categories, local descriptors and global descriptors. Local descriptors utilized centroid/distance/angle/curvature [7] around a point to describe geometric characteristics, however they reduced potential connections between features. Thus, researchers began to focus on the global descriptors. Point Feature Histograms (PFH) [8] formed a multi-dimensional histogram that can represent the overall shape of soft objects and robust to sampling density and noise outliers. Afterward, [9] improved PFH into the Fast Point Feature Histograms (FPFH) which only determined between the query and neighboring points and reduced computation time. Then, [10] presented the extended FPFH which only computed the features between the centroid and other points, further enhanced real-time. In addition, [11] presented an innovative feedback representation of the object's contour based on a truncated Fourier series and effectively used it in the shape deformation issue. [12] presented a more general shape representation framework based on the linearly parametrized regression model on the basis of [11].

Learning-based solutions have received considerable attention because they are usually more effective to describe the soft objects than traditional methods in the latent space [13]. [6] took the force and position measurement of the three-finger gripper as input, and the contour as output to train the neural network, which can predict shapes that have never been seen before without any prior knowledge about object's materials. [14] proposed a novel coarse-to-fine shape representation using Spatial Transformer Networks, which showed better generalization and without expensive ground truth observations. Growing Neural Gas [15] was used to represent the complex shapes of soft objects, as its adaptability coupled with intrinsic denoising and down-sampling properties. [16] designed a feature extractor based on the Convolutional Autoencoder Neural Network which aimed to map the cognitive problem to the low-dimensional latent space and avoided unnecessary calculation due to the sensor outputs' processing and allowed us to represent complex shapes rapidly.

However, above methods generally require the fixed-length ordered data. Different from the simulator [12] that can meet the requirements, in the experiment, we usually use the *OpenCV/findcontour* or the thinning algorithms [17] [18] to get the contour [11] and centerline [12] (this paper focuses on), respectively. This leads to the data obtained cannot meet the standard. In this paper, we use a self-organizing network (SOM) [19] to extract the centerline, which has the nature of directly generating a fixed-length equidistantly arranged centerline (without thinning and down-sampling process, greatly saving time than the previous algorithm [12]).

This work is supported in part by the Germany/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the German Academic Exchange Service under grant G-PolyU507/18, in part by the Research Grants Council of Hong Kong under grant number 14203917, in part by the Key-Area Research and Development Program of Guangdong Province 2020 under project 76.

J. Qi, G. Ma and Y. Lyu are with the Harbin Institute of Technology, Harbin, 150001, China.

D. Navarro-Alarcon is with the Hong Kong Polytechnic University, Hung Hom, KLN, Hong Kong. Corresponding author: dna@ieee.org

Haibo Zhang is with the Beijing Institute of Control Engineering, Beijing, 100190, China. National Key Laboratory of Science and Technology on Space Intelligent Control, Beijing, 100190, China.

It is regarded as the first comprehensive description of the shape deformation [20]. However, these methods needed to identify the object's physical properties, which hindered its application in the industrial field. Algorithms based on the latent space has made significant progress in the field of shape deformation, as it can extract effective low-dimensional features from high-dimensional data space [21]. [22] used Convolutional Neural Network (CNN) to build the inverse kinematics model of the rope, taking a series of images of the human manipulating the rope as input, and training in a self-supervised manner so that the robot can finally manipulate the rope to the goal configurations. [23] learned the physical model of the soft object in the latent space without any prior knowledge of the object in the learning process. [14] utilized CNN to estimate the rope's state, combined with model predictive control, a series of optimized action sequences were designed. The above-mentioned methods illustrate the effectiveness of the latent space algorithms in the field of shape deformation.

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

1) We efficiently represent the soft objects using a fixed-length feature which is based on the Deep Autoencoder Network (DAE).
2) We present a robust SOM-based centerline extraction algorithm due to its strong clustering power.
3) We report detailed simulations and experimental results (dual UR5) to validate the proposed method, better than previous experiment (single UR5) [12].

According to our knowledge, this is the first time that a shape servo-controller uses DAE to establish an explicit shape servo-loop.

The rest of this paper is organized as follows: Section II presents the preliminaries. Section III presents the proposed overall deformation control implementation process. Section IV and Section V present various visually servoed deformation tasks of elastic rods. Section VI gives final conclusions and future work.

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.

The goal of this paper is to study the deformation control scheme of robot manipulating soft objects based on visual servoing. In order to enable readers to better understand this article, we first give the following premise conditions:

- Use a fixed camera to measure the centerline of the elastic rod, namely, eye-to-hand configuration (depicted in Fig. 1), and the coordinates obtained are denoted as:

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

where N represents the number of points on the extracted centerline, u_i and v_i represents the coordinates of the ith $(i = 1, \cdots, N)$ point in the image frame.

- Before the experiment, the robot has tightly grasped the soft object, namely, we do not consider the problem of object grasping. Meanwhile, there is no measurement loss during the manipulation process, and no occlusion occurs.
- The robot supports velocity control mode, and can accurately execute the given desired kinematic controls $\Delta\mathbf{r}_k \in \mathbb{R}^q$ [24], and satisfy the incremental position motions $\mathbf{r}_k = \mathbf{r}_{k-1} + \Delta\mathbf{r}_k$.
- The robot manipulates soft objects at low speeds, so the shape is uniquely determined by elastic potential energy.

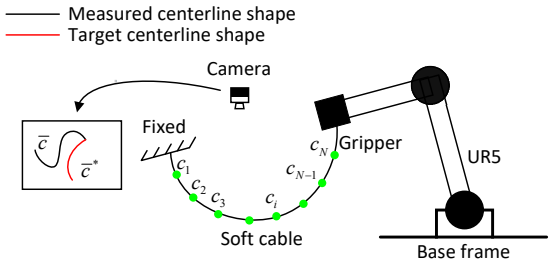


Fig. 1: Schematic diagram of soft object shape deformation, the camera is utilized to determine shape feature $\mathbf{s}$ in real time, and automatically move the robot to deform the soft object into the target feature $\mathbf{s}^*$ within the controller.

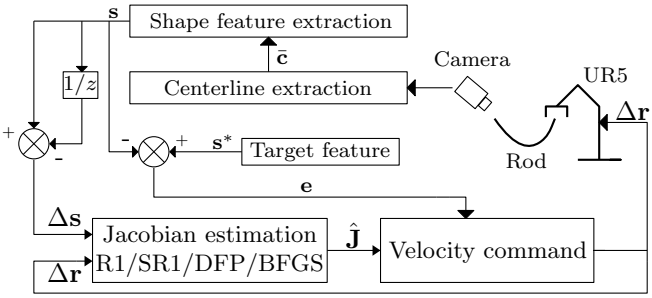


Fig. 2: The overall implementation flow of the proposed shape deformation strategy.

Problem Statement. Without any prior knowledge of the object's physical characteristics, designing a model-free vision-based controller which commands the robot to deform the elastic rod into the desired configuration in the 2D image space.

III. METHODS

Fig. 2 shows the overall implementation flow of the proposed shape deformation strategy, and the centerline extraction will be introduced in Section V-A.

A. Feature Extraction

For shape deformation, the most direct way is to design a controller to make all the coordinates $\bar{\mathbf{c}}$ of the soft object reach the desired 2D shape. The main problem of this method is that due to the huge dimensions of the original coordinates $\bar{\mathbf{c}}$, the real-time performance of the control system is poor, the deformation control accuracy is low, and it may even cause many adverse effects such as loss of control. Thus, designing a shape feature extraction algorithm for soft objects to decrease the feature dimension on the premise of effectively describing the soft object's geometric information is necessary.

In this article, we use DAE to explore the representation of elastic rods, and extract the shape features $\mathbf{s} \in \mathbb{R}^p$ from 2D centerline $\bar{\mathbf{c}} \in \mathbb{R}^{2N}$. DAE is a type of artificial neural network with three or more layers trained in the unsupervised-learning manner. It can automatically learn features from unlabeled data and give a better description than the original data. The traditional DAE [25] consists of an Encoder and Decoder, shown in Fig. 3.

Encoder. The Encoder $\mathbf{s} = \mathbf{f}(\bar{\mathbf{c}})$ reduces the dimension of input vector $\bar{\mathbf{c}}$ from one hidden layer to the next, optimally resulting in a set of pseudo-orthogonal features $\mathbf{s} \in \mathbb{R}^p$ (usual, $p \ll 2N$) with minimal information loss, so that the Decoder is able to reconstruct the input $\bar{\mathbf{c}}$. Its typical form is an affine mapping followed by a nonlinear function.

Decoder. The Decoder $\hat{\bar{\mathbf{c}}} = \mathbf{g}(\mathbf{s})$ maps the resulting shape feature $\mathbf{s}$ back to a reconstructed vector $\hat{\bar{\mathbf{c}}}$. Its typical form is again an affine mapping that is optionally followed by a nonlinear function.

The work flow of DAE is described as follows: Consider the centerline $\bar{\mathbf{c}}$, the DAE transforms it into a feature vector $\mathbf{s}$ with lower dimension than that of $\bar{\mathbf{c}}$, and then uses the backpropagation algorithm to train the network (minimize the reconstruction error L_{DAE}, defined in (2)), finally decodes $\mathbf{s}$ back to $\bar{\mathbf{c}}$. The obtained dimension reduction data is used as the shape feature $\mathbf{s}$.

$$L_{DAE} = L\left(\bar{\mathbf{c}}, \hat{\bar{\mathbf{c}}}\right) \tag{2}$$

where $\bar{\mathbf{c}}$ and $\hat{\bar{\mathbf{c}}}$ are the input and reconstruction output, and L_{DAE} is usually a square loss function. For DAE, we need to emphasize that we don't care about the output $\hat{\bar{\mathbf{c}}}$ (reproduce the input anyway), we care about the mapping from input to encoding. Once the DAE is trained, only the Encoder is deployed in applications, providing a smaller dimension feature vector. Simultaneously, the Decoder is only used as a means to train the Encoder and is disregarded after training. At present, there are various neural networks that can form DAE. In this paper, we use Multi-Layer Perceptron (MLP) as the basic blocks of the DAE due to its ability to efficiently handle 2D data and for speed and simplicity.

At this stage, what needs to be explained to readers is that when choosing the value of p is large or small, there needs to exist a trade balance. A small p will make the system have better controllability (given the finite robot inputs), whereas a large p will enhance the representation

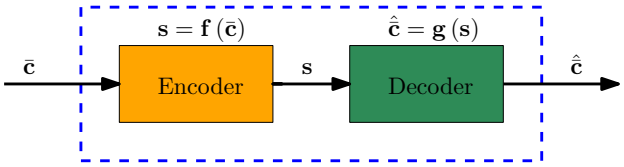


Fig. 3: Schematic diagram of DAE structure

accuracy of centerline. In the simulation and experiment, we will compare the shape representation capabilities in various cases of p.

Here, we compare DAE with PCA [26]:

- DAE is more suitable for extensive data than PCA.
- DAE with a single-layer network and a linear function as the activation function has the same performance as PCA, but DAE with a multi-layer network and nonlinear functions as the activation functions can have a better understanding.
- The reconstruction accuracy of the PCA has an upper bound when considering the dimension limitation (usual, $L_{DAE} < 1$). In contrast, for lots of training data, DAE's reconstruction accuracy is close to 1 ($L_{DAE} \approx 1$), theoretically.

B. Robust SOM-based Centerline Extraction Algorithm

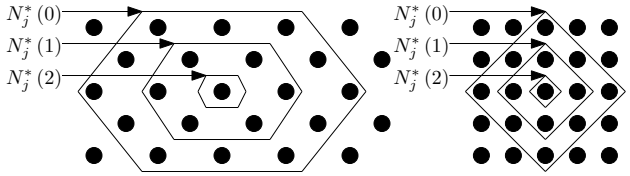
In this section, we design a robust SOM-based centerline extraction algorithm. It should be noted that the input of SOM is the white area (actually the elastic rod on the binary picture) in Fig. 13b. The points in the white area is defined by $\bar{\mathbf{m}} = [\mathbf{m}_1^\intercal, \ldots, \mathbf{m}_M^\intercal]^\intercal \in \mathbf{R}^{2M}$, $\mathbf{m}_i \in \mathbf{R}^2$ represents coordinates of each point under the image picture, and $M \gg N$. Therefore, we utilize the clustering nature of SOM to obtain a fixed-length centerline directly from the white area, namely, $SOM : 2M \to 2N$, shown in Fig. 13d.

Comparison of old and the proposed algorithm as follows:

- The old algorithm [12] sorts all points first, and performs down-sampling processing. Therefore, when there are numerous points, the processing speed will slow down.
- The proposed algorithm gets the fixed-length centerline first, then executes sorting, this greatly increases the speed.

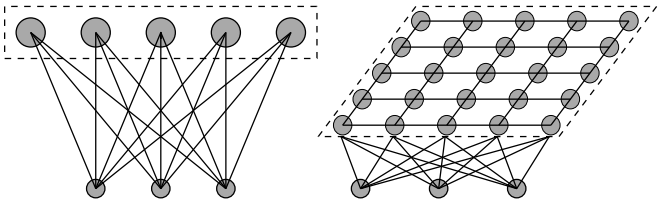
The topological structure of SOM has two layers, the input layer and the competition layer. There are many kinds of competition layer, such as a one-dimensional linear array and two-dimensional planar array, shown in Fig. 5. In this paper, we use a one-dimension linear array.

Competitive learning (CL) [27] is the most commonly used learning strategy in SOM. The most classic CL is the "Winner-Take-All" algorithm. However, this algorithm only updates the winning neuron, the network update speed is slow, resulting in poor clustering performance. Therefore, for improving the above problems, the concept of winning field is introduced based on CL. The winning field is the adjustment area around the winning neuron, and can be hexagonal or rectangular, as shown in Fig. 4. The winning



(a) Hexagonal field (b) Rectangular field

Fig. 4: Two kinds of winning field.



(a) One-dimensional linear array (b) Two-dimensional planar array

Fig. 5: SOM output array.

field is initially set to be very large and as the number of training increases, it will shrink from large to small and eventually shrink to zero. In the SOM, the neurons in winning field adjust their weights according to their distance from the winning neuron.

The SOM's implementation are divided into six steps.

1) Assign initial values to the weight vectors of the competition layer and normalize them, then, getting $\hat{\mathbf{W}}_j, j = 1, 2, \ldots, N$. Initialize winning field $N_j^*(0)$, and learning rate $\alpha(k, d)$ with $\alpha(0)$.
2) Randomly select an input vector $\mathbf{m}_i$ from the data set $\bar{\mathbf{m}}$ and normalize, then, getting $\hat{\mathbf{m}}_i$, $i = 1, 2, \ldots, M$.
3) Calculate the Euclidean distance (3) between the sample $\hat{\mathbf{m}}_i$ and the weight vector $\hat{\mathbf{W}}_j$, the neuron with the smallest distance wins the competition, marked by the winning neuron.

$$\left\|\hat{\mathbf{m}}_i - \hat{\mathbf{W}}_j^*\right\| = \min_{j \in 1,\ldots,N} \left\{\left\|\hat{\mathbf{m}}_i - \hat{\mathbf{W}}_j\right\|\right\} \tag{3}$$

4) Determine the weight adjustment domain at time k with the neuron j^* as the center. The initial domain $N_j^*(0)$ is generally chosen to be larger. As the training time increases, the winning domain $N_j^*(k)$ gradually shrinks to zero, shown in Fig. 4.
5) Adjust the weights of all neurons in the winning field according to the formula (4).

$$\begin{aligned} \hat{\mathbf{W}}_m(k+1) &= \hat{\mathbf{W}}_m(k) + \alpha(k, d)\left(\hat{\mathbf{m}}_i - \hat{\mathbf{W}}_m(k)\right) \\ \alpha(k, d) &= \alpha_1(k) e^{-d} \\ \alpha_1(k) &= \alpha(0)\left(1 - \frac{k}{T}\right) \end{aligned} \tag{4}$$

where $m \in N_j^*(k)$, k is kth step, T is total iteration steps, d is lateral distance which represents Euclidean distance between the winning neuron j^* and the jth neuron in the winning field.

6) Determine whether to stop according to whether the learning rate $\alpha(k, d)$ decays to zero or lower than the threshold. If not, return to step 2 to execute the loop.

After training, $\hat{\mathbf{W}} \in \mathbb{R}^{N \times 2}$ is the fixed-length centerline with the isometric arrangement.

Remark 1. The proposed SOM-based centerline extraction algorithm is used in the experiment, not for simulation.

C. Approximation of the Local Deformation Model

Since we consider regular (i.e. mechanically well-behaved) elastic objects, thus, the centerline $\bar{\mathbf{c}}$ is extremely dependent on the robot command $\mathbf{r} \in \mathbb{R}^q$ ($\mathbf{r}$ can be defined as the joint angles or end-effector's pose, we use later in this paper). The relationship between $\bar{\mathbf{c}}$ and $\mathbf{r}$ can be represented by the following unknown function (5).

$$\bar{\mathbf{c}} = \mathbf{h}(\mathbf{r}) \tag{5}$$

Refer to Section III-A, we have the mapping, $\mathbf{s} = \mathbf{f}(\bar{\mathbf{c}})$, thus, based on (5), the overall kinematics model from robot command $\mathbf{r}$ to shape feature $\mathbf{s}$ can be constructed as follows:

$$\mathbf{s} = \mathbf{f}(\mathbf{h}(\mathbf{r})) \tag{6}$$

Differentiating (6) with respect to time variable t yields

$$\dot{\mathbf{s}} = \mathbf{J}(t)\dot{\mathbf{r}} \tag{7}$$

where $\mathbf{J}(t) = \partial\mathbf{s}/\partial\mathbf{r} \in \mathbb{R}^{p \times q}$ represents a Jacobian-like matrix that describes the mapping between the feature change speed $\dot{\mathbf{s}}$ and the velocity command $\dot{\mathbf{r}}$, and because the properties of soft objects are unknown, the analytical form of $\mathbf{J}(t)$ cannot be obtained. We discrete (7) yields the first-order discretization format as follows:

$$\mathbf{s}_k = \mathbf{s}_{k-1} + \mathbf{J}_k \Delta\mathbf{r}_k \tag{8}$$

where $\Delta\mathbf{r}_k = \mathbf{r}_k - \mathbf{r}_{k-1} \in \mathbf{R}^q$. Because this paper focuses on the application of DAE as feature extraction, thus, we use simple Broyden algorithms to computes local approximations of $\mathbf{J}_k$ in real-time. Define the following differential signal:

$$\mathbf{y}_k = \mathbf{s}_k - \mathbf{s}_{k-1} \quad \mathbf{u}_k = \mathbf{r}_k - \mathbf{r}_{k-1} \tag{9}$$

The Broyden algorithms are as follows:

1) R1 update formula [28]:

$$\hat{\mathbf{J}}_k = \hat{\mathbf{J}}_{k-1} + \frac{\left(\mathbf{y}_k - \hat{\mathbf{J}}_{k-1}\mathbf{u}_k\right)\mathbf{u}_k^\intercal}{\mathbf{u}_k^\intercal \mathbf{u}_k} \tag{10}$$

This is the simplest form with simple structure and fast calculation speed.

2) SR1 update formula [28]:

$$\hat{\mathbf{J}}_k = \hat{\mathbf{J}}_{k-1} + \frac{\left(\mathbf{y}_k - \hat{\mathbf{J}}_{k-1}\mathbf{u}_k\right)\left(\mathbf{y}_k - \hat{\mathbf{J}}_{k-1}\mathbf{u}_k\right)^\intercal}{\mathbf{u}_k\left(\mathbf{y}_k - \hat{\mathbf{J}}_{k-1}\mathbf{u}_k\right)^\intercal} \tag{11}$$

The structure of SR1 is similar to R1, but the calculation accuracy is higher.

3) DFP update formula [29]:

$$\hat{\mathbf{J}}_k = \hat{\mathbf{J}}_{k-1} + \frac{\left(\mathbf{y}_k - \hat{\mathbf{J}}_{k-1}\mathbf{u}_k\right)\mathbf{y}_k^\intercal + \mathbf{y}_k\left(\mathbf{y}_k - \hat{\mathbf{J}}_{k-1}\mathbf{u}_k\right)^\intercal}{\mathbf{u}_k\mathbf{y}_k^\intercal} - \frac{\mathbf{y}_k^\intercal\mathbf{y}_k}{\|\mathbf{u}_k\mathbf{y}_k^\intercal\|}\left(\mathbf{y}_k - \hat{\mathbf{J}}_{k-1}\mathbf{u}_k\right)\mathbf{u}_k^\intercal \tag{12}$$

DFP is a rank 2 quasi-Newton method, which is one effective method for solving nonlinear optimization.

4) BFGS update formula [30]:

$$\hat{\mathbf{J}}_k = \hat{\mathbf{J}}_{k-1} - \frac{\hat{\mathbf{J}}_{k-1}\mathbf{u}_k\mathbf{u}_k^\intercal\hat{\mathbf{J}}_{k-1}^\intercal}{\mathbf{u}_k\mathbf{u}_k^\intercal\hat{\mathbf{J}}_{k-1}^\intercal} + \frac{\mathbf{y}_k\mathbf{y}_k^\intercal}{\mathbf{u}_k\mathbf{y}_k^\intercal} \tag{13}$$

It is recognized with the best numerical stability.

When a new data pair $(\mathbf{y}_k, \mathbf{u}_k)$ enters the system, we update the Jacobian $\hat{\mathbf{J}}_k$ with the above estimators.

Remark 2. This article assumes that the robot manipulates soft objects at low speed, so the deformation of soft object is relatively slow. Thus, the formula (6) is smooth, and the deformation Jacobian matrix $\mathbf{J}_k$ can be estimated online.

D. Shape Servoing Controller

Let us assume that at the time instant k, the Jacobian matrix $\hat{\mathbf{J}}_k$ has been exactly estimated by the proposed online estimator, so that the shape-motion difference model satisfies:

$$\mathbf{s}_k = \mathbf{s}_{k-1} + \hat{\mathbf{J}}_k \cdot \Delta\mathbf{r}_k \tag{14}$$

In this section, different from the previous velocity controller [12], we utilize a model predictive controller [31] to minimize the shape error $\mathbf{e}_k = \mathbf{s}^* - \mathbf{s}_k$ between the measured feature $\mathbf{s}_k$ and a constant target feature $\mathbf{s}^*$. The following sliding surface can be used to plan the desired deformation of the soft object at time $k+w$.

$$\mathbf{s}_{k+w}^d = \mathbf{s}^* - e^{-\rho w} \cdot \mathbf{e}_k \tag{15}$$

where $w \in [0, H]$ represents the length of prediction horizon, e is nature exponential, ρ is a positive constant. Velocity command $\mathbf{u}_k$ is assumed to remain constant from k to $k+w$. Error $\boldsymbol{\varepsilon}$ between the planned and the approximated deformation at time $k+w$ is defined as follows:

$$\boldsymbol{\varepsilon}_{k+w} = \left(1 - e^{-\rho w}\right)\mathbf{e}_k - w\hat{\mathbf{J}}_k\mathbf{u}_k \tag{16}$$

Velocity command $\mathbf{u}_k$ is then calculated by minimizing error $\boldsymbol{\varepsilon}$ over the period from k to $k+H$, expressed as follows:

$$\min \frac{1}{2}\left(\sum_{w=0}^{H}\alpha^w\left\|\left(1 - e^{-\rho w}\right)\mathbf{e}_k - w\hat{\mathbf{J}}_k\mathbf{u}_k\right\|^2 + \mathbf{u}_k^\intercal\mathbf{Q}\mathbf{u}_k\right) \tag{17}$$

where $0 < \alpha \leq 1$, and $\mathbf{Q}$ is a symmetric and positive definite matrix. When the command $\mathbf{u}_k$ is too large, which will cause the robot to move too fast. Therefore, the manipulated object will oscillate, which will affect the estimation accuracy of the Jacobian matrix. Therefore, $\mathbf{Q}$ is used to adjust the velocity command $\mathbf{u}_k$. Differentiating (17) with respect to $\mathbf{u}_k$, the gradient ∇ is calculated as follows:

$$\nabla = \sum_{w=0}^{H} -w\alpha^w\hat{\mathbf{J}}_k^\intercal\left(\left(1-\beta^w\right)\mathbf{e}_k - w\hat{\mathbf{J}}_k\mathbf{u}_k\right) + \mathbf{Q}\mathbf{u}_k \tag{18}$$

where $\beta = e^{-\rho}$, and by setting $\nabla = 0$, we derive the following velocity command $\mathbf{u}_k$.

$$\begin{aligned}
\mathbf{u}_k &= \left(a\hat{\mathbf{J}} + \hat{\mathbf{J}}^{\intercal+}\mathbf{Q}\right)^+(b-c)\,\mathbf{e}_k \\
a &= \left(H^2\alpha^H - 2b\right)/\ln\alpha \\
b &= \left(H\alpha^H\ln\alpha - \alpha^H + 1\right)/\ln^2\alpha \\
c &= \left(H(\alpha\beta)^H\ln(\alpha\beta) - (\alpha\beta)^H + 1\right)/\ln^2(\alpha\beta)
\end{aligned} \tag{19}$$

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

$$\mathbf{r}_k = \mathbf{r}_{k-1} + \mathbf{u}_k \tag{20}$$

Then, according (14), we have

$$\mathbf{e}_k - \mathbf{e}_{k-1} = -\hat{\mathbf{J}}_k\Delta\mathbf{r}_k \tag{21}$$

Note, we assume $\hat{\mathbf{J}}_k$ has a full column rank, and substituting (19) into (21) yields the error dynamics as follows:

$$\left(a\mathbf{E}_n + \hat{\mathbf{J}}_k^{\intercal+}\mathbf{Q}\hat{\mathbf{J}}_k^+\right)(\mathbf{e}_k - \mathbf{e}_{k-1}) + (b-c)\,\mathbf{e}_k = 0 \tag{22}$$

As $a > 0, b - c > 0$, and $\hat{\mathbf{J}}_k^{\intercal+}\mathbf{Q}\hat{\mathbf{J}}_k^+$ is a positive definite matrix, thus, the error $\mathbf{e}_k$ asymptotically converges to error, namely, $\lim_{t\to\infty}\mathbf{s}_k = \mathbf{s}_k^*$. However, when the reachability of the desired goal $\mathbf{s}_k^*$ is not satisfied, the $\hat{\mathbf{J}}_k$ may not be a column full-rank matrix. The feedback error $\|\mathbf{e}_k\|$ may only converge to the neighborhood near the origin. For such under-actuated visual servo control tasks, the global asymptotic convergence is challenging to guarantee [32].

Remark 3. The velocity controller (19) and Jacobian estimators (10)(11)(12)(13) designed in this paper requires only visual feedback data, no additional sensors to measure the system state, no prior knowledge of the system model, and no need to calibrate the camera.

IV. SIMULATION RESULTS

We consider a planar robot (2DOF) that rigidly grasps one end of an elastic rod, whose other end is static. And, for brevity, in the figures, we do not display the robot. The cable simulator is simulated as in [33] by using the minimum energy principle [34], and publicly available at https://github.com/q546163199/shape_deformation. All numerical simulations are implemented in Python.

A. Feature Extraction Comparison

In this section, we randomly control the robot's movement to get 40,000 samples ($N = 100$) utilized to train and verify the performance of DAE compared with PCA [26]. Refer to Section III-A, we use MLP as the basic building blocks of the Encoder and Decoder, both are $[200-1200-160-128-$

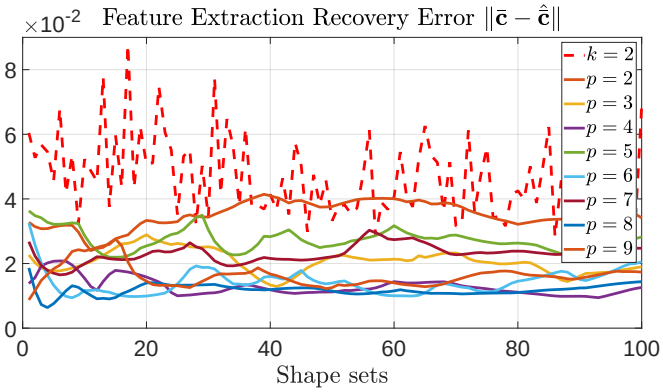


Fig. 6: Feature extraction comparison between DAE and PCA among 100 shape sets in the simulation.

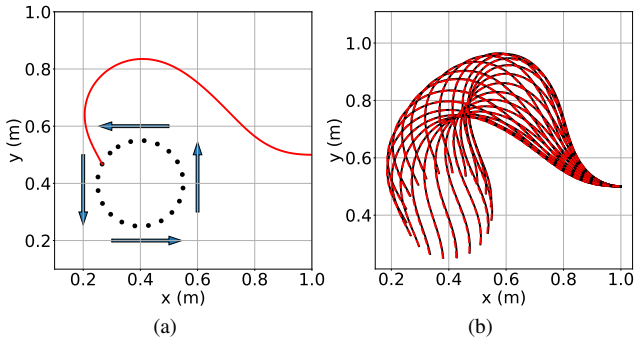


Fig. 7: Jacobian test framework. (a) Motion trajectory of robot's end-effector. (b) Comparison between the simulated cable profile (black solid line) and its reconstruction shape obtained by DAE (red dashed line) with $p = 4$.

$64 - p]$ and $[p - 64 - 128 - 160 - 1200 - 200]$, respectively. We implement DAE on PyTorch and train by using ADAM optimizer with an initial learning rate of 0.001 and a batch size of 500 and employ RELU activation functions in the Encoder and Decoder. Note, k controls the dimension of feature $\mathbf{s}$ in PCA [26], while p controls that in DAE.

In Fig. 6, $\left\|\bar{\mathbf{c}} - \hat{\bar{\mathbf{c}}}\right\|$ represents the error between the simulated shape $\bar{\mathbf{c}}$ and the reconstruction shape $\hat{\bar{\mathbf{c}}}$ obtained from DAE or PCA, and it shows that the shape reconstruction accuracy of the DAE is better than PCA.

Table I shows in detail the accuracy of shape reconstruction under various cases of p. It shows that the accuracy of $p = 4$ is the best, $p = 6$ is the second good, and $p = 2$ is the worst. This means that when the feature dimension p is too low to represent elastic rods' potential geometric information fully. Considering the trade balance of system controllability and shape representation capability, we use DAE as the feature extractor with $p = 4$ in the following sections.

TABLE I: Comparison results between DAE and PCA among 100 shape sets in the simulation

	p	k	$\left\|\bar{\mathbf{c}} - \hat{\bar{\mathbf{c}}}\right\|$ (DAE)	$\left\|\bar{\mathbf{c}} - \hat{\bar{\mathbf{c}}}\right\|$ (PCA)
Case1	2	2	0.0352	0.0461
Case2	3	2	0.0206	0.0461
Case3	4	2	0.0129	0.0461
Case4	5	2	0.0273	0.0461
Case5	6	2	0.0142	0.0461
Case6	7	2	0.0231	0.0461
Case7	8	2	0.0178	0.0461
Case8	9	2	0.0162	0.0461

B. Validation of the Jacobian Estimation

This section evaluates the proposed Jacobian estimators' performance, namely, R1, SR1, DFP, and BFGS. The planar robot grasps one end of the simulated rod and makes a counterclockwise circular motion with center (0.4, 0.4), shown in Fig. 7a. And, different from the simple initialization of $\hat{\mathbf{J}}_0$, such as the identity matrix, we move the robot in the initial sampling area (ensure that the motions cannot be collinear and close to the start point) to initialize $\hat{\mathbf{J}}_0$. Using this method can improve the accuracy of initialization, and reduce the possibility of singular problems with the Jacobian matrix, thus enhance the safety of operation. And, we give two error criteria (23) to compare each Jacobian estimator qualitatively.

$$T_1 = \|\mathbf{s}_k - \hat{\mathbf{s}}_k\| \quad T_2 = \left\|\Delta\mathbf{s}_k - \hat{\mathbf{J}}_k \Delta\mathbf{r}_k\right\| \tag{23}$$

where $\mathbf{s}_k$ is feedback shape feature measured by DAE, $\hat{\mathbf{s}}_k$ is estimated by (10)(11)(12)(13), and calculated by (24).

$$\hat{\mathbf{s}}_k = \hat{\mathbf{s}}_{k-1} + \hat{\mathbf{J}}_k \Delta\mathbf{r}_k \tag{24}$$

Note, while validating the Jacobian estimators, we also verify the shape reconstruction accuracy of DAE and depicted in Fig. 7b. It shows that the shape reconstruction accuracy of the DAE is good, which further verifies the effectiveness of DAE in the shape representation issue.

Fig. 8 demonstrates the plots of T_1 and T_2 during the circle motion. For T_1 curve, we know that all four estimators can accurately update the Jacobian matrix $\hat{\mathbf{J}}_k$ and the average error of BFGS is the smallest. For T_2, the BFGS has no apparent fluctuations, and the estimation accuracy is the best. DFP is second good, while R1 and SR1 have similar results, consistent with the theoretical analysis. The above two analyses prove that the BFGS algorithm has excellent Jacobian matrix update ability. When the cable shape changes (different from the initial shape), BFGS can still calibrate and update the Jacobian matrix in time to identify the pseudo-physical parameters of the object (more specifically, BFGS can estimate the change direction of shape feature $\mathbf{s}$ in the latent space).

C. Manipulation of Elastic Rods

In this section, we use the velocity command (19) to control the robot to deform the elastic rod into the predetermined desired constant configuration $\bar{\mathbf{c}}^*$, corresponding to $\mathbf{s}^*$. Also, we give the error criterion (25) to assess the deformation performance.

$$T_3 = \|\bar{\mathbf{c}}_k - \bar{\mathbf{c}}_k^*\| \tag{25}$$

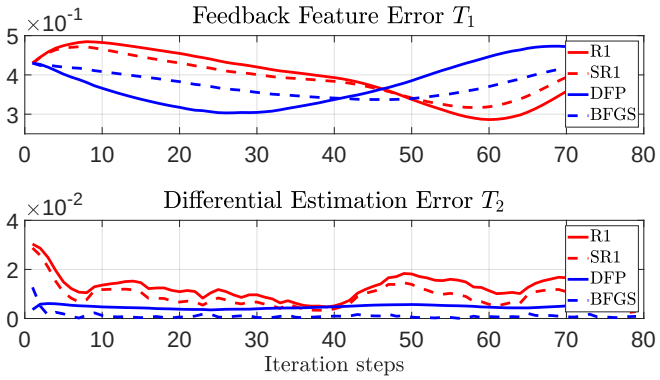


Fig. 8: Profiles of the criteria T_1 and T_2 that are computed along the circular trajectory around the center (0.4, 0.4).

TABLE II: Results among R1, SR1, DFP and BFGS

	R1	SR1	DFP	BFGS
Steps	58	48	32	22
Time (second)	33.64	27.84	18.56	12.76

Fig. 9 depicts the progress of the cable deformation under the velocity command (19) based on R1, SR1, DFP and BFGS, respectively, while Fig. 10 depicts the curve of T_3 and the velocity command $\Delta\mathbf{r}_k$, and Table II gives the detailed time comparison. Both figures clearly shows that BFGS is the best method with the shortest convergence time and smallest deformation error, followed by DFP, and the effect of both R1 and SR1 is similar.

V. EXPERIMENTAL RESULTS

We conduct various experiments with two UR5 (6DOF) which support velocity control mode (see Fig. 11). $\Delta\mathbf{r}_1 = [\Delta r_{11}, \Delta r_{12}, \Delta r_{13}]^\intercal \in \mathbb{R}^3$, $\Delta\mathbf{r}_2 = [\Delta r_{21}, \Delta r_{22}, \Delta r_{23}]^\intercal \in \mathbb{R}^3$ represents the velocity command $\Delta\mathbf{r} = [\Delta\mathbf{r}_1^\intercal, \Delta\mathbf{r}_2^\intercal]^\intercal \in \mathbb{R}^6$ of left and right UR5, respectively.

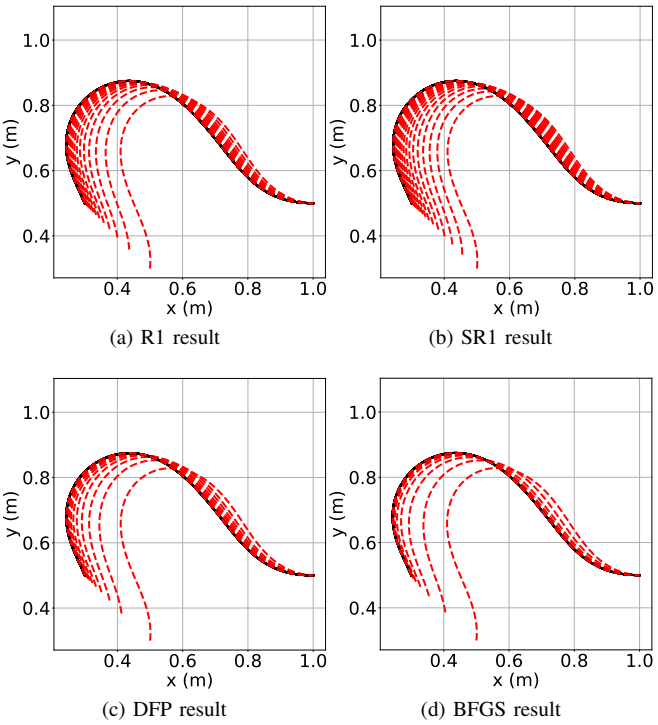
- Δr_{i1} and Δr_{i2}, $i = 1, 2$ represents the linear velocity of end-effector along x-axis and y-axis of each UR5 in the world frame.
- Δr_{i3}, $i = 1, 2$ represents the angular velocity of the sixth joint of each UR5 along z-axis in the world frame.

We use a Logitech C270 camera to capture the rod's image, and use OpenCV to process on the Linux-based PC at 30 fps. Meanwhile, we display the deformation trajectories once every two frames to clearly compare the convergence effects of each algorithm.

A. Image Processing

This section contains two parts, one is to verify the proposed SOM-based centerline extraction algorithm, and the other is to give the image processing steps. First, we compare three centerline extraction algorithms:

- Method proposed in our previous paper [12], including sorting algorithm and down-sampling algorithm.
- Based on CL, which is the traditional clustering method.
- The proposed SOM-based centerline extraction algorithm.



(a) R1 result (b) SR1 result

(c) DFP result (d) BFGS result

Fig. 9: Profiles of of the shape deformation simulation among R1, SR1, DFP and BFGS (red dashed curves represent the initial and transitional trajectories, and the black solid curve represents the target shape $\bar{\mathbf{c}}^*$ with shape feature $\mathbf{s}^*$).

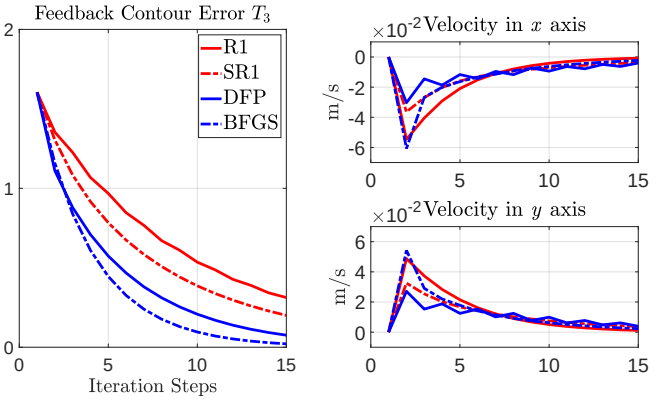


Fig. 10: Profiles of the criterion T_3 and velocity command $\Delta\mathbf{r}_k$ among R1, SR1, DFP and BFGS within manipulation.

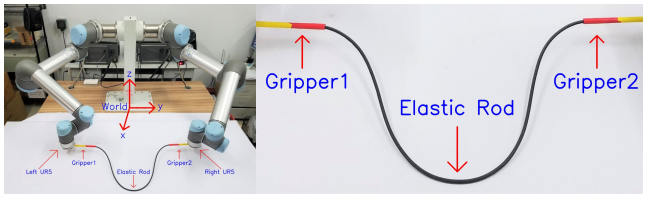
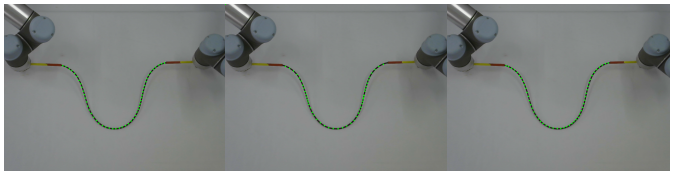


Fig. 11: Experimental setup.



(a) Previous in [12] (b) CL [27] (c) Proposed SOM

Fig. 12: Comparison of three centerline extraction algorithms, including previous algorithm [12], CL-based algorithm [27] and the proposed SOM-based algorithm.

For a fair comparison, all three algorithms need to provide an ordered, fixed-length $N = 50$, equidistant centerline. As both the CL-based and SOM-based algorithms only generate a unordered fixed-length centerline, thus we need to use the sorting algorithm [12] to sort unordered centerline. For the SOM implementation, an open-source toolbox provided by [35] was utilized.

TABLE III: Comparison results among three centerline extraction algorithms with $N = 50$

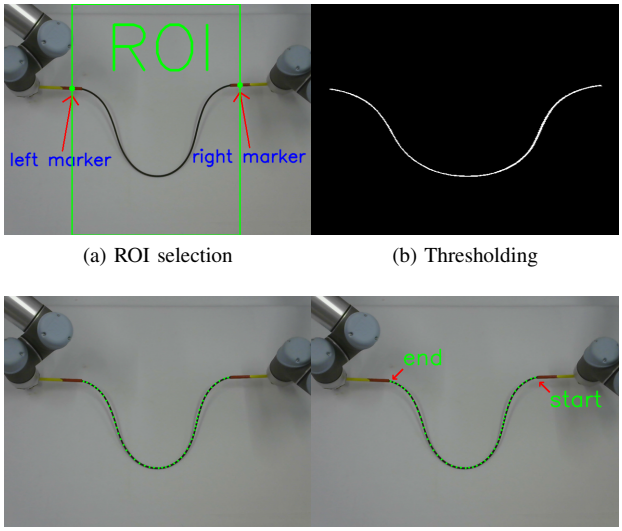
	Previous [12]	CL [27]	SOM
Time (Second)	1.68	0.98	0.38

As shown in Table III, the SOM-based algorithm is the fastest and able to efficiently perform centerline extraction. One reason is that the provided SOM toolbox is already highly optimized. Another reason is that the centerline produced by CL-based and SOM-based clustering algorithms has the nature of fixed-length and equidistant-sampling. This saves much time than the previous method [12], because the previous way is to sort all the points first and then perform equidistant sampling processing.

Fig. 12 shows that the SOM-based algorithm has the same accuracy as the previous algorithm [12], and CL-based has the worst sampling accuracy. This verifies the effectiveness of the proposed SOM-based algorithm. Under the premise of ensuring the same sampling accuracy, considering that SOM's processing speed is the fastest, thus in the following sections, we use the SOM-based algorithm to implement the centerline extraction in a real-time working environment.

Second, we present the relevant image processing for centerline extraction. The overall process (shown in Fig. 13) is as follows:

1) Segment the red areas nearby the Gripper1 and Gripper2 based on HSV color space and mark them as two green marker points, and segment the region of the interest (ROI) containing the rod according to both green marker points (see Fig. 13a).
2) Identify the rod in ROI, remove the noise, and obtain a binary image of the rod using OpenCV morphological opening algorithm (see Fig. 13b).
3) Use the proposed SOM-based algorithm to get a unordered centerline with $N = 50$ (see Fig. 13c).



(a) ROI selection (b) Thresholding

(c) Centerline extraction (SOM) (d) Coordinate sorting

Fig. 13: Image processing steps.

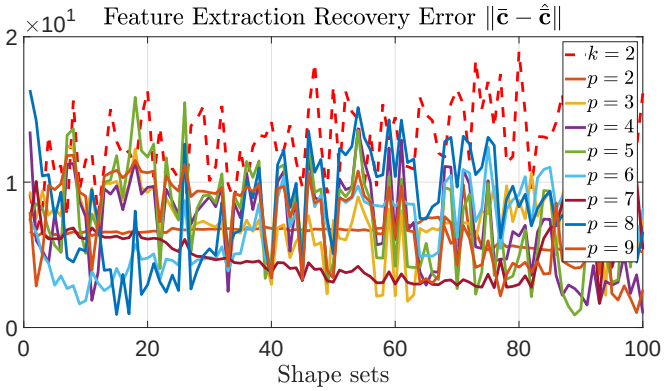


Fig. 14: Feature extraction comparison between DAE and PCA among 100 shape sets in the experiment.

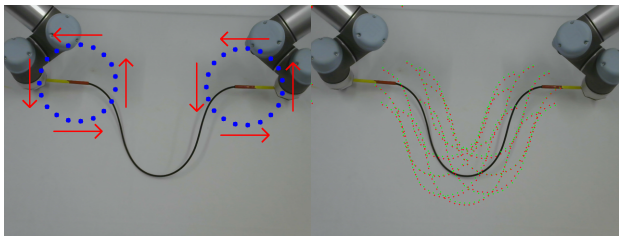
4) Apply the sorting algorithm [12] to get ordered centerline (see Fig. 13d). Note that start point is the closest point to the right marker point on the centerline.

B. Feature Extraction Comparison

Similar to Section IV-A, we still get 40,000 samples by controlling the dual-UR5 random movement to train the DAE. The network of DAE is similar with Section IV-A, $[100 - 1200 - 160 - 128 - 64 - p]$ for Encoder and $[p - 64 - 128 - 160 - 1200 - 100]$ for Decoder, respectively. We add Batch-Normalization (BN) after each layer (we find that BN is beneficial in the experiment). Fig. 14 shows these comparison results, we can see that the accuracy of DAE is still better than PCA. From Table IV, we know that the accuracy of $p = 4$ is the best, while $p = 8$ is the second. This is consistent with the simulation results. It further verifies the effectiveness of the feature extraction based DAE. In the following sections, we use DAE as the feature extractor with $p = 4$.

TABLE IV: Comparison results between DAE and PCA among 100 shape sets in the experiment

	p	k	$\|\bar{\mathbf{c}} - \hat{\bar{\mathbf{c}}}\|$ (DAE)	$\|\bar{\mathbf{c}} - \hat{\bar{\mathbf{c}}}\|$ (PCA)
Case1	2	2	6.6016	13.7617
Case2	3	2	6.6897	13.7617
Case3	4	2	5.8126	13.7617
Case4	5	2	6.8313	13.7617
Case5	6	2	6.3862	13.7617
Case6	7	2	6.5260	13.7617
Case7	8	2	5.8293	13.7617
Case8	9	2	6.1313	13.7617



(a) (b)

Fig. 15: Jacobian test framework. (a) Motion trajectory of robot's end-effector. (b) Comparison between the visually measured cable profile (green dot line) and its reconstruction shape obtained by DAE (red dot line) with $p = 4$.

C. Validation of the Jacobian Estimation

Similar to Section IV-B, we still control the dual-UR5 to move along a fixed circular trajectory, depicted in Fig. 15a.

Fig. 15b shows that the shape reconstruction ability of the DAE is still excellent in the experiment. Thus, it can be considered that whether it is for simulation or experiment, it is possible to apply DAE in shape representation in the shape deformation tasks.

Fig. 16 shows that BFGS still has a significant estimation effect, there is no obvious fluctuation, and the estimation error is minimal, which further illustrates the effectiveness of BFGS and its strong adaptive ability in different regions.

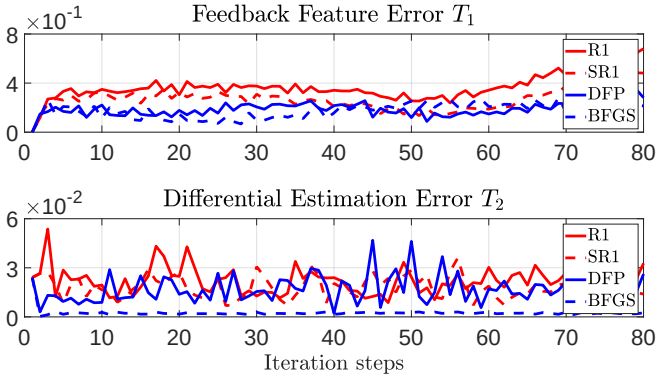


Fig. 16: Profiles of the criteria T_1 and T_2 that are computed along the circular trajectory.

D. Manipulation of Elastic Rods

Similar to Section IV-C, we also need to give a desired shape $\bar{\mathbf{c}}^*$ in advance. To get the feasible target shape, we move the robot to a pre-determined position and use the camera to record the shape at this time and record it as the desired shape $\bar{\mathbf{c}}^*$, and then command the robot back to the initial position and start the deformation. Considering safety, the saturation of the velocity command $\Delta\mathbf{r}$ is set to $(0.01m/s, 0.01m/s, 0.1rad/s, 0.01m/s, 0.01m/s, 0.1rad/s)$, respectively. We implement four sets of experiments with different initial and target shapes to verify the proposed algorithm's applicability.

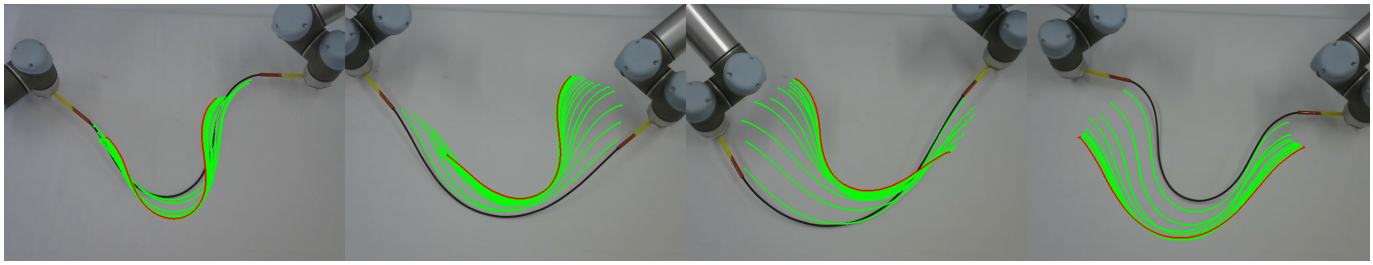
Fig. 17 shows that, under the proposed algorithm, the dual-UR5 can deform the elastic rod to the desired shape accurately, and there is no damage to the object during the deformation process.

Fig. 18 shows the profiles of T_3 and the velocity command $\Delta\mathbf{r}_k$. Corresponding to the simulation results, we know that BFGS still has the most significant control effect, the fastest convergence speed, no apparent fluctuations (large instantaneous deformation), which means that BFGS has excellent adaptability and robustness to the various conditions in the shape deformation issue. Meanwhile, the experiment taken in this paper has more control degree of freedoms (dual-UR5) than the previous experiment [12], which proves the effectiveness of the proposed algorithm for multi-degree-of-freedom control ($q > 2$).

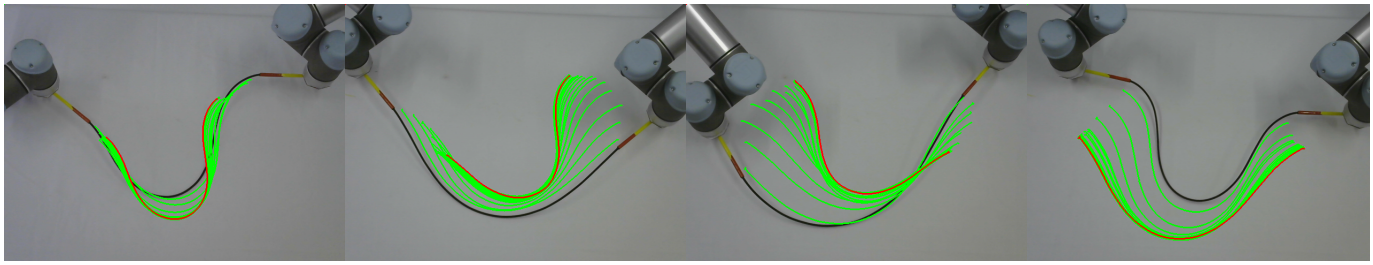
VI. Conclusions

This paper has proposed a framework for the deformation control of soft objects without any prior physical knowledge, and includes shape feature extraction, Jacobian matrix estimation, and a robust SOM-based centerline extraction algorithm. First, new shape features based on the DAE are utilized to represent the object's centerline in the low-dimensional feature space. Second, we evaluate the performance of four Jacobian estimators (namely R1, SR1, DFP, and BFGS) in estimating the nonlinear time-varying Jacobian matrix. Third, the velocity controller is derived and the system stability is proved. Finally, numerical and experimental results validate the effectiveness and feasibility of the proposed algorithm.

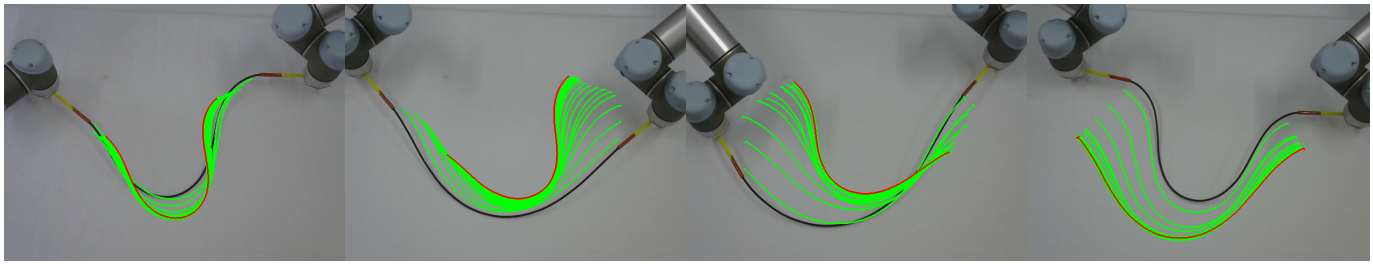
This paper uses DAE to flexibly reduce the high-dimensional geometric information of soft objects to a low-dimensional feature space. Compared with the traditional PCA, the proposed feature extraction algorithm has better shape representation capabilities and does not require any artificial markers, making it more widely used in practical applications. To avoid identifying the physical parameters and camera models, we use the Broyden algorithms to obtain a numerical approximation of the deformation Jacobian matrix in real-time. From the results, BFGS has the advantages of simple structure, fast calculation speed, accurate approximation performance. Simultaneously, we use design a robust SOM-based centerline extraction algorithm with faster calculation speed and higher extraction accuracy than the previous methods [12]. The overall visual-servoing



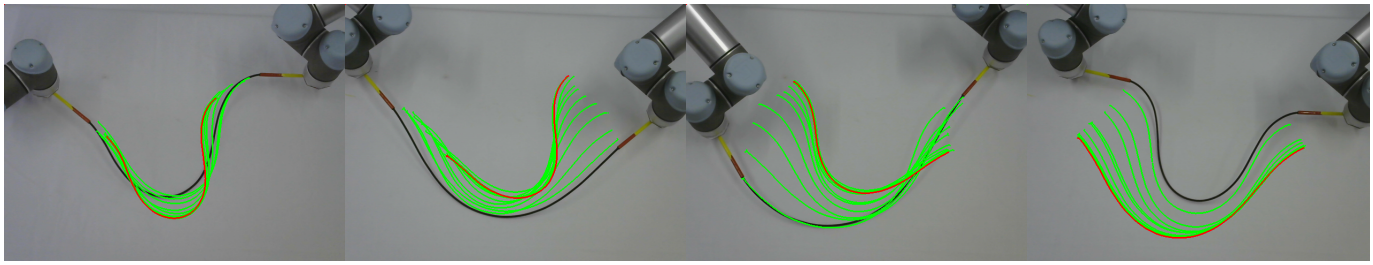
(a) Experiment1-R1 (b) Experiment2-R1 (c) Experiment3-R1 (d) Experiment4-R1



(e) Experiment1-SR1 (f) Experiment2-SR1 (g) Experiment3-SR1 (h) Experiment4-SR1



(i) Experiment1-DFP (j) Experiment2-DFP (k) Experiment3-DFP (l) Experiment4-DFP



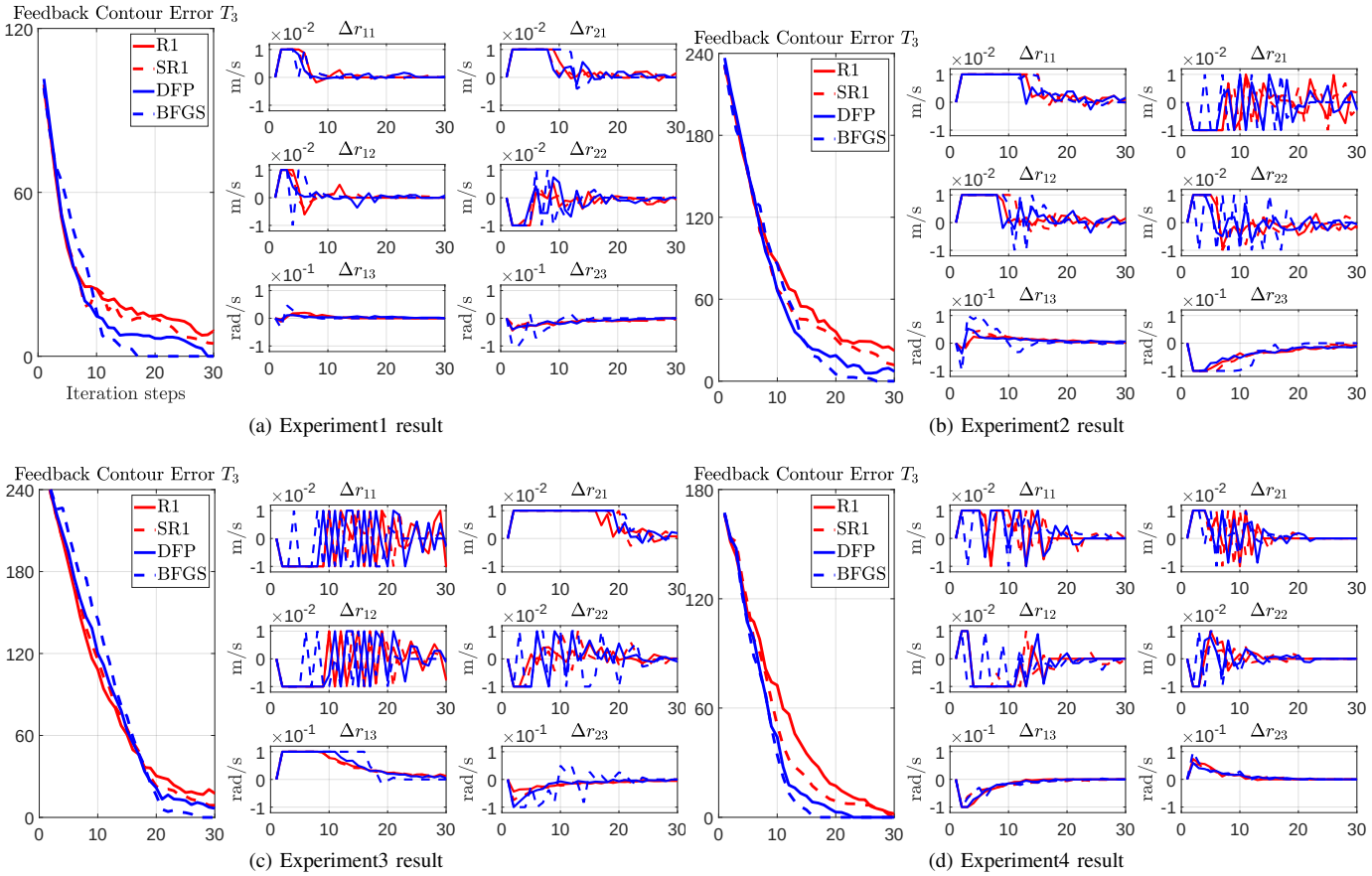
(m) Experiment1-BFGS (n) Experiment2-BFGS (o) Experiment3-BFGS (p) Experiment4-BFGS

Fig. 17: Initial (black solid line), transition (green solid line) and target (red solid line) configurations in the four shape deformation experiments which have a variety of different initial and target shape with dual-UR5 robot among R1, SR1, DFP and BFGS.

deformation control system is completely calculated from the visual feedback data, without any prior physical model of the object and camera calibration.

The proposed method also has some limitations. First, the manipulated object is only soft elastic objects (i.e., carbon fiber rod). Thus the proposed algorithm is not suitable for inelastic items (i.e., plasticine and rope). Second, although the DAE has good shape representation ability, it needs an extensive and rich-enough data set to train itself, which has particular difficulties in practical applications. Third, the approximation of the Jacobian matrix based on the Broyden algorithms is easy to fall into the local optimum, which may lead to generate the destructive operation of the object such as over-tension and over-compression in the manipulation process.

In the future, we will consider carrying out 3D deformation tasks to manipulate more complex shapes (i.e., M-shaped, spiral). Besides, improving the existing DAE to make it suitable for different scenarios and materials, we also consider path planning to avoid possible destructive operations during the manipulation process, and improve the system's operating safety.



(a) Experiment1 result (b) Experiment2 result

(c) Experiment3 result (d) Experiment4 result

Fig. 18: Profiles of the criterion T_3 and velocity command $\Delta\mathbf{r}_k$ among R1, SR1, DFP and BFGS within four shape deformation experiments.

REFERENCES

[1] 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.

[2] P. Abolmaesumi, S. E. Salcudean, W.-H. Zhu, M. R. Sirouspour, and S. P. DiMaio, "Image-guided control of a robot for medical ultrasound," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 11–23, 2002.

[3] T. Tang, C. Wang, and M. Tomizuka, "A framework for manipulating deformable linear objects by coherent point drift," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3426–3433, 2018.

[4] P. Sun, Z. Hu, and J. Pan, "A general robotic framework for automated cloth assembly," in *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE, 2019, pp. 47–52.

[5] 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.

[6] A.-M. Cretu, P. Payeur, and E. M. Petriu, "Soft object deformation monitoring and learning for model-based robotic hand manipulation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 740–753, 2011.

[7] D. Navarro-Alarcon, H. M. Yip, Z. Wang, Y.-H. Liu, F. Zhong, *et al.*, "Automatic 3-d manipulation of soft objects by robotic arms with an adaptive deformation model," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 429–441, 2016.

[8] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Persistent point feature histograms for 3d point clouds," in *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany*, 2008, pp. 119–128.

[9] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217.

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

[11] 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, 2017.

[12] 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," *arXiv preprint arXiv:2008.06896*, 2020.

[13] N. Yeo, K. Lee, Y. Venkatesh, and S. H. Ong, "Colour image segmentation using the self-organizing map and adaptive resonance theory," *Image and Vision Computing*, vol. 23, no. 12, pp. 1060–1079, 2005.

[14] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2372–2379, 2020.

[15] A. J. Valencia, F. Nadon, and P. Payeur, "Toward real-time 3d shape tracking of deformable objects for robotic manipulation and shape control," in *2019 IEEE SENSORS*. IEEE, 2019, pp. 1–4.

[16] M. Polic, I. Krajacic, N. Lepora, and M. Orsag, "Convolutional autoencoder for feature extraction in tactile sensing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3671–3678, 2019.

[17] T. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[18] Z. Guo and R. W. Hall, "Parallel thinning with two-subiteration algorithms," *Communications of the ACM*, vol. 32, no. 3, pp. 359–373, 1989.

[19] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[20] D. Henrich and H. Wörn, "Robot manipulation of deformable objects," 2000.

[21] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," *arXiv preprint arXiv:2003.05436*, 2020.

[22] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, *et al.*, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2146–2153.

[23] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *arXiv preprint arXiv:1812.00568*, 2018.

[24] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of intelligent and robotic systems*, vol. 3, no. 3, pp. 201–212, 1990.

[25] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[26] J. Zhu, D. Navarro-Alarcon, R. Passama, and A. Cherubini, "Vision-based manipulation of deformable and rigid objects using subspace projections of 2d contours," 2020.

[27] D.-C. Park, "Centroid neural network for unsupervised competitive learning," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 520–528, 2000.

[28] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Mathematics of computation*, vol. 19, no. 92, pp. 577–593, 1965.

[29] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[30] J. E. Dennis and J. J. Moré, "A characterization of superlinear convergence and its application to quasi-newton methods," *Mathematics of computation*, vol. 28, no. 126, pp. 549–560, 1974.

[31] B. Ouyang, H. Mo, H. Chen, Y. Liu, and D. Sun, "Robust model-predictive deformation control of a soft object by using a flexible continuum robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 613–618.

[32] S. Hutchinson and F. Chaumette, "Visual servo control, part i: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[33] H. Wakamatsu, S. Hirai, and K. Iwata, "Modeling of linear objects considering bend, twist, and extensional deformations," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 1995, pp. 433–438.

[34] P. Hamill, *A student's guide to Lagrangians and Hamiltonians*. Cambridge University Press, 2014.

[35] G. Vettigli, "Minisom: minimalistic and numpy-based implementation of the self organizing map," gitHub.[Online]. Available: https://github.com/JustGlowing/minisom/.