Model Predictive Manipulation of Compliant Objects with Multi-Objective Optimizer and Adversarial Network for Occlusion Compensation

ARTICLE INFO

Keywords:
Robotics
Visual servoing
Deformable objects
Occlusion compensation
Model predictive control

ABSTRACT

The robotic manipulation of compliant objects is currently one of the most active problems in robotics due to its potential to automate many important applications. Despite the progress achieved by the robotics community in recent years, the 3D shaping of these types of materials remains an open research problem. In this paper, we propose a new vision-based controller to automatically regulate the shape of compliant objects with robotic arms. Our method uses an efficient online surface/curve fitting algorithm that quantifies the object's geometry with a compact vector of features; This feedback-like vector enables to establish an explicit shape servo-loop. To coordinate the motion of the robot with the computed shape features, we propose a receding-time estimator that approximates the system's sensorimotor model while satisfying various performance criteria. A deep adversarial network is developed to robustly compensate for visual occlusions in the camera's field of view, which enables to guide the shaping task even with partial observations of the object. Model predictive control is utilized to compute the robot's shaping motions subject to workspace, saturation, and obstacle constraints. A detailed experimental study is presented to validate the effectiveness of the proposed control framework.

1. Introduction

The manipulation/shaping of deformable bodies by robots is a fundamental problem that has recently attracted the attention of many researchers Zhu, Cherubini, Dune, Navarro-Alarcon, Alambeigi, Berenson, Ficuciello, Harada, Li, Pan et al. (2021a); Its complexity has forced researchers to develop new methods in a wide range of fundamental areas that include representation, learning, planning, and control. From an applied research perspective, this challenging problem has shown great potential in various economically-important tasks such as the assembly of compliant/delicate objects Li, Su and Liu (2018), surgical/medical robotics Cao, Li, Phan, Tiong, Kaan, Liu, Lai, Huang, Le, Miyasaka et al. (2020), cloth/fabric folding Hu, Sun and Pan (2018), etc. The manipulation of compliant materials contrast with its rigid-body counterpart in that physical interactions will invariably change the object's shape, which introduces additional degrees-of-freedom to the typically unknown objects, and hence, complicates the manipulation task. While great progress has been achieved in recent years, the development of these types of embodied manipulation capabilities is still largely considered an open research problem in robotics and control.

There are various technical issues that hamper the implementation of these tasks in real-world unstructured environments, which largely differ from implementations in ideal simulation environments (a trend followed by many recent works). Here, we argue that to effectively servo-control the shape of deformable materials in the field, a sensor-based controller must possess the following features: 1) Efficient compression of the object's high-dimensional shape; 2) Occlusion-tolerant estimation of the objects geometry; 3) Adaptive prediction of the differential shape-motion model; Our aim in this paper is precisely to develop a new shape controller endowed with all the above-mentioned features. The proposed method is formulated under the model predictive control (MPC) framework that enables to compute shaping actions that satisfy multiple performance criteria (a key property for real engineering applications).

1.1. Related Work

Many researchers have previously studied this challenging problem (we refer the reader to Sanchez, Corrales, Bouzgarrou and Mezouar (2018); Yin, Varava and Kragic (2021) for comprehensive reviews). To servo-control the object's non-rigid shape, it is essential to design a low-dimensional feature representation that can capture the key geometric properties of the object. Several representation methods have been proposed before, e.g., geometric features based on points, angles, curvatures, etc Navarro-Alarcon, Liu, Romero and Li (2013, 2014); However, due to their

ORCID(s):

hard-coded nature, these methods can only be used to represent a single shaping action. Other geometric features computed from contours and centerlines Wang, Yang, Wang, Liang, Chen and Liu (2018); Navarro-Alarcon and Liu (2017); Qi, Ma, Zhu, Zhou, Lyu, Zhang and Navarro-Alarcon (2021) can represent soft object deformation in a more general way. Various data-driven approaches have also been proposed to represent shapes, e.g., using fast point feature histograms Hu, Han, Sun, Pan and Manocha (2019), bottleneck layers Qi, Ma, Zhou, Zhang, Lyu and Navarro-Alarcon (2022); Zhou, Zhu, Huo and Navarro-Alarcon (2021), principal component analysis Zhu, Navarro-Alarcon, Passama and Cherubini (2021b), etc. However, there is no widely accepted approach to compute efficient/compact feature representations for 3D shape; This is still an open research problem.

Occlusions of a camera's field of view pose many complications to the implementation of visual servoing controllers, as the computation of (standard) feedback features requires complete visual observations of the object at all times Lv, Yu, Pu, Jiang, Huang and Li (2023). Many methods have been developed to tackle this critical issue, e.g. Cazy, Wieber, Giordano and Chaumette (2015) used the estimated interaction matrix to handle information loss in visual servoing, yet, this method requires a calibrated visual-motor model. Coherent point drift was utilized in Tang, Wang and Tomizuka (2018) to register the topological structure from previous sequences and to predict occlusions; However, this method is sensitive to the initial point set, further affects the registration result. A structure preserved registration method was presented in Tang and Tomizuka (2018) to track occluded objects; This approach has good accuracy and robustness to noise, yet, its efficiency decreases with the number of points. To efficiently implement vision-based strategies in the field, it is essential to develop algorithms that can robustly guide the servoing task, even in the presence of occlusions.

To visually guide the manipulation task, control methods must have some form of model that (at least approximately) describes how the input robot motions produce output shape changes Navarro-Alarcon, Qi, Zhu and Cherubini (2020); In the visual servoing community, such differential relation is typically captured by the so-called interaction (Jacobian) matrix Chaumette and Hutchinson (2006). Many methods have been proposed to address this issue, e.g. the Broyden update rule Hosoda and Asada (1994) is a classical algorithm to iteratively estimate this transformation matrix Navarro-Alarcon et al. (2013); Alambeigi, Wang, Hegeman, Liu and Armand (2018); Lagneau, Krupa and Marchal (2020b). Although these types of algorithms do not require knowledge of the model's structure, its estimation properties are only valid locally. Other approaches with global estimation properties include algorithms based on (deep) artificial neural networks Li et al. (2018), and optimization-based algorithms Lagneau, Romain, Alexandre and Maud (2020a), etc. However, the majority of existing methods estimate this model based on a single performance criterion (typically, a first-order Jacobian-like relation), which limits the types of dynamic responses that the robot can achieve during a task. A multi-objective model estimation is particularly important in the manipulation of compliant materials, as their mechanical properties are rarely known in practice, thus, making it hard to meet various performance requirements.

To compute the active shaping motions, most control methods only formulate the problem in terms of the final desired shape and do not typically consider the system's physical constraints. MPC represents a feasible solution to these issues, as it performs the control tasks by optimizing cost functions over a finite time-horizon rather than finding the exact analytical solution Hajiloo, Keshmiri, Xie and Wang (2015); This allows MPC to compute controls that guide the task while satisfying a set of constraints, e.g., control saturation, workspace bounds, and obstacle constraint etc. Despite its valuable and flexible properties, MPC has not been sufficiently studied in the context of shape control of deformable objects.

1.2. Our Contribution

To solve the above-mentioned issues, in this paper we propose a new control framework to manipulate purely-elastic objects into desired configurations. The original features of our new methodology include: 1) A parametric shape descriptor to efficiently characterize 3D deformations based on online curve/surface fitting; 2) A robust shape prediction network based on adversarial neural networks to compensate visual occlusions; 3) An optimization-based estimator to approximate the deformation Jacobian matrix and satisfy various performance constraints; 4) An MPC-based controller to guide the shaping motions while simultaneously solving saturation, workspace, and obstacle constraints.

Although our preliminary approach Qi et al. (2022) addresses the manipulation of elastic rods, the new method reported in this paper has more comprehensive functions and is applicable in real-world scenarios. Specifically, it considers three types of shape geometries (viz. open curves, closed curves, and surfaces) and is not limited to centerline configurations; Various performance criteria are combined with the model estimator, which can better exploit kinematic relationships between deformable objects and robots than Qi et al. (2022). Furthermore, a new MPC-based control method with an integrated GAN-based algorithm is introduced to solve various constraints that exist in real-work environments and to deal with visual occlusions; All these valuable functions are missing in Qi et al. (2022). The reported experimental study also contrasts with that of our earlier method in that we now thoroughly evaluate our controller with shaping motions in 3D. To the best of the authors' knowledge, this is the first time that a shape servoing controller is developed with all the functions proposed in this paper.

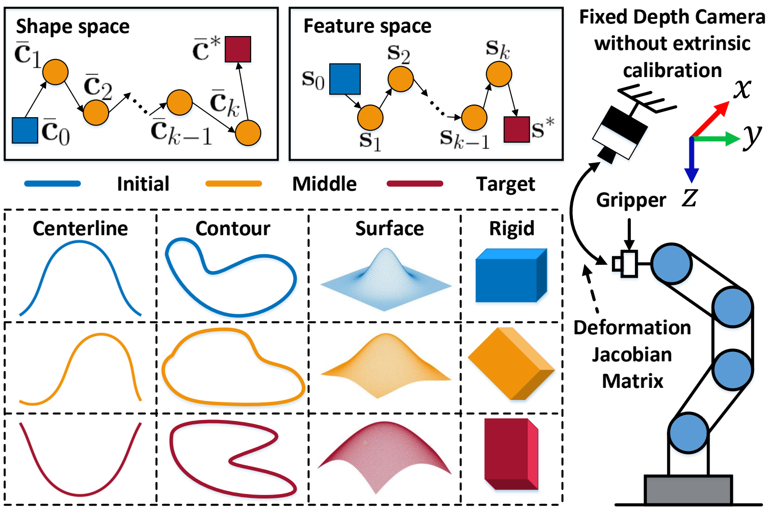


Figure 1: Schematic diagram of the manipulations of elastic objects, including deformations of various shape configurations (centerline, contour, and surface) and positioning tasks of rigid objects. The framework aims to command the robot to manipulate elastic objects into the target configurations. Experiments are conducted in 3D space within eye-to-hand configuration without extrinsic calibration. All shapes are ordered, fixed-sampled and equidistant. The rigid object adopts surface representation.

2. Problem Formulation

Notation: In this paper, we use the following frequently-used notation: Bold small letters, e.g., $\mathbf{v}$, denote column vectors, while bold capital letters, e.g., $\mathbf{M}$, denote matrices. Time evolving variables are denoted as $\mathbf{x}_k$, for k as the discrete time instant. The $n \times m$ matrix of ones is denoted by $\mathbf{I}_{n\times m}$ and the identity matrix as $\mathbf{E}_n$. $\mathbf{L}_n$ represents the low triangle matrix of $\mathbf{I}_{n\times n}$, and $\otimes$ represents the Kronecker product.

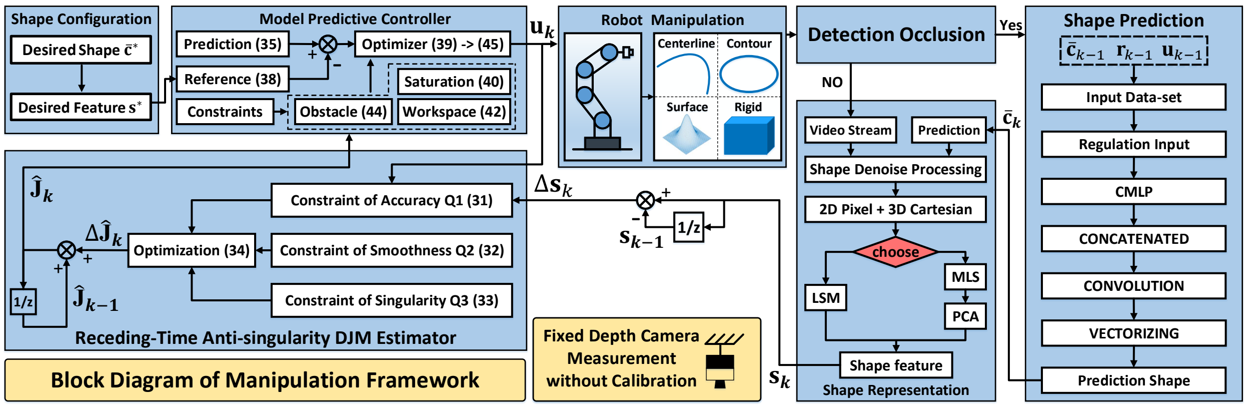


Figure 2: The block diagram of the proposed shape servoing framework, including representation, prediction, approximation, and manipulation within constraints.

The schematic diagram of the proposed shape servoing framework is conceptually illustrated in Fig. 1. A depth camera with eye-to-hand configuration without extrinsic calibration observes the shapes of elastic objects that are manipulated by the robot. Fig. 2 presents the conceptual block diagram of the proposed framework. We denote the 3D measurement points captured by the vision system as:

$$\bar{\mathbf{c}} = \left[\mathbf{c}_1^\intercal, \dots, \mathbf{c}_N^\intercal\right]^\intercal \in \mathbb{R}^{3N}, \quad \mathbf{c}_i = \left[x_i, y_i, z_i\right]^\intercal \in \mathbb{R}^3 \tag{1}$$

for N as the number of points, and $\mathbf{c}_i$ as the 3D coordinates of the ith point, expressed in the camera frame.

2.1. Feature Sensorimotor Model

Let us denote the position of the robot's end-effector by $\mathbf{r} \in \mathbb{R}^3$. As the dimension of the $3N$ observed points $\bar{\mathbf{c}}$ is typically very large, therefore, its direct use as a feedback signal for shape servocontrol is impractical. Therefore, an efficient controller may typically require to use some form of dimension reduction technique Yang, Lu, Chen, Zhong and Liu (2023). To deal with this issue, we construct a feature vector $\mathbf{s} = \mathbf{f}_s(\bar{\mathbf{c}}) : \mathbb{R}^{3N} \mapsto \mathbb{R}^p$, for $p \ll 3N$, to represent the geometric feedback $\bar{\mathbf{c}}$ of the object. This compact feedback-like signal will be used to design our automatic 3D shape controller.

For *purely elastic* objects undergoing deformations, it is reasonable to model that the object configuration is only dependant on its potential energy $\mathcal{P}$ (thus, all inertial and viscous effects are neglected from our analysis Navarro-Alarcon and Liu (2017)). We model that $\mathcal{P}$ is fully determined by the feedback feature vector $\mathbf{s}$ and the robot's position $\mathbf{r}$ Wakamatsu and Hirai (2004), i.e.: $\mathcal{P} = \mathcal{P}(\mathbf{s}, \mathbf{r})$. In steady-state, the extremum expression satisfies $(\partial\mathcal{P}/\partial\mathbf{s})^\intercal = \mathbf{a}(\mathbf{s}, \mathbf{r}) = \mathbf{0}$ Yu, Zhong and Li (2021). We then define the following matrices:

$$\frac{\partial^2\mathcal{P}}{\partial\mathbf{s}^2} = \mathbf{G}(\mathbf{s}, \mathbf{r}), \qquad \frac{\partial^2\mathcal{P}}{\partial\mathbf{r}\partial\mathbf{s}} = \mathbf{K}(\mathbf{s}, \mathbf{r}) \tag{2}$$

which are useful to linearize the extremum equation (by using first-order Taylor's series expansion) as follows:

$$\mathbf{a}(\mathbf{s} + \Delta\mathbf{s}, \mathbf{r} + \Delta\mathbf{r}) \approx \mathbf{a}(\mathbf{s}, \mathbf{r}) + \mathbf{G}(\mathbf{s}, \mathbf{r})\Delta\mathbf{s} + \mathbf{K}(\mathbf{s}, \mathbf{r})\Delta\mathbf{r} \tag{3}$$

for $\Delta\mathbf{s}$ and $\Delta\mathbf{r}$ as small changes. Note that as $\mathbf{a}(\mathbf{s} + \Delta\mathbf{s}, \mathbf{r} + \Delta\mathbf{r}) = \mathbf{a}(\mathbf{s}, \mathbf{r}) = \mathbf{0}$ is satisfied, we can obtain the following motion model:

$$\Delta\mathbf{s} \approx -\mathbf{G}(\mathbf{s}, \mathbf{r})^{-1}\mathbf{K}(\mathbf{s}, \mathbf{r})\Delta\mathbf{r} = \mathbf{J}(\mathbf{s}, \mathbf{r})\mathbf{u} \tag{4}$$

where $\mathbf{J}(\mathbf{s}, \mathbf{r}) = -\mathbf{G}(\mathbf{s}, \mathbf{r})^{-1}\mathbf{K}(\mathbf{s}, \mathbf{r}) \in \mathbb{R}^{p\times 3}$ represents the deformation Jacobian matrix (which depends on both the feature vector and robot position), and $\mathbf{u}$ represents the robot's motion control input. This model can be expressed in an intuitive discrete-time form:

$$\Delta\mathbf{s}_{k+1} = \mathbf{J}_k(\mathbf{s}_k, \mathbf{r}_k)\mathbf{u}_k \tag{5}$$

The deformation Jacobian matrix (DJM) $\mathbf{J}_k$ indicates how the robot's action $\mathbf{u}_k = \mathbf{r}_k - \mathbf{r}_{k-1}$ produces changes in the feedback features $\Delta\mathbf{s}_{k+1} = \mathbf{s}_{k+1} - \mathbf{s}_k$. Clearly, the analytical computation of $\mathbf{J}_k$ requires knowledge of the physical properties and model of the elastic object and the vision system, which are difficult to obtain in practice. Thus, numerical methods are often used to approximate this matrix in real-time, which enables to perform vision-guided manipulation tasks.

In this paper, we consider a robot manipulator whose control inputs represent velocity commands (here, modelled as the differential changes $\Delta\mathbf{r}$). It is assumed that $\Delta\mathbf{r}$ can be instantaneously executed without delay Qi et al. (2021).

Problem statement. Design a vision-based control method to automatically manipulate a compliant object into a target 3D configuration, while simultaneously compensating for visual occlusions of the camera and estimating the deformation Jacobian matrix of the object-robot system.

3. Shape Representation

This section presents how online curve/surface fitting (least-squares minimization (LSM) Abdi et al. (2007) and moving least squares (MLS) Lancaster and Salkauskas (1981)) is combined with a parametric shape descriptor to compute a compact vector of feedback shape features.

3.1. LSM-Based Features

3.1.1. Centerline and Contour Extraction

Both are expressed as a parametric curve dependent on the normalized arc-length $0 \leq \rho \leq 1$. Then, the point can be represented as $\mathbf{c}_i = \mathbf{f}(\rho_i)$, with ρ_i as the arc-length between the start point $\mathbf{c}_1$ and $\mathbf{c}_i$, where $\rho_1 = 0$ and $\rho_N = 1$. The fitting functional $\mathbf{f}(\cdot)$ is constructed as follows:

$$\mathbf{f}(\rho) = \sum_{j=0}^{n} \mathbf{p}_j B_{j,n}(\rho) \tag{6}$$

where the vector $\mathbf{p}_j \in \mathbb{R}^3$ denotes the shape weights, $n \in \mathbb{N}^*$ specifies the fitting order, and the scalar $B_{j,n}(\rho)$ represents a parametric regression function, which may take various forms, such as: Qi, Ma, Navarro-Alarcon, Gao and Ma (2020)

- Polynominal parameterization Smith (1918):

$$B_{j,n}(\rho) = \rho^j \tag{7}$$

- Bernstein parameterization Lorentz (2013):

$$B_{j,n}(\rho) = C_n^j (1-\rho)^{n-j} \rho^j \tag{8}$$

  where C_b^j represents the binomial coefficient.

- Cox-deBoor parameterization Schoenberg (1973):

$$B_{j,n}(\rho) = \frac{1}{n!} \sum_{l=0}^{n-j} \left((-1)^l C_{n+1}^l (\rho + n - j - l)^n \right) \tag{9}$$

- Trigonometric parameterization Powell et al. (1981):

$$B_{j,n}(\rho) = \begin{cases} 1, & j = 0 \\ \cos(\frac{j+1}{2}\rho), & j > 0,\ j \text{ is odd} \\ \sin(\frac{j}{2}\rho), & j > 0,\ j \text{ is even} \end{cases} \tag{10}$$

From (1) and (6), we can compute the following fitting cost function:

$$Q = (\mathbf{Bs} - \bar{\mathbf{c}})^\intercal (\mathbf{Bs} - \bar{\mathbf{c}}) \tag{11}$$

for a “tall” regression-like matrix $\mathbf{B}$ constructed as:

$$\mathbf{B} = [\mathbf{B}_1^\intercal, \dots, \mathbf{B}_N^\intercal]^\intercal \in \mathbb{R}^{3N \times 3(n+1)}, \quad \mathbf{B}_i = [B_{0,n}(\rho_i), \dots, B_{n,n}(\rho_i)] \otimes \mathbf{E}_3 \in \mathbb{R}^{3 \times 3(n+1)} \tag{12}$$

and $\mathbf{s} = [\mathbf{p}_0^\intercal, \dots, \mathbf{p}_n^\intercal]^\intercal \in \mathbb{R}^{3(n+1)}$ as a compact feedback feature vector that represents the shape. We seek to minimize (11) to obtain a feature vector $\mathbf{s}$ that closely approximates $\bar{\mathbf{c}} \approx \mathbf{Bs}$. The solution to (11) is:

$$\mathbf{s} = \left(\mathbf{B}^\intercal \mathbf{B}\right)^{-1} \mathbf{B}^\intercal \bar{\mathbf{c}} \tag{13}$$

where it is assumed that $N \gg n + 1$[1].

[1] In the following sections, Q is used to generically represent different, albeit related, functions.

3.1.2. Surface Extraction

The equation of the surface is defined as follows:

$$z = f(x, y) = \sum_{j=0}^{n_x} \sum_{l=0}^{n_y} B_{j,n_x}(x)\, B_{l,n_y}(y)\, q_{jl} \tag{14}$$

where $n_x, n_y \in \mathbb{N}^*$ are the fitting order along with x and y direction, and $q_{jl} \in \mathbb{R}$ is the shape weight. Same as with (11), as fitting cost function is introduced:

$$Q = (\mathbf{Ds} - \mathbf{z})^\intercal (\mathbf{Ds} - \mathbf{z}) \tag{15}$$

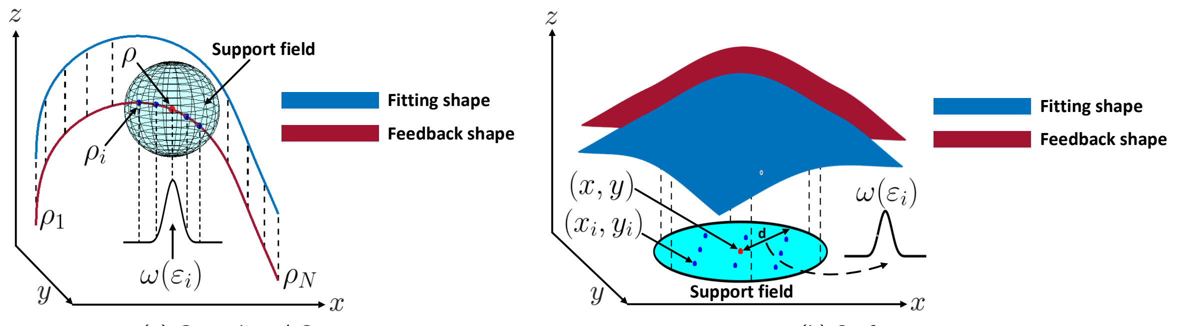
for the depth vector is defined as $\mathbf{z} = [z_1, \dots, z_N]^\intercal \in \mathbb{R}^N$, the feature vector is $\mathbf{s} = [q_{00}, \dots, q_{n_x n_y}]^\intercal \in \mathbb{R}^{(n_x+1)(n_y+1)}$, and a "augmented" regression matrix $\mathbf{D}$ satisfying:

$$\begin{aligned} \mathbf{D} &= [\mathbf{D}_1^\intercal, \dots, \mathbf{D}_N^\intercal]^\intercal \in \mathbb{R}^{N\times(n_x+1)(n_y+1)} \\ \mathbf{D}_i &= [B_{0,n_x}(x_i)B_{0,n_y}(y_i), \dots, B_{n_x,n_x}(x_i)B_{n_y,n_y}(y_i)] \in \mathbb{R}^{1\times(n_x+1)(n_y+1)}, \quad for\ \ i = 1, \dots, N \end{aligned} \tag{16}$$

The solution to the minimization of (15) that approximates $\mathbf{z} \approx \mathbf{Ds}$ is as follows:

$$\mathbf{s} = \left(\mathbf{D}^\intercal\mathbf{D}\right)^{-1}\mathbf{D}^\intercal\mathbf{z} \tag{17}$$

for $N \gg (n_x + 1)(n_y + 1)$.



(a) Centerline / Contour (b) Surface

Figure 3: Scheme diagram of MLS. The weight $\omega(\varepsilon_i)$ is the square error between the fitted value and the given value. The fitting smoothness is regulated by adjusting the weight values.

3.2. MLS-based Feature Extraction

Although LSM has an efficient one-step calculation, the weights of $B_{j,n}$ are the same all over the variable parameters (e.g., ρ or x, y). Thus, to approximate complex curves/surfaces, the fitting order needs to be increased, which may lead to over-fitting problems. To address this issue, MLS assumes that $\mathbf{s}$ is parameter-dependent, i.e., it changes with respect to ρ (for centerline and contour), or to (x, y) (for surface). This enables MLS to represent complex shapes with a lower fitting order. A support field Tey, Che Sidik, Asako, W Muhieldeen and Afshar (2021) is introduced to ensure that the value of each weight is only influenced by data points within the support field. A conceptual diagram is given in Fig. 3.

3.2.1. Centerline and Contour Extraction

The parametric equation with $\boldsymbol{\sigma}_j(\rho)$ is presented by referring to (6):

$$\mathbf{f}(\rho) = \sum_{j=0}^{n} \boldsymbol{\sigma}_j(\rho) B_{j,n}(\rho) \tag{18}$$

where the parameter-dependent weight is denoted as $\boldsymbol{\sigma}_j(\rho) \in \mathbb{R}^3$. The weighted square residual function is defined as:

$$Q(\rho) = \sum_{i=1}^{N} \omega\left(\varepsilon_i\right) \left\| \sum_{j=0}^{n} \boldsymbol{\sigma}_j(\rho) B_{j,n}\left(\rho_i\right) - \mathbf{c}_i \right\|^2 \tag{19}$$

where $\varepsilon_i = |\rho - \rho_i|/d > 0$, and d is the constant support field radius. The scalar function $\omega(\varepsilon_i)$ is calculated as follows Zhang, Guo, Su and Zhu (2015):

$$\omega\left(\varepsilon_i\right) = \begin{cases} \frac{2}{3} - 4\varepsilon_i^2 + 4\varepsilon_i^3, & \varepsilon_i \leq 0.5 \\ \frac{4}{3} - 4\varepsilon_i + 4\varepsilon_i^2 - \frac{4}{3}\varepsilon_i^3, & 0.5 < \varepsilon_i \leq 1 \\ 0, & \varepsilon_i > 1 \end{cases} \tag{20}$$

The function $\omega(\varepsilon_i)$ indicates the weight of ρ relative to ρ_i, $\omega(\varepsilon_i)$ decreases as ε_i increasing inside the support field. When ρ is outside the support field, $\omega(\varepsilon_i) = 0$. MLS reduces into LSM when $\omega(\varepsilon_i)$ is constant. The cost (19) can be equivalently expressed in matrix form as:

$$Q(\rho) = (\mathbf{B}\boldsymbol{\vartheta}(\rho) - \bar{\mathbf{c}})^{\intercal}\, \mathbf{W}(\varepsilon)\, (\mathbf{B}\boldsymbol{\vartheta}(\rho) - \bar{\mathbf{c}}) \tag{21}$$

where $\boldsymbol{\vartheta}(\rho)$ and $\mathbf{W}(\varepsilon)$ are defined as follows:

$$\boldsymbol{\vartheta}(\rho) = [\boldsymbol{\sigma}_0^{\intercal}(\rho), \ldots, \boldsymbol{\sigma}_n^{\intercal}(\rho)]^{\intercal} \in \mathbb{R}^{3(n+1)}, \quad \mathbf{W}(\varepsilon) = \mathrm{diag}(\omega(\varepsilon_1), \ldots, \omega(\varepsilon_N)) \otimes \mathbf{E}_3 \in \mathbb{R}^{3N\times 3N} \tag{22}$$

The value of $\boldsymbol{\vartheta}(\rho)$ that minimizes (21) is computed as follows:

$$\boldsymbol{\vartheta}(\rho) = (\mathbf{B}^{\intercal}\mathbf{W}(\varepsilon)\mathbf{B})^{-1}\mathbf{B}^{\intercal}\mathbf{W}(\varepsilon)\bar{\mathbf{c}} \tag{23}$$

By completing N iterations along the parameter $\rho_1, ..., \rho_N$, we can compute the augmented shape features as:

$$\boldsymbol{\Pi} = [\boldsymbol{\vartheta}(\rho_1), \ldots \boldsymbol{\vartheta}(\rho_N)] \in \mathbb{R}^{3(n+1)\times N} \tag{24}$$

As the dimension of (24) is very large, it is impractical to use its components as a feedback signal for control. Thus, principal components analysis (PCA) Zhu et al. (2021b) is used to reduce the augmented structure (24) into a compact form $\widetilde{\boldsymbol{\Pi}} \in \mathbb{R}^{3(n+1)\times m}$ where $m \ll N$ by selecting the first m most significant dimensions. The feature vector $\mathbf{s} \in \mathbb{R}^{3m(n+1)}$ is computed by vectorizing the elements of $\widetilde{\boldsymbol{\Pi}}$.

3.2.2. *Surface Extraction*

The equation of the surface is constructed as:

$$f(x, y) = \sum_{j=0}^{n_x} \sum_{l=0}^{n_y} B_{j,n_x}(x)\, B_{l,n_y}(y)\, \varpi_{jl}(x, y) \tag{25}$$

where $\varpi_{jl}(x, y) \in \mathbb{R}$ is the parameter-dependent weight related to (x, y). Algorithm 1 gives a pseudocode description of this method.

Remark 1. *In addition to the proposed basis functions, there are other approaches that can be used to obtain similar results, e.g., Chebyshev polynomials Mason and Handscomb (2002) and Legendre basis transformations Farouki (2000).*

4. Shape Prediction Network

During the manipulation process, occlusions caused by obstacles or the robot itself may affect the integrity of observed shapes, and hence, the vector $\mathbf{s}$ cannot properly describe the object's configuration. As a solution to this critical issue, in this paper we propose an occlusion compensation shape prediction network (SPN), which is composed of a regulation input (RI), a multi-resolution encoder (MRE) and a discriminator network (DN) Huang, Yu, Xu, Ni and Le (2020). The proposed SPN utilizes the robot and object configurations and the active robot motions (i.e., $\bar{\mathbf{c}}_k$, $\mathbf{r}_k$, $\mathbf{u}_k$) as input to the network to predict the next instance shape, here denoted by $\hat{\mathbf{c}}_{k+1}$. Fig. 4 shows the overall architecture of the SPN.

Algorithm 1: Surface fitting procedure of MLS.

```
Require: $\bar{\mathbf{c}}$, $n_x$, $n_y$, $m$, and $d$;
1: Calculate node distance: $\varepsilon_i = \sqrt{(x-x_i)^2+(y-y_i)^2}/d$
2: Construct the fitting cost function: $Q = (\mathbf{D}\boldsymbol{\phi}(x,y)-\mathbf{z})^\intercal\mathbf{W}(\varepsilon)(\mathbf{D}\boldsymbol{\phi}(x,y)-\mathbf{z})$
   for $\boldsymbol{\phi}(x,y)$ and $\mathbf{W}(\varepsilon)$ satisfying: $\boldsymbol{\phi} = [\varpi_{00},\dots,\varpi_{n_xn_y}]^\intercal,\quad \mathbf{W} = \mathrm{diag}\left(\omega\left(\varepsilon_1\right),\dots,\omega\left(\varepsilon_N\right)\right)$
3: Compute the structure $\boldsymbol{\phi}(x,y)$: $\boldsymbol{\phi} = (\mathbf{D}^\intercal\mathbf{W}(\varepsilon)\mathbf{D})^{-1}\mathbf{D}^\intercal\mathbf{W}(\varepsilon)\mathbf{z}$
4: Use $x_i, y_i, i \in [1,N]$ to compute: $\boldsymbol{\Pi} = [\boldsymbol{\phi}(x_1,y_1),\dots,\boldsymbol{\phi}(x_N,y_N)]$
5: Use PCA to calculate $\widetilde{\boldsymbol{\Pi}}$;
6: Vectorize $\widetilde{\boldsymbol{\Pi}}$ to obtain $\mathbf{s}$;
7: return $\mathbf{s}$;
```

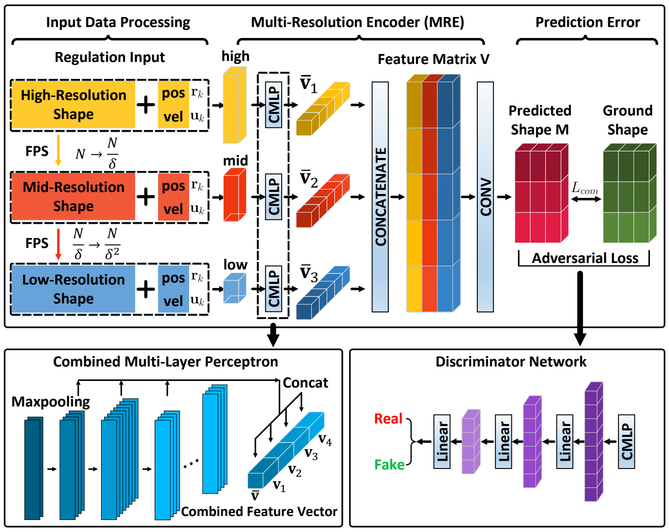


Figure 4: SPN predicts the next-moment shape $\bar{\mathbf{c}}_{k+1}$ by using the current-moment data in the case of occlusion, i.e., $\bar{\mathbf{c}}_k + \mathbf{r}_k + \mathbf{u}_k \rightarrow \hat{\bar{\mathbf{c}}}_{k+1}$. FPS Yan et al. (2020b) regulates $\bar{\mathbf{c}}_k$ (shown in yellow) uniformly to different scales (shown in red and blue). MRE stacks three-resolutions data together to form the total feature information, and obtains the predicted next-moment shape through convolution. DN is used to further improve the accuracy of the network.

4.1. Input Data Preprocessing

As the input data $\bar{\mathbf{c}}_k, \mathbf{r}_k, \mathbf{u}_k$ to the network have different sizes, they need to be rearranged into structures with unified dimensions. To this end, $\bar{\mathbf{c}}_k$ is first rearranged into the matrix $\mathbf{T}_k^{high} = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_N]^\intercal \in \mathbb{R}^{N\times 3}$. Then, farthest point sampling (FPS) Yan, Zheng, Li, Wang and Cui (2020b) is used to downsample $\mathbf{T}_k^{high}$ to two resolutions $\mathbf{T}_k^{mid} \in \mathbb{R}^{\frac{N}{\delta}\times 3}$ and $\mathbf{T}_k^{low} \in \mathbb{R}^{\frac{N}{\delta^2}\times 3}$, for δ as the resolution scale. Finally, the vectors $\mathbf{r}_k$ and $\mathbf{u}_k$ are rearranged into the matrices $\boldsymbol{\Lambda}_k^{high} = \mathbf{I}_{N\times 1} \otimes \mathbf{r}_k^\intercal \in \mathbb{R}^{N\times 3}$ and $\boldsymbol{\Sigma}_k^{high} = \mathbf{I}_{N\times 1} \otimes \mathbf{u}_k^\intercal \in \mathbb{R}^{N\times 3}$, which are similarly downsampled into mid and low resolutions as follows:

$$\boldsymbol{\Lambda}_k^{mid} = \mathbf{I}_{\frac{N}{\delta}\times 1} \otimes \mathbf{r}_k^\intercal, \quad \boldsymbol{\Lambda}_k^{low} = \mathbf{I}_{\frac{N}{\delta^2}\times 1} \otimes \mathbf{r}_k^\intercal, \quad \boldsymbol{\Sigma}_k^{mid} = \mathbf{I}_{\frac{N}{\delta}\times 1} \otimes \mathbf{u}_k^\intercal, \quad \boldsymbol{\Sigma}_k^{low} = \mathbf{I}_{\frac{N}{\delta^2}\times 1} \otimes \mathbf{u}_k^\intercal \tag{26}$$

Thus, three different resolutions are generated for the network, high $\{\mathbf{T}_k^{high}, \boldsymbol{\Lambda}_k^{high}, \boldsymbol{\Sigma}_k^{high}\}$, mid $\{\mathbf{T}_k^{mid}, \boldsymbol{\Lambda}_k^{mid}, \boldsymbol{\Sigma}_k^{mid}\}$, and low $\{\mathbf{T}_k^{low}, \boldsymbol{\Lambda}_k^{low}, \boldsymbol{\Sigma}_k^{low}\}$, that provides a total input data of dimension $[N \times 9, \frac{N}{\delta} \times 9, \frac{N}{\delta^2} \times 9]$. $\mathbf{T}_k^{high}, \mathbf{T}_k^{mid}, \mathbf{T}_k^{low}$ are the geometric shapes of the object under different compression sizes specified by δ. Thus, these three terms can describe the potential feature structure of objects. The proposed SPN aims to predict the object's shape that results from the robots actions, under the current object-robot configuration. By using this multi-resolution data $\{$high, mid, low$\}$, the encoder can better learn the potential feature information of shapes.

4.2. Multi-Resolution Encoder

A combined multi-layer perceptron (CMLP) is the feature extractor of MRE, which uses the output of each layer in a MLP to form a multiple-dimensional feature vector. Traditional methods adopt the last layer of the MLP output as features, and do not consider the output of the intermediate layers, which leads to potentially losing important local information Qi, Su, Mo and Guibas (2017). CMLP enables to make good use of low-level and mid-level features that include useful intermediate-transition information Huang et al. (2020). CMLP utilizes MLP to encode input data into multiple dimensions $[64, 128, 256, 512, 1024]$. Then, we maxpool the output of the last four layers to construct a multiple-dimensional feature vector as follows:

$$\mathbf{v}_1 \in \mathbb{R}^{128}, \quad \mathbf{v}_2 \in \mathbb{R}^{256}, \quad \mathbf{v}_3 \in \mathbb{R}^{512}, \quad \mathbf{v}_4 \in \mathbb{R}^{1024} \tag{27}$$

The combined feature vector is constructed as: $\bar{\mathbf{v}}_i = [\mathbf{v}_1^{\intercal}, \mathbf{v}_2^{\intercal}, \mathbf{v}_3^{\intercal}, \mathbf{v}_4^{\intercal}]^{\intercal} \in \mathbb{R}^{1920}$. Three independent CMLPs map three resolutions into three individual $\bar{\mathbf{v}}_i$, for $i = 1, 2, 3$. Each $\bar{\mathbf{v}}_i$ represents the extracted potential information of each resolution. Then, the augmented feature matrix $\mathbf{V}$ is generated by arranging its columns as $\mathbf{V} = [\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3] \in \mathbb{R}^{1920\times 3}$, and further through 1D-convolution to obtain $\mathbf{M} \in \mathbb{R}^{N\times 3}$. Finally, the predicted next-moment shape $\hat{\bar{\mathbf{c}}}_{k+1} \in \mathbb{R}^{3N}$ is obtained by vectorizing $\mathbf{M}$. The prediction loss of MRE is:

$$L_{mre} = \left\| \hat{\bar{\mathbf{c}}}_{k+1} - \bar{\mathbf{c}}_{k+1} \right\| \tag{28}$$

where $\bar{\mathbf{c}}_{k+1}$ represents the ground-truth next-moment shape in the training dataset.

4.3. Discriminator Network

Generative Adversarial Network (GAN) is chosen as DN to enhance the prediction accuracy. For simplicity, we define $\Phi = \text{MRE}()$ and $\Psi = \text{DN}()$. We define $(\bar{\mathbf{c}}_k, \mathbf{r}_k, \mathbf{u}_k)$ as the $\mathcal{X}$ input into Φ, while $\mathcal{Y}$ represents the true shape $\bar{\mathbf{c}}_{k+1}$. Ψ is a classification network with a similar structure as CMLP, constituted by serial MLP layers $[128, 256, 512, 1024]$ to distinguish the predicted shape $\Phi(\mathcal{X})$ and the real shape $\mathcal{Y}$. We maxpool the last three layers of Ψ to obtain feature vector $[256, 512, 1024]$. Three feature vectors are concatenated into a latent vector $\mathbf{m} \in \mathbb{R}^{1792}$, and then passed through the fully-connected layers $[512, 256, 64, 1]$ followed by sigmoid-classifier to obtain the evaluation. The adversarial loss is defined as follows:

$$L_{adv} = \sum_{1\le i\le v} \log\left(1 - \Psi\left(\beta_i\right)\right) + \sum_{1\le j\le v} \log\left(\Psi\left(\Phi(\alpha_i)\right)\right) \tag{29}$$

where $\alpha_i \in \mathcal{X}, \beta_i \in \mathcal{Y}, i = 1, \dots, v$, and v is size of the dataset including $\mathcal{X}$ and $\mathcal{Y}$. The total loss of SPN is:

$$L = \zeta_{mre} L_{mre} + \zeta_{adv} L_{adv} \tag{30}$$

where ζ_{mre} and ζ_{adv} are the weights of L_{mre} and L_{adv}, respectively, which satisfy the condition: $\zeta_{mre} + \zeta_{adv} = 1$.

5. Receding-time Model Estimation

In this paper, the objects are assumed to be manipulated by the robot slowly, thus $\mathbf{J}_k(\mathbf{s}_k, \mathbf{r}_k)$ is expected to change smoothly. For ease of presentation, we omit the arguments of $\mathbf{J}_k(\mathbf{s}_k, \mathbf{r}_k)$ and denote it as as $\mathbf{J}_k$ from now on. To estimate the DJM, three indicators are considered, viz., accuracy, smoothness, and singularity. To this end, an optimization-based receding-time model (RTM) estimator is presented to estimate the changes of the Jacobian matrix, denoted by $\Delta\hat{\mathbf{J}}_k = \hat{\mathbf{J}}_k - \hat{\mathbf{J}}_{k-1}$, which enables to monitor the estimation procedure. $\Delta\hat{\mathbf{J}}_k$ can be obtained by considering the following three constraints:

- (Q_1) Constraint of receding-time error Mo, Ouyang, Xing, Dong, Liu and Sun (2020). As $\mathbf{J}_k$ depicts the relationship between $\Delta\mathbf{s}_{k+1}$ and $\mathbf{u}_k$ in a local range, thus we consider the accumulated error in η past moments to ensure the estimation accuracy. η is the receding window size. The receding-time error is given by:

$$Q_1 = \sum_{j=1}^{\eta} \gamma^j \left\| \Delta\mathbf{s}_{k+1-j} - (\hat{\mathbf{J}}_{k-1} + \Delta\hat{\mathbf{J}}_k)\mathbf{u}_{k-j} \right\|^2 \tag{31}$$

The sensitivity to noise can be improved by adjusting η, which helps to address the measurement fluctuations. $0 < \gamma \le 1$ is a constant forgetting factor giving less weight to the past observation data.

- (Q_2) Constraint of estimation smoothness Mo et al. (2020). As $\mathbf{J}_k$ is assumed to be smooth, thus $\Delta\hat{\mathbf{J}}_k$ should be estimated smoothly to avoid sudden large fluctuations, which can be achieved by minimizing the Frobenius norm of $\Delta\hat{\mathbf{J}}_k$:

$$Q_2 = \|\Delta\hat{\mathbf{J}}_k\|_F^2 \tag{32}$$

- (Q_3) Constraint of shape manipulability Lynch and Park (2017). It evaluates the feasibility of changing the object's shape under the current object-robot configuration:

$$Q_3 = \left\| \frac{\lambda_{\max}((\hat{\mathbf{J}}_{k-1} + \Delta\hat{\mathbf{J}}_k)^\intercal(\hat{\mathbf{J}}_{k-1} + \Delta\hat{\mathbf{J}}_k))}{\lambda_{\min}((\hat{\mathbf{J}}_{k-1} + \Delta\hat{\mathbf{J}}_k)^\intercal(\hat{\mathbf{J}}_{k-1} + \Delta\hat{\mathbf{J}}_k))} \right\|^2 \tag{33}$$

  where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum eigenvalue, respectively, and $Q_3 \geq 1$. When $Q_3 = 1$, the object can deform isotropically in any direction. A growing Q_3 indicates that the object is reaching a singular (non-manipulable) configuration.

Finally, the total weighted optimization index is given by:

$$Q(\Delta\hat{\mathbf{J}}_k) = \mu_1 Q_1 + \mu_2 Q_2 + \mu_3 Q_3 \tag{34}$$

where $\mu_i > 0$ are the weights that specify the contribution of each constraint, and which satisfy $\mu_1 + \mu_2 + \mu_3 = 1$. The index (34) is a nonlinear optimization problem. Thus, the general nonlinear solver (e.g., *Matlab/fmincon* or *Python/SciPy*) is used. After determining the constraints (Q_1, Q_2, and Q_3), through iterative update, $\Delta\hat{\mathbf{J}}_k$ can be obtained. Finally, the deformation Jacobian matrix is calculated by $\hat{\mathbf{J}}_k = \hat{\mathbf{J}}_{k-1} + \Delta\hat{\mathbf{J}}_k$.

6. Model Predictive Controller

It is assumed that the matrix $\mathbf{J}_k$ has been accurately estimated at the time instant k by the RTM, such that it satisfies $\Delta\mathbf{s}_{k+1} = \hat{\mathbf{J}}_k\mathbf{u}_k$. Based on this model, we propose an MPC-based controller to derive the velocity inputs $\mathbf{u}_k$ for the robot, while taking saturation, workspace, and obstacle constraints into account. Two vectors are defined as follow:

$$\bar{\mathbf{s}}_k = [\mathbf{s}_{k+1|k}^\intercal, \dots, \mathbf{s}_{k+h|k}^\intercal]^\intercal \in \mathbb{R}^{ph}, \quad \bar{\mathbf{u}}_k = [\mathbf{u}_{k|k}^\intercal, \dots, \mathbf{u}_{k+h-1|k}^\intercal]^\intercal \in \mathbb{R}^{3h} \tag{35}$$

where $\bar{\mathbf{s}}_k$ and $\bar{\mathbf{u}}_k$ represent the predictions of $\mathbf{s}_k$ and $\mathbf{u}_k$ in the next h periods, respectively. The vectors $\mathbf{s}_{k+i|k}$ and $\mathbf{u}_{k+i|k}$ denote the ith predictions of $\mathbf{s}_k$ and $\mathbf{u}_k$ from the time instant k, where $\mathbf{s}_{k|k} = \mathbf{s}_k$, and $\mathbf{u}_{k|k} = \mathbf{u}_k$ must hold. The prediction $\bar{\mathbf{s}}_k$ can be calculated from the estimated Jacobian matrix by noting that $\hat{\mathbf{J}}_k \approx \hat{\mathbf{J}}_{k+h}$ is satisfied during period $[k, k+h]$ (which is reasonable, given the regularity of the object). This way, the predictions are computed as follows:

$$\mathbf{s}_{k+j|k} = \mathbf{s}_k + \sum_{i=0}^{j-1} \hat{\mathbf{J}}_k\mathbf{u}_{k+i|k}, \quad j = 1, \dots, h \tag{36}$$

All predictions are then grouped and arranged into a single vector form:

$$\bar{\mathbf{s}}_k = \mathbf{A}\mathbf{s}_k + \boldsymbol{\Theta}\bar{\mathbf{u}}_k, \quad \mathbf{A} = \mathbf{I}_{h\times 1} \otimes \mathbf{E}_p \in \mathbb{R}^{ph\times p}, \quad \boldsymbol{\Theta} = \mathbf{L}_h \otimes \hat{\mathbf{J}}_k \in \mathbb{R}^{ph\times 3h} \tag{37}$$

In addition to $\bar{\mathbf{s}}_k$ and $\bar{\mathbf{u}}_k$, we define the constant sequence vector $\bar{\mathbf{s}}_k^*$ that represents the desired shape feature as:

$$\bar{\mathbf{s}}_k^* = \left[\mathbf{s}_{k+1|k}^{*\intercal}, \dots, \mathbf{s}_{k+h|k}^{*\intercal}\right]^\intercal \in \mathbb{R}^{ph} \tag{38}$$

The cost function $Q\left(\bar{\mathbf{u}}_k\right)$ for the optimization of the control input is formulated as:

$$Q\left(\bar{\mathbf{u}}_k\right) = \left(\bar{\mathbf{s}}_k - \bar{\mathbf{s}}_k^*\right)^\intercal \boldsymbol{\Upsilon}_1 \left(\bar{\mathbf{s}}_k - \bar{\mathbf{s}}_k^*\right) + \bar{\mathbf{u}}_k^\intercal \boldsymbol{\Upsilon}_2 \bar{\mathbf{u}}_k \tag{39}$$

where $\boldsymbol{\Upsilon}_1 > 0$ and $\boldsymbol{\Upsilon}_2 > 0$ are the weights for the error convergence rate and the smoothness of $\mathbf{u}_k$, respectively. Three constraints are considered:

- *Saturation limits.* In practice, robots have limits on their achievable joint speeds. These constraints are useful in soft object manipulation tasks to avoid damaging the object. Therefore, $\bar{\mathbf{u}}_k$ needs to be constrained:

$$\bar{\mathbf{u}}_{\min} \leq \bar{\mathbf{u}}_k \leq \bar{\mathbf{u}}_{\max} \tag{40}$$

  where $\bar{\mathbf{u}}_{\min}$ and $\bar{\mathbf{u}}_{\max}$ are the constant lower and upper bounds, respectively. Generally, $\bar{\mathbf{u}}_{\min} = \mathbf{0}$.

- *Workspace limits.* Robots are also often required to operate in a confined workspace to avoid colliding with the environment. In soft object manipulation, this constraint is needed to avoid over-stretching or over compressing the manipulated object. To this end, the following constant bounds are introduced:

$$\mathbf{r}^{\min}_{k+i|k} \leq \mathbf{r}_{k+i|k} \leq \mathbf{r}^{\max}_{k+i|k}, \quad 0 \leq i \leq h-1 \tag{41}$$

  Similarly as in (36), the recursive structure of (41) can be obtained follows:

$$\mathbf{\Xi}_{\min} \leq \mathbf{C}\bar{\mathbf{u}}_k \leq \mathbf{\Xi}_{\max} \tag{42}$$

  for $\mathbf{C} = \mathbf{L}_h \otimes \mathbf{E}_3 \in \mathbb{R}^{3h\times 3h}$ and $\mathbf{\Xi}_{\min}, \mathbf{\Xi}_{\max} \in \mathbb{R}^{3h}$ defined as:

$$\mathbf{\Xi}_{\min} = [(\mathbf{r}^{\min}_{k|k} - \mathbf{r}_{k-1})^\intercal, ..., (\mathbf{r}^{\min}_{k+h-1|k} - \mathbf{r}_{k-1})^\intercal]^\intercal, \quad \mathbf{\Xi}_{\max} = [(\mathbf{r}^{\max}_{k|k} - \mathbf{r}_{k-1})^\intercal, ..., (\mathbf{r}^{\max}_{k+h-1|k} - \mathbf{r}_{k-1})^\intercal]^\intercal \tag{43}$$

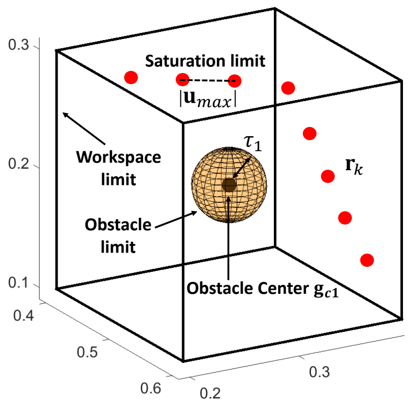
- *Obstacle limits.* During the motion, the end-effector's trajectory $\mathbf{r_k}$ should avoid certain areas (e.g., obstacles) in the workspace, which we model as sphere. Let $\mathbf{g}_{cj} = [g_{xj}, g_{yj}, g_{zj}]^\intercal \in \mathbb{R}^3$ and $\tau_j, j \in [1, v]$ be the center and radius of these spherical regions, respectively, with v as the number of the obstacles. The obstacle avoidance constraint is designed as:

$$\|\mathbf{r}_{k+i|k} - \mathbf{g}_{cj}\| \geq \tau_j, \quad 0 \leq i \leq h-1, \quad 1 \leq j \leq v \tag{44}$$

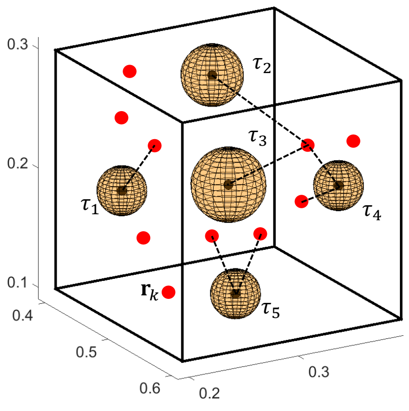
where $\mathbf{r}_{k+i|k} = \mathbf{r}_{k+i-1|k} + \mathbf{u}_{k+i|k}$. Considering that (40) and (42) are linear constraints and (44) is nonlinear, general nonlinear solvers (e.g., *Matlab/fmincon* or *Python/SciPy*) are used to solve (39) instead of traditional quadratic solvers.

Finally, $\mathbf{u}_k$ is calculated by the receding horizon scheme:

$$\mathbf{u}_k = [\mathbf{E}_3, \mathbf{0}, \dots, \mathbf{0}]\bar{\mathbf{u}}_k \tag{45}$$



(a)

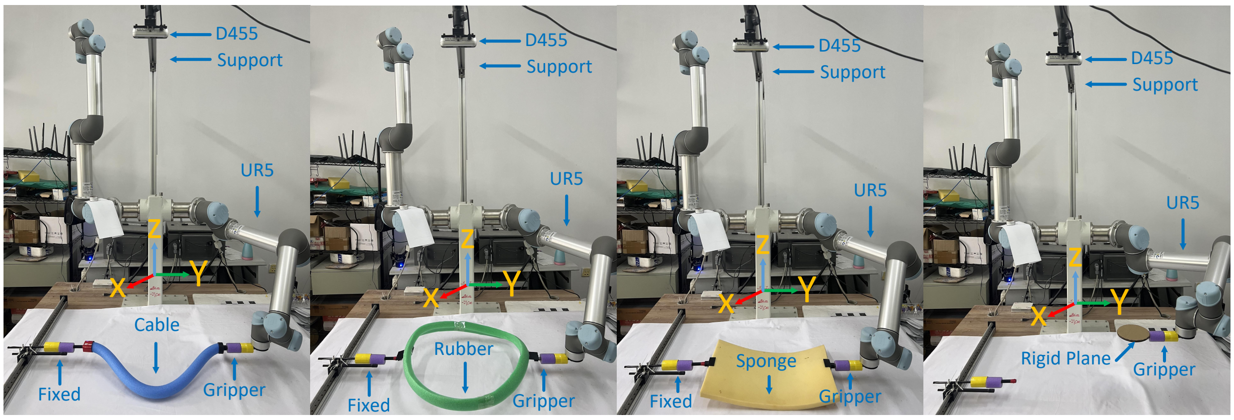


(b)

Figure 5: (a): Illustrations of saturation, workspace, and obstacle limit. The black line represents the constrained workspace, and the yellow one is the sphere obstacle area with the black dot as the center $\mathbf{g}_{c1}$ and the radius τ_1. The red dot indicates the end-effector's trajectory $\mathbf{r}_k$, and the distance between the two points is the velocity command $\mathbf{u}_k$. (b): Multiple obstacle areas with different obstacle radius in the constrained workspace, i.e., $v = 5$.

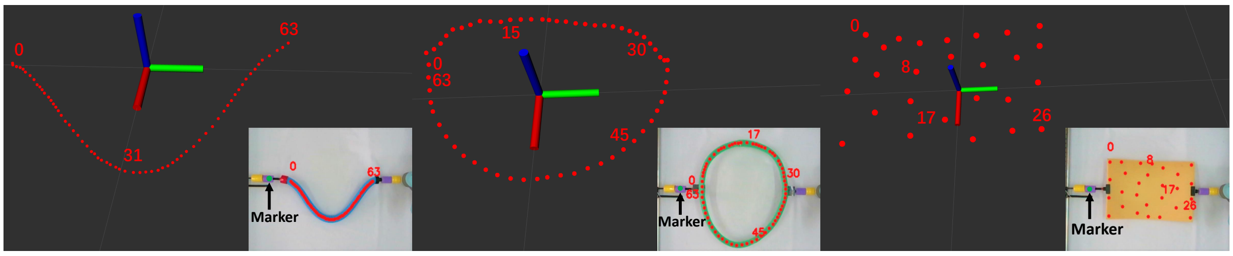
Remark 2. *The proposed MPC-based technique computes the robot's shaping actions based on a performance objective and subject to system's constraints; This approach does not require the identification of the full analytical model of the deformable object. Its quadratic optimization form enables to integrate additional metrics into the problem, e.g., rising time and overshoot.*

Remark 3. *The workspace limit constrains the allowable volume of the end-effector, while obstacle limit denotes specific areas where the end-effector cannot pass. Fig. 5 illustrates three constraints. Note that the spherical areas in Fig. 3 and Fig. 5 represent different things, the former is the support field of MLS, and the latter is the obstacle area.*



(a) Centerline (b) Contour (c) Surface (d) Plane

Figure 6: The experimental setup, including the objects (elastic and rigid), single-arm robot (UR5), and D455 without extrinsic calibration.



(a) Centerline (b) Contour (c) Surface

Figure 7: Shape extraction for centerline, contour and surface. Base pictures are 3D shapes drawn in ROS/RVIZ, and the thumbnail show the 2-D image pixels. All shapes are fixed-sampled, equidistant, and ordered sorting.

7. Results

7.1. Experimental Setup

Vision-based manipulation experiments are conducted to validate our proposed framework. The experimental platform includes a fixed D455 depth sensor without extrinsic calibration, a single-arm UR5, and various deformable objects shown in Fig. 6. The depth sensor receives the video stream, from which it computes the 3D shapes by using the OpenCV and RealSense libraries. As previous studies have demonstrated that rotations produce very little effect on deformations Navarro-Alarcon and Liu (2014), thus, due to page limitation, we use 3 linear DOF in our experiments. Note that our manipulation framework can adapt to arbitrary DOF. Therefore, the control input $\mathbf{u} = [u_x, u_y, u_z]^\intercal \in \mathbb{R}^3$

represents the linear velocity of the end-effector; A saturation limit of $|u_i| \leq 0.01$ m/s, is applied for $i = x, y, z$. The motion control algorithm is implemented on ROS/Python, which runs with a servo-control loop of 10 Hz.

The proposed shape extraction algorithm is depicted in Fig. 7. The RGB image from the camera is transformed into HSV and combined with mask processing to obtain a binary image. $OpenCV/thinning$ is utilized with FPS to extract a fixed number of object points. The point on the centerline closest to the gripper's green marker is chosen to sort the centerline along the cable. Then we obtain the 3D shapes by using the RealSense sensor and checking its 2D pixels. We adopt the method in Qi et al. (2021) to compute the object's contour. A surface is obtained in the similar way to the centerline, i.e., by sorting points from top to bottom, and from left to right. As 2D pixels and 3D points have a one-to-one correspondence in a depth camera, thus, our extraction method improves the robustness to measurement noise and is simpler than traditional point cloud processing algorithms.

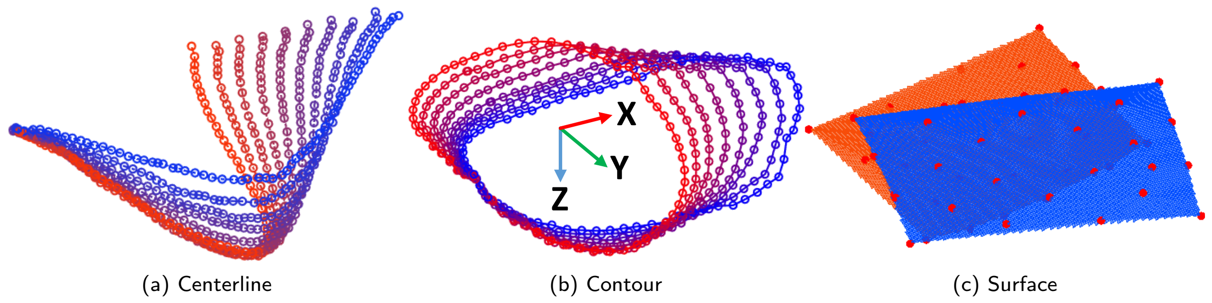


Figure 8: Shapes of various objects manipulated by UR5.

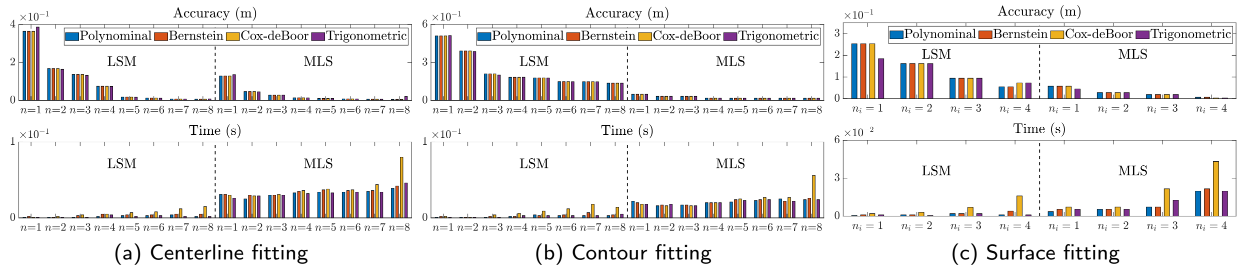


Figure 9: Comparison of centerline, contour and surface fitting in accuracy and time-consuming among (7)(8)(9)(10) between LSM and MLS with $d = 0.9$, $d = 0.1$, and $d = 0.2$ for three shape configurations, respectively. $n_i, i = x, y$ are the fitting order used in the surface fitting.

7.2. Online Fitting of the Parametric Shape Representation

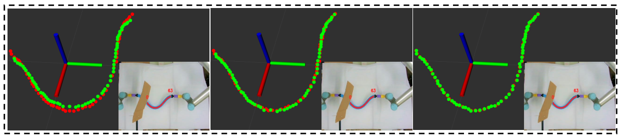
In this section, ten thousand samples of centerlines, contours and surfaces with $N = 64, 64, 27$, respectively, are collected by commanding the robot to manipulate the objects, whose configuration is then captured by a depth sensor. Such shaping actions are shown in Fig. 8, and visualized in the accompanying multimedia attachment. This data is used to evaluate the performance (viz. its accuracy and computation time) of our representation framework. For that, we calculate the average error between the feedback shape $\bar{\mathbf{c}}$ and the reconstructed shape $\hat{\mathbf{c}}$ as follows mean$(\sum \left\| \bar{\mathbf{c}}_i - \hat{\mathbf{c}}_i \right\|)$. The computation time is defined as the average of the overall processing time of all sample data among each method.

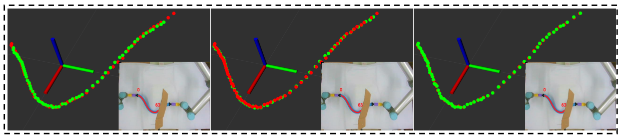
Fig. 9 shows that the larger the scalars n, n_x, n_y are, the better the fitting accuracy of LSM and MLS is. MLS fits better than LSM under the same condition, as MLS calculates the independent weight while LSM assumes that each node has the same weight. MLS works better in fitting contour because the parametric curve may not be continuous in the end corner, thus, the equal weight assumption of LSM is not suitable here. As the number of data points for surface is $N = 27$, it does not satisfy the condition $N \gg (n_x + 1)(n_y + 1)$ for higher order fitting models (e.g., $n_x = n_y \geq 5$), thus, we only use $n_x = n_y \leq 4$. The results show that MLS performs better than LSM in the surface representation; Interestingly, MLS obtains satisfactory performance even with $n_x = n_y = 1$. Fig. 9 also shows that larger

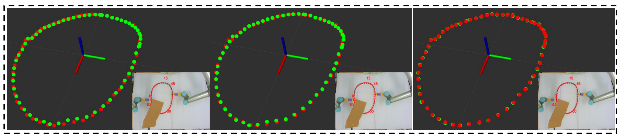
Table 1
Fitting configurations. "N/A" stands for "Not Applicable".

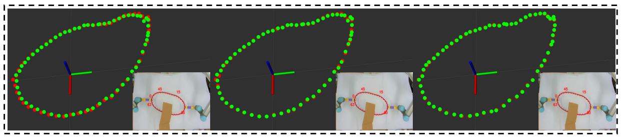
	Centerline	Contour	Surface / Rigid
Method	LSM	MLS	MLS
Support radius	N/A	$d = 0.2$	$d = 0.2$
PCA	N/A	$m = 1$	$m = 1$
Fitting order	$n = 6$	$n = 4$	$n_x = n_y = 2$
Basis function	Bernstein	Trigonometric	Polynominal
Numbers	$N = 64$	$N = 64$	$N = 27$

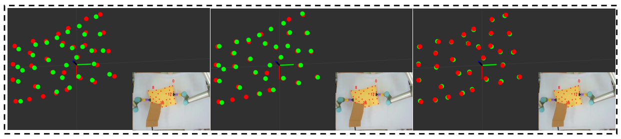
n, n_x, n_y will also increase the computation time. MLS has a more noticeable increase, as it calculates the weights of all nodes while LSM calculates them once. The trigonometric approach is the fastest, polynomial and Bernstein follow, while Cox-deBoor is the slowest with the most iterative operations. The above analysis verifies the effectiveness of the proposed extraction framework, which can represent objects with a low-dimensional feature. Details of the shape fitting configuration is given in Table. 1.


(a) Centerline - case 1


(b) Centerline - case 2


(c) Contour - case 1


(d) Contour - case 2


(e) Surface - case 1

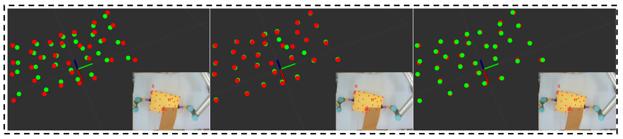

(f) Surface - case 2

Figure 10: Validation of the occlusion removal among three shape configurations with moving the obstacle manually. The RGB image describes the occluded case. In each subfigure, from left to right, is method1 Tang et al. (2017), method2 Wang et al. (2020), and the proposed SPN. Red dot gives the ground-truth shape $\bar{\mathbf{c}}_{k+1}$ within the occlusion at the current instant, and green ones are the predicted shape $\hat{\bar{\mathbf{c}}}_{k+1}$ according to the past data.

7.3. Occlusion-Robust Prediction of Object Shapes

This section aims to compare SPN with Tang et al. (2017) and Wang et al. (2020) in evaluating the shape prediction ability. The object is manipulated by the robot with the pre-defined motion ($\mathbf{u}_k$ and $\mathbf{r}_k$) to obtain the shape $\bar{\mathbf{c}}_{k+1}$, i.e., the input dataset ($\bar{\mathbf{c}}_k, \mathbf{u}_k, \mathbf{r}_k$), the output dataset $\bar{\mathbf{c}}_{k+1}$ is complete without occlusion during the sampling process. 80% of the data is used as the training set, and the remaining 20% is used as the test set, with $\delta = 2$ for centerline and contour, and $\delta = 3$ for surface style. SPN is built using PyTorch and trained by an ADAM optimizer with a batch size of 500, and the initial learning rate set to 0.0001. RELU activation and batch normalization are adopted to improve the network's performance.

During the test, the robot deforms the objects with small babbling-like motions while a cardboard sheet covers parts of objects. Fig. 10 gives the comparison results among three methods, with the RGB image depicting the occlusion situation. These results show that three methods can predict and provide relatively complete shapes for the manipulated

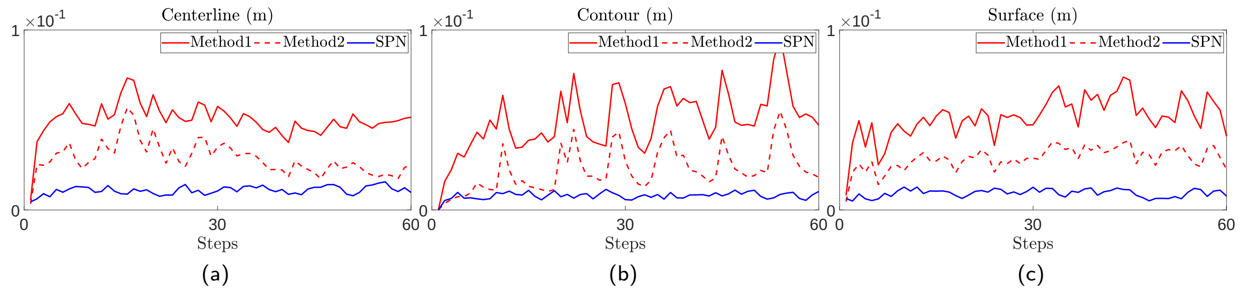


Figure 11: The prediction error $\|\bar{\mathbf{c}}_{k+1} - \hat{\bar{\mathbf{c}}}_{k+1}\|$ among three methods over three shape configurations in the occlusion-removal evaluation task.

objects. Fig. 11 gives the profiles of the prediction error $\|\bar{\mathbf{c}}_{k+1} - \hat{\bar{\mathbf{c}}}_{k+1}\|$ for the three methods. In terms of accuracy, SPN is the best and does not show significant fluctuations, Wang et al. (2020) is the second best, and Tang et al. (2017) is the worst. Also for further comparison, we simultaneously evaluate the effect of different methods on occlusion size. In Fig. 11, we see that when the occlusion is large (corresponding to the peak points in the figure), Tang et al. (2017) and Wang et al. (2020) cannot predict the shape well (due to the inherent limitations of the pointcloud registration algorithms used by both methods). In contrast, as the proposed SPN models the kinematics of the object, it can predict even with large size occlusions by suitable training. In terms of speed, the average calculation time of Tang et al. (2017), Wang et al. (2020) and ours are 0.11ms 0.08ms, and 0.03ms, respectively. As Tang et al. (2017) and Wang et al. (2020) have many iterative operations, thus, they are not as good as SPN in terms of their performance. In addition, we found that ζ_{mre} has a greater impact on the training accuracy than ζ_{adv}, as ζ_{mre} dominates the training, and ζ_{adv} is only used to improve accuracy. We set $\zeta_{mre} = 0.87$ and $\zeta_{adv} = 0.13$ in the experiments. The above results show that SPN is efficient and with high prediction accuracy. The accompanying video demonstrates the performance with experiments.

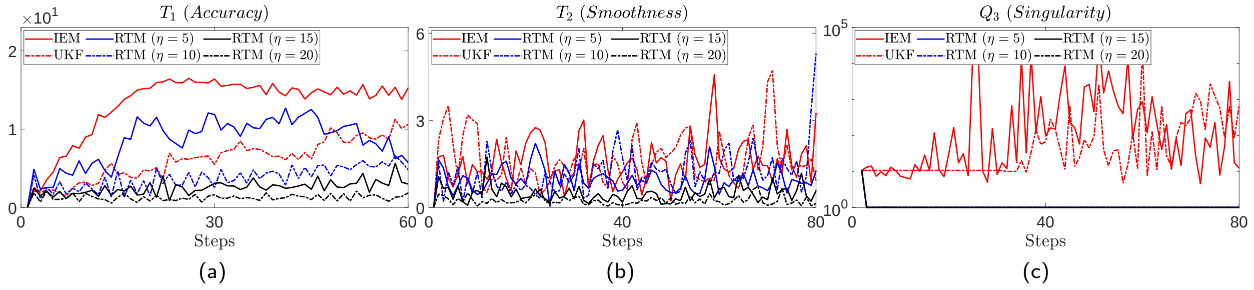


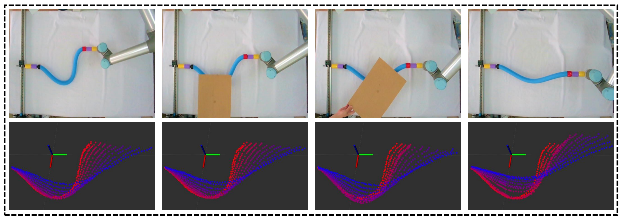
Figure 12: Curves of T_1 (feature estimation error), T_2 (differential estimation error), and Q_3, reviewing the accuracy, smoothness, and singularity of DJM, respectively. Besides, $\mu_1 = 0.8, \mu_2 = 0.1, \mu_3 = 0.1$.

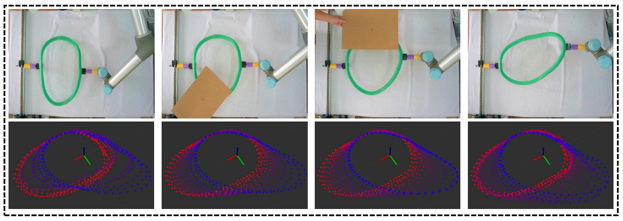
7.4. Estimation of the Sensorimotor Model

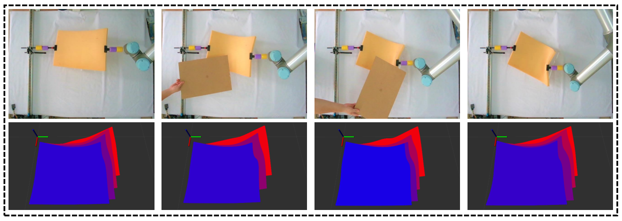
This section aims to evaluate the RTM in (34) that approximates the deformation Jacobian matrix, and compare with the interaction matrix estimator (IEM) in Zhu et al. (2021b), and the unscented Kalman filter (UKF) in Qi et al. (2020). Same as before, the robot moves along a pre-defined motion command $\mathbf{r}_k$. Two metrics (T_1 and T_2) are introduced to quantitatively compare performances of these algorithms:

$$T_1 = \|\hat{\mathbf{s}}_{k+1} - \mathbf{s}_{k+1}\|, \quad T_2 = \|\Delta\mathbf{s}_{k+1} - \hat{\mathbf{J}}_k\mathbf{u}_k\| \tag{46}$$

where $\hat{\mathbf{s}}_{k+1} = \hat{\mathbf{s}}_k + \hat{\mathbf{J}}_k\mathbf{u}_k$ is the approximated shape feature that is computed based on the control actions. Due to the page limitation, the comparison is achieved in the centerline style, shown in Fig. 12. It shows that RTM ($\eta = 20$) approximates best, as illustrated in T_1; This means that the constraint Q_1 (31) enables RTM to learn from past data efficiently by adjusting η. Low T_2 reflects that RTM accurately predicts the differential changes induced by the DJM; This means that constraint Q_2 (32) helps to compute a smooth matrix $\hat{\mathbf{J}}_k$. As the constraint Q_3 (33) is incorporated, thus, RTM can prevent singularities in the estimation of the Jacobian matrix, while IEM and UKF are prone to reach ill-conditioned estimations. We use RTM with $\eta = 10$ in the following sections.


(a) Exp1 / Centerline


(b) Exp2 / Contour


(c) Exp3 / Surface

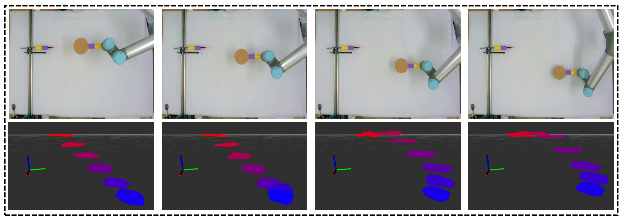
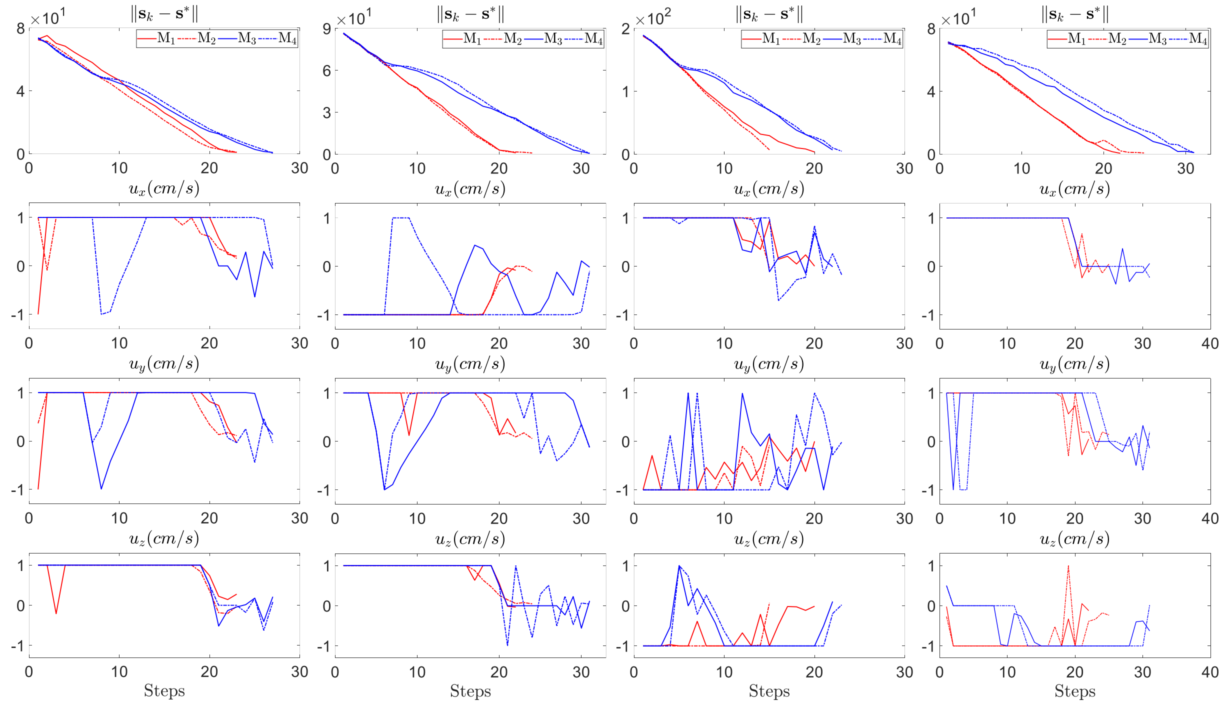

(d) Exp4 / Rigid Plane

Figure 13: Shape manipulation experiments, Exp1, Exp2, and Exp3 are for the deformation tasks, and Exp4 is the positioning task. The first row in each subfigure shows 2D image manipulation process (from left to right are initial, intermediate, intermediate, and desired shape), and the second row represents 3D shape manipulation process of each method. The obstacle moves randomly during the process. The desired shape is shown with blue, the red represents the start shapes, and the gradient color are intermediate shapes.



(a) Exp1 / Centerline (b) Exp2 / Contour (c) Exp3 / Surface (d) Exp4 / Rigid Plane

Figure 14: Profiles of the manipulation error $\|\mathbf{s}_k - \mathbf{s}^*\|$ and the velocity command $\mathbf{u}_k$ among four experiments (Exp1, ..., Exp4). Each experiment is achieved by four methods, i.e., $\mathrm{M}_1, \ldots, \mathrm{M}_4$.

7.5. Automatic Shape Servoing Control

This section conducts four automatic manipulation experiments, labeled as Exp1, Exp2, Exp3, and Exp4, respectively. The desired shape $\bar{\mathbf{c}}^*$ is obtained from previous demonstrations of the manipulation task, which ensures its reachability. A cardboard sheet is manually placed over the object to produce (partial) occlusions and test the robustness of our algorithm. Two learning-based manipulation frameworks (Hu et al. (2019) and Yan, Zhu, Jin and Bohg (2020a)) are compared with the proposed receding-time model ($\eta = 10$) with MPC ($h = 10$ and $h = 30$). We label these methods as $\mathrm{M}_1, \ldots, \mathrm{M}_4$ in sequence, and each method has been optimized to achieve a balance of stability, convergence, and responsiveness. In addition to the feedback shape error $\|\mathbf{s}_k - \mathbf{s}^*\|$, we also compare $T_{\max}$ (i.e., the number of steps from start to finish), d_{eff} (the total moving distance of the end-effector), success rate (we repeat each task five times to calculate), and the percentage of the final error over the initial error (the smaller the better) Lagneau et al. (2020b).

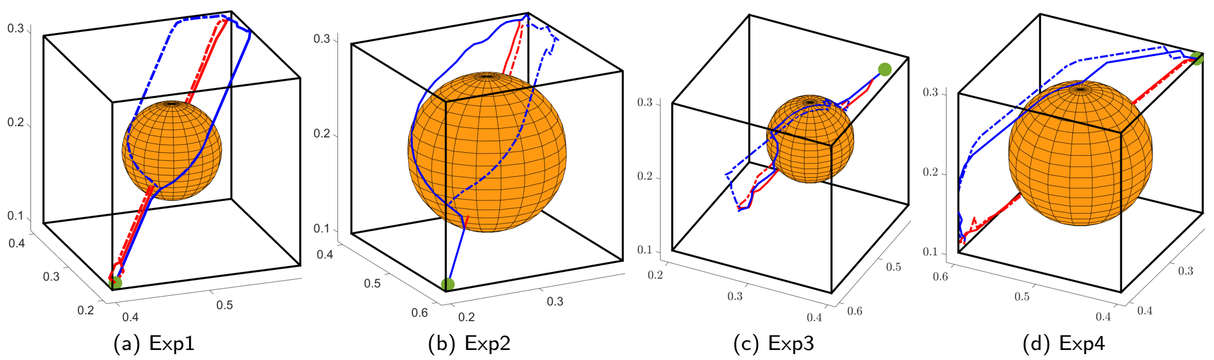

(a) Exp1 (b) Exp2 (c) Exp3 (d) Exp4

Figure 15: The 3D motion trajectories $\mathbf{r}_k$ of the end-effector among four methods. Black dashed line represents the constrained workspace. The yellow area is the obstacle area. The green dot indicates the initial position of the end-effector. The legend information is same as that in Fig. 14. The unit of the coordinate axis is meter (m).

Fig. 13 shows the shaping motions of the objects manipulated from initial shape (red curve) to the desired configuration (blue curve), with the cardboard blocking the view at various instances. The results demonstrate that all methods can complete the task well, and SPN can assist the manipulation through predicting the object's shape to solve occlusions and feeding it back to the controller to enforce the shape servo-loop, which confirms the effectiveness and compatibility with other methods of SPN. Fig. 15 gives the end-effector's trajectories in the Cartesian coordinate system. The results show that M_3 and M_4 can simultaneously avoid the obstacle area and work within the constrained workspace, whereas the other two methods cannot perform this. Interestingly, MPC also works well when avoiding a large obstacle (reflected in Exp2 and Exp4 with bigger sphere), as function that is particularly important in practice. In contrast, the error norm $\|\mathbf{s}_k - \mathbf{s}^*\|$ plots in Fig. 14 show that M_1 and M_2 provides the fastest control performance for the error minimization, with M_3 performing better than M_4. In addition, M_3 and M_4 move along the boundary of the obstacle area, which shows that MPC can find the optimal path in a restricted situation. As hard input saturation is implemented in this work (for safety reasons), thus, all methods satisfy the control saturation constraint, the detailed velocity command is shown in Fig. 14. Exp4 shows that our framework has good universality, not only for the shape control, but also for traditional rigid object positioning.

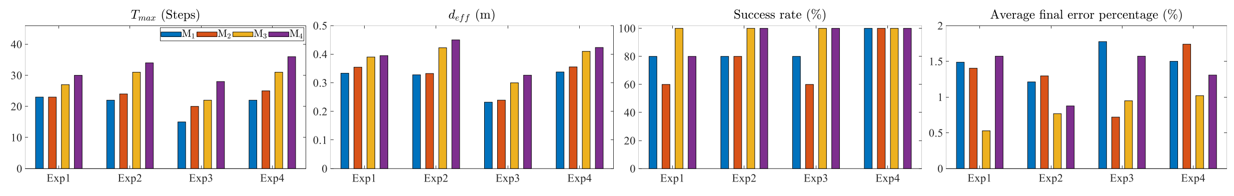


Figure 16: Performance comparison in the manipulation tasks (Exp1 to Exp4).

Generally, a higher h is helpful for feature prediction, yet, since we assume that $\widehat{\mathbf{J}}_k$ is constant in the window period h, it may lead to inaccurate predictions and even wrong manipulation of objects (reflection is that M_4 is worse than M_3). Therefore, h should be chosen according to the performance requirements of the system. A performance comparison (four indices) of these experiments is given in Fig. 16. It illustrates that our methods (M_3 and M_4) have a high success rate, with the final manipulation error being the smallest one. Since obstacle avoidance is considered, $T_{\max}$ and d_{eff} of M_3 and M_4 are larger than the other two. The above results demonstrate that our proposed shape servoing framework can effectively conduct deformation and positioning tasks in a dynamic process (solved by RTM), under occlusions (solved by SPN), and with various constraints (solved by MPC).

8. Conclusions

In this paper, we present an occlusion-robust shape servoing framework to control shapes of elastic objects into target configurations, while considering workspace, saturation, and obstacle constraints. A low-dimensional feature extractor is proposed to represent 3D shapes based on LSM and MLS. A deep neural network is introduced to predict the object's configuration subject to occlusions and feed it to the shape servo-controller. A receding-time model estimator is designed to approximate the deformation Jacobian matrix with various constraints such as accuracy, smoothness, and singularity. The conducted experiments validate the proposed methodology with multiple unstructured shape servoing tasks in visually occluded situations and with unknown deformation models.

However, there are some limitations in our framework. For example, as the support field radius d is constant (i.e., it does not adjust with dynamic shapes), the computed representation lacks flexibility. Also, the SPN needs to obtain substantial offline training data to properly work, which might pose complications in practice. Note that our method may not accurately shape objects with negligible elastic properties (e.g., fabrics, food materials, etc). Future work includes the incorporation of shape reachability detection into the framework in order to determine the feasibility of a given shaping task beforehand.

References

Abdi, H., et al., 2007. The method of least squares. Encyclopedia of Measurement and Statistics. CA, USA: Thousand Oaks .

Alambeigi, F., Wang, Z., Hegeman, R., Liu, Y.H., Armand, M., 2018. Autonomous data-driven manipulation of unknown anisotropic deformable tissues using unmodelled continuum manipulators. IEEE Robotics and Automation Letters 4, 254–261.

Cao, L., Li, X., Phan, P.T., Tiong, A.M.H., Kaan, H.L., Liu, J., Lai, W., Huang, Y., Le, H.M., Miyasaka, M., et al., 2020. Sewing up the wounds: A robotic suturing system for flexible endoscopy. IEEE Robotics & Automation Magazine 27, 45–54.

Cazy, N., Wieber, P.B., Giordano, P.R., Chaumette, F., 2015. Visual servoing when visual information is missing: Experimental comparison of visual feature prediction schemes, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 6031–6036.

Chaumette, F., Hutchinson, S., 2006. Visual servo control, part I: Basic approaches. IEEE Robotics and Automation Magazine 13, 82–90.

Farouki, R.T., 2000. Legendre–bernstein basis transformations. Journal of Computational and Applied Mathematics 119, 145–160.

Hajiloo, A., Keshmiri, M., Xie, W.F., Wang, T.T., 2015. Robust online model predictive control for a constrained image-based visual servoing. IEEE Transactions on Industrial Electronics 63, 2242–2250.

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

Hu, Z., Han, T., Sun, P., Pan, J., Manocha, D., 2019. 3-d deformable object manipulation using deep neural networks. IEEE Robotics and Automation Letters 4, 4255–4261.

Hu, Z., Sun, P., Pan, J., 2018. Three-dimensional deformable object manipulation using fast online gaussian process regression. IEEE Robotics and Automation Letters 3, 979–986.

Huang, Z., Yu, Y., Xu, J., Ni, F., Le, X., 2020. Pf-net: Point fractal network for 3d point cloud completion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7662–7670.

Lagneau, Romain, K., Alexandre, M., Maud, 2020a. Automatic shape control of deformable wires based on model-free visual servoing. IEEE Robotics and Automation Letters 5, 5252–5259.

Lagneau, R., Krupa, A., Marchal, M., 2020b. Active deformation through visual servoing of soft objects, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 8978–8984.

Lancaster, P., Salkauskas, K., 1981. Surfaces generated by moving least squares methods. Mathematics of computation 37, 141–158.

Li, X., Su, X., Liu, Y.H., 2018. Vision-based robotic manipulation of flexible pcbs. IEEE/ASME Transactions on Mechatronics 23, 2739–2749.

Lorentz, G.G., 2013. Bernstein polynomials. American Mathematical Soc.

Lv, K., Yu, M., Pu, Y., Jiang, X., Huang, G., Li, X., 2023. Learning to estimate 3-d states of deformable linear objects from single-frame occluded point clouds, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 7119–7125. doi:10.1109/ICRA48891.2023.10160784.

Lynch, K.M., Park, F.C., 2017. Modern robotics. Cambridge University Press.

Mason, J.C., Handscomb, D.C., 2002. Chebyshev polynomials. CRC press.

Mo, H., Ouyang, B., Xing, L., Dong, D., Liu, Y., Sun, D., 2020. Automated 3-d deformation of a soft object using a continuum robot. IEEE Transactions on Automation Science and Engineering .

Navarro-Alarcon, D., Liu, Y.h., 2014. 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. pp. 4457–4462.

Navarro-Alarcon, D., Liu, Y.H., 2017. Fourier-based shape servoing: a new feedback method to actively deform soft objects into desired 2-d image contours. IEEE Transactions on Robotics 34, 272–279.

Navarro-Alarcon, D., Liu, Y.H., Romero, J.G., Li, P., 2013. Model-free visually servoed deformation control of elastic objects by robot manipulators. IEEE Transactions on Robotics 29, 1457–1468.

Navarro-Alarcon, D., Liu, Y.h., Romero, J.G., Li, P., 2014. On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments. The International Journal of Robotics Research 33, 1462–1480.

Navarro-Alarcon, D., Qi, J., Zhu, J., Cherubini, A., 2020. A lyapunov-stable adaptive method to approximate sensorimotor models for sensor-based control. Frontiers in Neurorobotics 14.

Powell, M.J.D., et al., 1981. Approximation theory and methods. Cambridge university press.

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 652–660.

Qi, J., Ma, G., Zhou, P., Zhang, H., Lyu, Y., Navarro-Alarcon, D., 2022. Towards latent space based manipulation of elastic rods using autoencoder models and robust centerline extractions. Advanced Robotics 36, 101–115.

Qi, J., Ma, G., Zhu, J., Zhou, P., Lyu, Y., Zhang, H., Navarro-Alarcon, D., 2021. Contour moments based manipulation of composite rigid-deformable objects with finite time model estimation and shape/position control. IEEE/ASME Transactions on Mechatronics .

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

Sanchez, J., Corrales, J.A., Bouzgarrou, B.C., Mezouar, Y., 2018. Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. Int. J. of Rob. Res. .

Schoenberg, I.J., 1973. Cardinal spline interpolation. SIAM.

Smith, K., 1918. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations. Biometrika 12, 1–85.

Tang, T., Fan, Y., Lin, H.C., Tomizuka, M., 2017. State estimation for deformable objects by point registration and dynamic simulation, in: The 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017).

Tang, T., Tomizuka, M., 2018. Track deformable objects from point clouds with structure preserved registration. The International Journal of Robotics Research , 0278364919841431.

Tang, T., Wang, C., Tomizuka, M., 2018. A framework for manipulating deformable linear objects by coherent point drift. IEEE Robotics and Automation Letters 3, 3426–3433.

Tey, W.Y., Che Sidik, N.A., Asako, Y., W Muhieldeen, M., Afshar, O., 2021. Moving least squares method and its improvement: A concise review. Journal of Applied and Computational Mechanics 7, 883–889.

Wakamatsu, H., Hirai, S., 2004. Static modeling of linear object deformation based on differential geometry. The International Journal of Robotics Research 23, 293–311.

Wang, H., Yang, B., Wang, J., Liang, X., Chen, W., Liu, Y.H., 2018. Adaptive visual servoing of contour features. IEEE/ASME Transactions on Mechatronics 23, 811–822.

Wang, Y., Mcconachie, D., Berenson, D., 2020. Tracking partially-occluded deformable objects while enforcing geometric constraints .

Yan, M., Zhu, Y., Jin, N., Bohg, J., 2020a. Self-supervised learning of state estimation for manipulating deformable linear objects. IEEE robotics and automation letters 5, 2372–2379.

Yan, X., Zheng, C., Li, Z., Wang, S., Cui, S., 2020b. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5589–5598.

Yang, B., Lu, B., Chen, W., Zhong, F., Liu, Y.H., 2023. Model-free 3-d shape control of deformable objects using novel features based on modal analysis. IEEE Transactions on Robotics .

Yin, H., Varava, A., Kragic, D., 2021. Modeling, learning, perception, and control methods for deformable object manipulation. Science Robotics 6.

Yu, M., Zhong, H., Li, X., 2021. Shape control of deformable linear objects with offline and online learning of local linear deformation models. arXiv preprint arXiv:2109.11091 .

Zhang, H., Guo, C., Su, X., Zhu, C., 2015. Measurement data fitting based on moving least squares method. Mathematical Problems in Engineering 2015.

Zhou, P., Zhu, J., Huo, S., Navarro-Alarcon, D., 2021. LaSeSOM: A latent and semantic representation framework for soft object manipulation. IEEE Robotics and Automation Letters 6, 5381–5388.

Zhu, J., Cherubini, A., Dune, C., Navarro-Alarcon, D., Alambeigi, F., Berenson, D., Ficuciello, F., Harada, K., Li, X., Pan, J., et al., 2021a. Challenges and outlook in robotic manipulation of deformable objects. arXiv preprint arXiv:2105.01767 .

Zhu, J., Navarro-Alarcon, D., Passama, R., Cherubini, A., 2021b. Vision-based manipulation of deformable and rigid objects using subspace projections of 2d contours. Robotics and Autonomous Systems 142, 103798.