Vision-based Online Learning Kinematic Control for Soft Robots using Local Gaussian Process Regression

Ge Fang*, Xiaomei Wang*, Kui Wang, Kit-Hang Lee, Justin D.L. Ho, Hing-Choi Fu, Denny K.C. Fu and Ka-Wai Kwok, *Senior Member*, *IEEE*

Abstract— **Soft robots, owing to their elastomeric material, ensure safe interaction with their surroundings. These robot compliance properties inevitably impose a trade-off against precise motion control, as to which conventional model-based methods were proposed to approximate the robot kinematics. However, too many parameters, regarding robot deformation and external disturbance, are difficult to obtain, even if possible, which could be very nonlinear. Sensors self-contained in the robot are required to compensate modelling uncertainties and external disturbances. Camera (eye) integrated at the robot end-effector (hand) is a common setting. To this end, we propose an eye-in-hand visual servo that incorporates with learning-based controller to accomplish more precise robotic tasks. Local Gaussian process regression (GPR) is used to initialize and refine the inverse mappings online, without prior knowledge of robot and camera parameters. Experimental validation is also conducted to demonstrate the hyper-elastic robot can compensate an external variable loading during trajectory tracking.**

Index Terms— **Eye-in-hand visual-servo, Learning-based control, Local Gaussian process regression, Soft robot control.**

I. Introduction

Soft robots made of elastomeric materials [1, 2] have attracted increasing research interest. This is accredited not only to their high-power density [3] actuation, but also their adaptability with confined and unstructured surroundings. These resolve various manipulation challenges commonly encountered in robotic tasks demanding for safe interaction with human, e.g. manufacturing [4], exoskeleton/wearable devices [5] and minimally invasive surgery [6]. However, their flexibility, as well as nonlinear actuated deformation usually hinder their uses in precise manipulation, compared to their rigid counterparts.

To develop effective control strategies, several models have been investigated to approximate the kinematics behavior of limber manipulators without skeletons [7]. The piecewise constant curvature (PCC) assumption was popularly applied to simplify the bending kinematics of continuum robots with uniform shape and symmetrical actuation [8, 9]. Combined with positional sensors, e.g. electro-magnetic trackers [10], the PCC-based geometric solution enabled real-time closed-loop control of the robot pose in free space. Recent work utilized the PCC assumption and a self-contained curvature sensor to control the locomotion of a soft robotic snake [11]. Parallel kinematics was investigated to achieve position control of a soft robot using elastomer strain sensors [12]. Other modelling approaches, such as those based on the Cosserat rod theory [13, 14], have been used to investigate the kinematics mapping by establishing force equilibrium, which can account for gravity and external load. But unknown disturbance to the robot, such as unpredictable payload and interaction with surroundings, can promptly deteriorate the model. Finite element modelling (FEM) was also applied to accurately estimate complex robot deformations, by which the kinematics mapping could be generated and incorporated in the soft robot control [15]. Recent works described that asynchronous FEM could be combined with a quadratic programming algorithm to achieve real-time control of soft robots [16]. But the modeling accuracy is sensitive to geometric and material parameters, of which the searches are also heuristic. Moreover, the aforementioned models are design-specific to particular robot structures.

Data-driven control approaches circumvent analytical modeling by deriving the kinematics mapping or control policies from acquired sensing data. Neural networks (NNs) have been studied to approximate the global inverse mapping of nonredundant soft continuum robots [17, 18]. NNs could be specifically designed to learn a global mapping accurately, but it is not efficient for online learning because all network parameters have to be updated in every iteration. Novel data-based approach [19] was also applied optimal control to estimate the kinematic Jacobian matrix online. It demonstrated stable control of tendon-driven continuum robot in a 2D statically constrained environment. Recently, we have also proposed a locally weighted online learning controller [20] used in the 3D orientation control of a fluid-driven soft robot. It could encounter with externally applied disturbance; however, the needs for heuristic tuning of multiple data-dependent parameters becomes the major weakness of this approach [21].

Apart from accurate inverse mapping, closing the control loop with sensing feedback is also essential. Vision-based systems are a viable choice for integration with soft robots, as they can be small and self-contained. Making use of camera feedback, visual servoing has been extensively studied over the last decades and many approaches have been proposed

This work is supported by the Croucher Foundation, the Research Grants Council (RGC) of Hong Kong (Ref. No.: 17202317, 17227616, 27209515).

G. Fang, X. Wang, K. Wang, K.H. Lee, Justin D.L. Ho, H.C. Fu, Denny K.C. Fu and K.W. Kwok are with Department of Mechanical Engineering, The University of Hong Kong, Hong Kong (corresponding author, Tel: +852-3917-2636; e-mail: kwokkw@hku.hk).

* indicates co-first authorships

[22, 23]. Wang et al. [24] first achieved eye-in-hand visual servo control of a cable-driven soft robot based on analytical kinematic modelling and an interaction matrix, where the intrinsic and extrinsic camera parameters are estimated beforehand. Other studies addressed visual servo of a concentric-tube robot [25] and series pneumatic artifical muscles (sPAMs) [26] by estimating the task space Jacobian matrix from image feedback. But these controllers were only validated in free space. Recently, an PCC-based adaptive visual servo controller was proposed for a cable-driven robot in a constrained environment [27]. It could handle the control of robot statically constrained by physical interaction.

In this paper, we propose an adaptive eye-in-hand visual servo control framework based on local online learning technique. The controller is constructed by learning the inverse mapping solely from collected camera images, without any prior knowledge of the robot and camera parameters. Promising accuracy in learning of inverse mapping is assured without having to tune the hyper parameters in the learning approach. Localized GPR models enable fast online update in other to accommodate new input data that reflect the latest robot status. As a result, precise manipulation can be achieved even when the robot encounters unknown and varying external disturbances. The major contributions of this work are:

i) First attempt to address a learning-based visual servo control for a fluid-driven soft robot such that the inverse kinematics can be directly approximated by local Gaussian process regression (GPR);

ii) Efficient update of inverse motion mapping to compensate *dynamic* disturbance by adjusting the most relevant local GPR model;

iii) Novel experimental validations demonstrating precise point tracking and path following of a hyper-elastic low-stiffness soft robot with *variable* tip load.

II. Methodology

A. Task space definition

A camera mounted at robot end-effector allows image-based control strategy, namely eye-in-hand visual servo. Mappings from the spaces of actuation, configuration to task have to be defined successively. The actuator input (at equilibrium) is represented as $\boldsymbol{\alpha}(k) \in U^m$ at time step k, where U^m denotes the m-dimensional actuation space. Let $\mathbf{s}(k)$ be the manipulator configuration under input $\boldsymbol{\alpha}(k)$, which corresponds to an end-effector position $\mathbf{p}(k) \in \mathrm{R}^3$ and orientation normal $\mathbf{n}(k) \in \mathrm{R}^3$ in the Cartesian space. The collective variable $\boldsymbol{\theta}(k) = [\mathbf{p}(k), \mathbf{n}(k)] \in \mathrm{R}^6$ depends on robot configuration $\mathbf{s}(k)$:

$$\boldsymbol{\theta}(k) = h(\mathbf{s}(k)) \tag{1}$$

With quasi-static movement, the forward transition model can be expressed as:

$$\Delta\mathbf{s}(k) = f(\mathbf{s}(k), \Delta\boldsymbol{\alpha}(k)) \tag{2}$$

where $\Delta\boldsymbol{\alpha} = \boldsymbol{\alpha}(k+1) - \boldsymbol{\alpha}(k)$ is the difference of inputs between time step k and k+1, and $\Delta\mathbf{s}(k) = \mathbf{s}(k+1) - \mathbf{s}(k)$ represents the change of robot configuration due to the input difference $\Delta\boldsymbol{\alpha}(k)$.

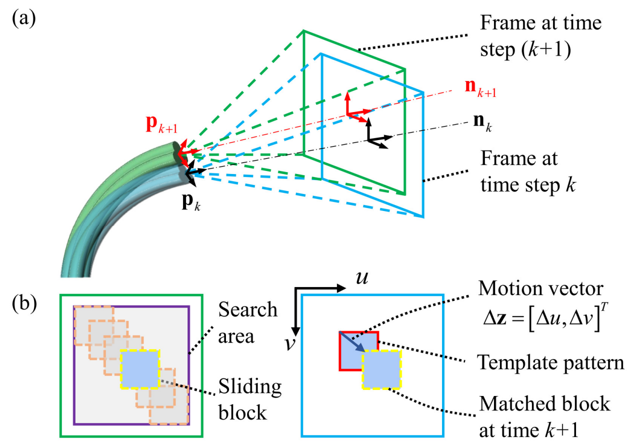


Fig. 1. Schematic diagram of motion estimation. **(a)** Camera coordinate frames at time step k and k+1; **(b)** Incremental motion in image plane can be acquired based on the displacement from the template pattern (in red block) to the matched block (in yellow) that is searched by block sliding.

The task space is defined in the camera frame (**Fig. 1a**), with the incremental displacement denoted as $\Delta\mathbf{z}(k) \in \mathrm{R}^2$. The frame is always perpendicular to the robot tip normal. Combining with the mapping from end-effector states $\boldsymbol{\theta}(\cdot)$ to camera frames, equation **(2)** can thereby be extended to:

$$\Delta\mathbf{z}(k) = g(\mathbf{s}(k), \Delta\boldsymbol{\alpha}(k)) \tag{3}$$

The control objective is to generate the actuation command, achieving a desired movement $\Delta\mathbf{z}^*(k)$ in task space, mapping **(4)** is necessary to approximate the inverse kinematics (IK) of **(3)**, i.e.,

$$\Delta\boldsymbol{\alpha}(k) = \hat{\Phi}(\mathbf{s}(k), \Delta\mathbf{z}^*(k)) \tag{4}$$

The inverse transition $\hat{\Phi}$ heavily depends on the current robot configuration $\mathbf{s}(k)$ that supposes to be an unknown without any sensing data of the end-effector pose. However, this can be resolved by a new IK function of the actuator input $\boldsymbol{\alpha}(k)$ that is defined based on a direct mapping from $\mathbf{s}(k)$ to $\boldsymbol{\alpha}(k)$ during quasi-static movements. Hence, such an inverse mapping is presented as:

$$\Delta\boldsymbol{\alpha}(k) = \Phi(\boldsymbol{\alpha}(k), \Delta\mathbf{z}^*(k)) \tag{5}$$

which approximates the true inverse transition $\hat{\Phi}$ as in **(4)**. We proposed to employ image feedback $\Delta\mathbf{z}(\cdot)$, actuation input $\boldsymbol{\alpha}(\cdot), \Delta\boldsymbol{\alpha}(\cdot)$ to directly estimate the robot IK Φ, but without having to construct an analytical (kinematics) model.

B. Motion estimation on image plane

To "learn" the IK mapping using experimental data, an effective algorithm to measure the end-effector motion $\Delta\mathbf{z}(\cdot)$ with respect to camera frames, i.e. motion on the image plane, is in demand. To estimate the 2D incremental movement between two successive frames, a target square of image intensity features is defined as reference for comparison (**Fig. 1b**). We assume that the orientation change of the robot is small within a short time interval (20Hz). The change of camera orientation is primarily attributed to rotation about the camera normal, which was found to be <5° between successive frames. This small range of rotation corresponds to a movement error of <3 pixels when calculated from template matching. Therefore, in the cases of continuous robot movements, the camera frames would follow a more-or-less planar motion. The translational displacement in this image plane can be estimated using block template matching method (**Fig. 1b),** named "*matchTemplate*" in OpenCV [28]. A square block at time step *k* is first selected as the target template. At time (*k*+1), the same size of block is sliding along the image plane to search for a block containing the intensity pattern that is coherent to the template. The coherence value is calculated referring to the metric function "TM_CCORR_NORMED" as:

$$R(\xi,\eta)=\frac{\sum_{i,j}\left(T(i,j)\cdot I(i+\xi,j+\eta)\right)}{\sqrt{\sum_{i,j}T(i,j)^2\cdot\sum_{i,j}I(i+\xi,j+\eta)^2}} \tag{6}$$

where T and I represent the intensities of template and the sliding block, i and j are the indices of pixels in the local blocks, ξ and η are the movements of sliding block along u and v axes of camera view respectively. The motion vector $\Delta\mathbf{z}(k)=[\Delta u,\Delta v]^T$ between two successive frames is therefore obtained using "*minMaxLoc"* function, by maximizing the coherence in **(6)**:

$$[\Delta u,\Delta v]=\arg\max_{\xi,\eta}R(\xi,\eta) \tag{7}$$

We assume that axial movement of the camera has minimal effect on the template matching process, relative to lateral movement. This is because axial motion results in mainly scaling of the tracked objects rather than translation. In addition, template matching is performed between successive frames, which means those differences in scale will be small while having a reasonable rate (≥20Hz) of camera imaging. During continuous motion, the template matching is iteratively updated referring to the previous frame, providing an accurate displacement estimation successively.

C. Local GPR based control

Learning the inverse mapping from the camera motion to actuator input is a regression problem. Gaussian process regression (GPR) is a nonparametric method by-design to approximate nonlinearity [29]. In our case, it would resolve the parametric uncertainties induced by soft robot fabrication and camera calibration, and even the noise of camera feedback. After the initialization using GPR, the inverse mapping model is updated online with the newly collected sensing data during the execution of the robot. Additionally, with a locally weighted learning scheme, we can increase the computational efficiency for both predicting and updating.

1) Gaussian process regression

Training: As given in **Section IIA**, the inverse mapping $\Delta\boldsymbol{\alpha}=\Phi(\boldsymbol{\alpha},\Delta\mathbf{z}^*)$ will be learned, where the input is defined as $\mathbf{x}=[\boldsymbol{\alpha}^T,\Delta\mathbf{z}^{*T}]^T\in\mathrm{R}^n$ and output as $\mathbf{y}=\Delta\boldsymbol{\alpha}\in\mathrm{R}^m$. A training dataset is collected from the real robot for model initialization. Consider the training set with input data $\mathbf{X}=\{\mathbf{x}_i\}$ and output data $\mathbf{Y}=\{\mathbf{y}_i\}$, $i=1,2,\ldots,N$, where each dimension of the output $\mathbf{y}^s=\{y_i^s\},s=1,2,...,m$ is independently trained. GPR assumes that the input and output of the training data satisfies a nonlinear mapping $y_i^s=G(\mathbf{x}_i)+\varepsilon$, where ε is a white Gaussian noise with zero mean and variance σ_n^2. The output is modeled as Gaussian distribution $\mathbf{y}^s\sim N\left(\mathbf{0},\mathbf{K}(\mathbf{X},\mathbf{X})+\sigma_n^2\mathbf{I}\right)$, where $\mathbf{I}$ is the identity matrix and $\mathbf{K}(\mathbf{X},\mathbf{X})$ is a covariance matrix. Here, the zero-mean prior is adopted, since the change of actuation $\Delta\boldsymbol{\alpha}$ should have zero mean. The i^{th}-row, and j^{th}-column element $k_{ij}=k(\mathbf{x}_i,\mathbf{x}_j)$ in covariance matrix $\mathbf{K}(\mathbf{X},\mathbf{X})$ is a customized function. Here, squared-exponential kernel function [29] is used:

$$k_{ij}=k(\mathbf{x}_i,\mathbf{x}_j)=\sigma_s^2\exp\left(-0.5(\mathbf{x}_i-\mathbf{x}_j)^T\boldsymbol{\Lambda}(\mathbf{x}_i-\mathbf{x}_j)\right) \tag{8}$$

where σ_s^2 is the signal variance, and $\boldsymbol{\Lambda}=\mathrm{diag}(\boldsymbol{\lambda})$ is a diagonal matrix with characteristic length-scales $\boldsymbol{\lambda}=[\lambda_1,\ldots,\lambda_n]^T$ acting on each dimension of the input $\mathbf{X}$ individually. Hyperparameters σ_s, σ_n and $\boldsymbol{\lambda}$ in this regression can be determined by maximizing the negative log marginal likelihood. The hyperparameters can be found by standard optimization methods such as conjugate gradient, which is an automatically seeking procedure without the need of heuristic intervention. With these, GPR can generate a global nonlinear mapping model ready for prediction.

Prediction: Given a query input set $\hat{\mathbf{x}}$, the joint distribution of the observed target values $\mathbf{y}^s$ and predicted value $g(\hat{\mathbf{x}})$ are expressed as [29]:

$$\begin{bmatrix}\mathbf{y}^s\\ g(\hat{\mathbf{x}})\end{bmatrix}\sim\mathrm{N}\left(\mathbf{0},\begin{bmatrix}\mathbf{K}(\mathbf{X},\mathbf{X})+\sigma^2\mathbf{I} & \mathbf{k}(\mathbf{X},\hat{\mathbf{x}})\\ \mathbf{k}(\hat{\mathbf{x}},\mathbf{X}) & k(\hat{\mathbf{x}},\hat{\mathbf{x}})\end{bmatrix}\right) \tag{9}$$

The predicted mean $\bar{g}(\hat{\mathbf{x}})$ and covariance $V(\hat{\mathbf{x}})$ can be obtained by conditioning the above joint distribution:

$$\bar{g}(\hat{\mathbf{x}})=\mathbf{k}^T(\mathbf{X},\hat{\mathbf{x}})\left(\mathbf{K}(\mathbf{X},\mathbf{X})+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}^s=\mathbf{k}^T(\mathbf{X},\hat{\mathbf{x}})\boldsymbol{\beta} \tag{10}$$

$$V(\hat{\mathbf{x}})=k(\hat{\mathbf{x}},\hat{\mathbf{x}})-\mathbf{k}^T(\mathbf{X},\hat{\mathbf{x}})\left(\mathbf{K}(\mathbf{X},\mathbf{X})+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{k}(\mathbf{X},\hat{\mathbf{x}}) \tag{11}$$

where $\boldsymbol{\beta}$ denotes the prediction vector.

2) Localized model

The most time-consuming operation in GPR is the inversion of matrix $\left(\mathbf{K}(\mathbf{X},\mathbf{X})+\sigma_n^2\mathbf{I}\right)$ with complexity of $O(N^3)$. To improve the computational efficiency for robot control, reducing the dimension of input matrix is an effective choice. Therefore, we partition the training data distributed in the whole workspace into M clusters, $D_j, j=1,2,\ldots,M$, using the *k*-means clustering algorithm, where the Euclidean distance is replaced by Gaussian kernel-based similarity measure as in **Eq. (8)**. Each observation is assigned to the cluster that the similarity between the observation and cluster center reaches maximum. The actuator input $\boldsymbol{\alpha}$ performs as the clustering basis, since it could reflect the robot state. The center of j^{th} cluster could be represented as $\mathbf{c}_j\in U^m$. Every cluster of training data containing N_j samples generates a local model Φ_j. Moreover, a maximum model size could be predefined as $N_j^{\max}$ for each cluster, thus simplifying the calculation. Each local model Φ_j will generate a relevant prediction $\hat{\mathbf{y}}_j$ by **(10)** at each step. The actuator input for the next step is determined upon the weighted average of M local GPR predictions [30]:

$$\Delta\boldsymbol{\alpha}(k)=\sum_{j=1}^{M}\omega_j\hat{\mathbf{y}}_j\Big/\sum_{j=1}^{M}\omega_j,\ \ j=1,2,\ldots,M \tag{12}$$

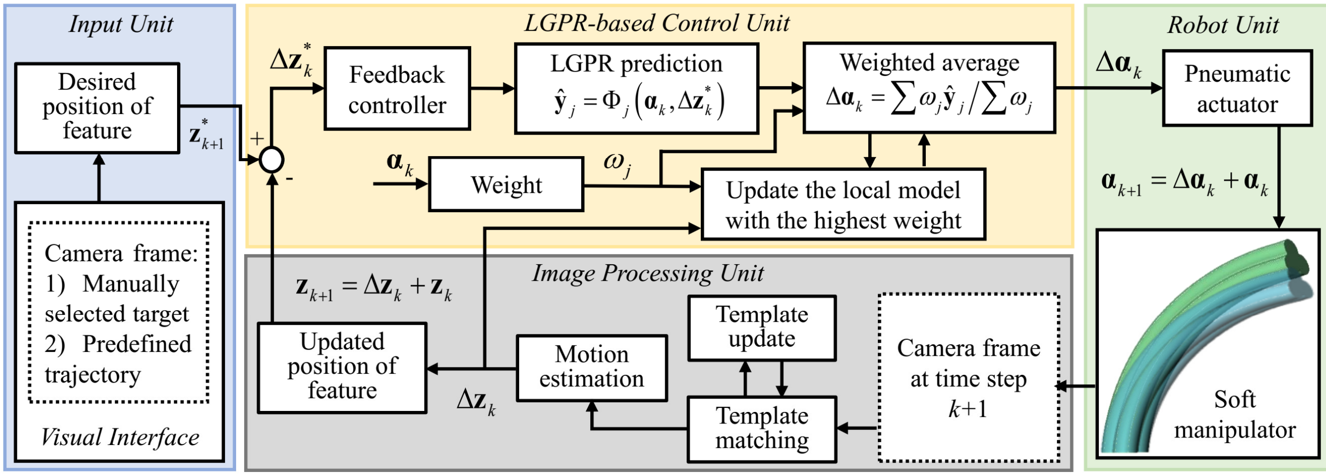


Fig. 2. Proposed learning-based control architecture. Parameters $\boldsymbol{\alpha}$ and $\mathbf{z}$ denote, respectively, the actuation command and the position of captured image features. The input unit provides the positional command in image domain, where the target position $\mathbf{z}^*_{k+1}$ can be selected manually or predefined by a reference trajectory. The local-GPR-based control unit generates the actuation command $\Delta\boldsymbol{\alpha}_k$, referring to the desired displacement $\Delta\mathbf{z}^*_k$ and current state $\boldsymbol{\alpha}_k$. The image processing unit esitmates the real-time displacement $\Delta\mathbf{z}_k$ for the online update of local GPR models and the feedback control.

where ω_j quantifies the similarity between current actuator input $\boldsymbol{\alpha}(k)$ and the j^{th} cluster center. Here, a Gaussian kernel is employed to measure this similarity.

The procedures of clustering training data and initialization are summarized in **Algorithm 1**.

Algorithm 1: Initialization of the GPR Model with Clustering

Input: $\mathbf{X}$ (inputs), $\mathbf{Y}$ (observation),
$\omega(\cdot)$ (similarity kernel function)
Partition the inputs samples into M clusters using *k*-means clustering.
for each cluster $j = 1, 2\ldots, M$ **do**
    Train j^{th} Local GPR model **(8)**
end for

Algorithm 2: Online Update of Local GPR Models

for each new data point $(\mathbf{x}_i, \mathbf{y}_i)$
    for each local GPR model Φ_j, $j = 1, 2\ldots, M$ **do**
        Compute similarity between the model center and input
            $\omega_j = \omega(\mathbf{x}_i, \mathbf{c}_j)$
    end for
    Choose the closest model:
        $r = \arg\max_j \omega_j$
    if $\omega_r >$ similarity threshold **then**
        if $N_r > N_r^{\max}$
            Delete the oldest point in model Φ_r
        end if
        Insert $(\mathbf{x}_i, \mathbf{y}_i)$ to the local model data set D_r:
            $\mathbf{X}_r = \mathbf{X}_r \cup \mathbf{x}_i$, $\mathbf{Y}_r = \mathbf{Y}_r \cup \mathbf{y}_i$
        Update model center:
            $\mathbf{c}_r = \mathrm{mean}(\mathbf{X}_r)$
        Update the Cholesky matrix and the prediction vector of local model **(13)**
    else
        Create a new local model
            $\mathbf{c}_{M+1} = \mathbf{x}_i, \mathbf{X}_{M+1} = \mathbf{x}_i, \mathbf{Y}_{M+1} = \mathbf{y}_i$, $M = M + 1$
        Initialize new Cholesky matrix and new prediction vector
    end if
end for

3) *Incremental learning*

Online update of the inverse model enables the controller to adapt with various changes of robot interactions and mechanical property, e.g. loading on the end-effector. Once the new actuation $\Delta\boldsymbol{\alpha}(k)$ is executed at each step, the corresponding actual motion vector $\Delta\mathbf{z}(k)$ could be obtained by the image processing unit. Thereby, a set of new sample data with input $\mathbf{x} = [\boldsymbol{\alpha}(k)^T, \Delta\mathbf{z}(k)^T]^T$ and output $\mathbf{y} = \Delta\boldsymbol{\alpha}(k)$ will be produced. This online sample could represent the latest working environment, and be added into the nearest cluster D_r, i.e., the one with maximal value of ω_j, for updating the corresponding local model Φ_r. The dataset update, including vector $\mathbf{Y}$ and input matrix $\mathbf{X}$, is straightforward. If the current model size $N_r \le N_r^{\max}$, the samples in cluster $D_r = \{[\mathbf{x}_i, \mathbf{y}_i]\}, i = 1, \ldots, N_r$ is retrained for a new inverse model Φ_r^{new}; otherwise $N_r > N_r^{\max}$, the oldest sample $[\mathbf{x}_1, \mathbf{y}_1]$ will be discarded and the cluster $D_r = \{[\mathbf{x}_i, \mathbf{y}_i]\}, i = 2, \ldots, N_r^{\max}$ is retrained. Under the size limitation, the point prediction can be kept fast and effective.

To update prediction vector $\boldsymbol{\beta}$, we could directly adjust the Cholesky decomposition $\mathbf{L}\mathbf{L}^T$ of matrix $\left(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2\mathbf{I}\right)$ presented in [30]. Then the prediction vector can be solved from $\mathbf{y} = \mathbf{L}\mathbf{L}^T\boldsymbol{\beta}$. A new point can be considered by adding a new row to the bottom of matrix $\mathbf{L}$:

$$\mathbf{L}_{new} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{l}^T & l_* \end{bmatrix} \tag{13}$$

$$\mathbf{L}\mathbf{l} = \mathbf{k}(\mathbf{X}, \mathbf{x}_{new}), \quad l_* = \sqrt{k(\mathbf{x}_{new}, \mathbf{x}_{new}) - \|\mathbf{l}\|^2} \tag{14}$$

Deleting the oldest data is achieved in two steps: Firstly, exchange the oldest data in $\mathbf{L}$ to the last row by multiplying permutation matrix $\mathbf{R} = \mathbf{I} - (\boldsymbol{\delta}_1 - \boldsymbol{\delta}_{Nr})(\boldsymbol{\delta}_1 - \boldsymbol{\delta}_{Nr})^T$, i.e. **RL**, where $\boldsymbol{\delta}_i$ is a zero vector whose i^{th} element is one; then $\mathbf{L}_{new}$ can be obtained by removing the last row of matrix **RL**.

The incremental learning procedures are summarized in Algorithm 2. The control block diagram in Fig. 2 shows the key processing components including the aforementioned local GPR prediction and motion estimation.

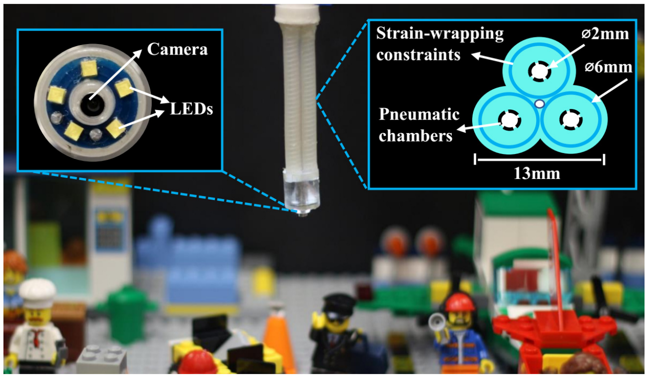


Fig. 3. Experimental setup in a scene of LEGO. The soft manipulator was made of silicone rubber, and which is driven by three fiber-constrained air chambers. An endoscopic camera and five LEDs are mounted at the tip.

III. Experiments and Results

A. Experiment setup

Experimental setup of our visual servo test is illustrated in **Fig. 3**. A soft manipulator is fixed downward, viewing the workspace scene built from LEGO. A hyper-elastic soft manipulator in small size (ø 13 mm × 67 mm) was fabricated from room-temperature-vulcanization (RTV) silicone (Ecoflex 0050; Smooth-On, Inc.), which is a relatively *low-stiffness* rubber [20]. Such a "floppy" robot comprises three cylindrical air chambers that can be inflated individually. A layer of helical Kevlar string with 1-mm pitch is wrapped around each chamber to restrict its radial expansion, giving rise to pure elongation/shortening of chambers upon inflation/deflation. This provides the effective bending motion with a maximum bending angle larger than 90°. Cooperation of the three chamber pressures allows the omni-directional servo of the camera (Depth of view: 8 to 150 mm) and LED illumination module mounted at the robot tip. With a 90° diagonal field of view, the camera captures images of 400 × 400 pixels, indicating that a pixel translates to 0.16° field of view. The inflation volume of each chamber is controlled precisely with a pneumatic cylinder actuated by a stepper motor.

The major challenge that hurdles precise control of the soft robot can be attributed to its nonlinear kinematic behavior. The twisting of the continuum structure will also cause rotation of the camera view in unexpected directions.

B. Pre-train of Local GPR Inverse Model

Before operation, the local GPR model is first initialized using the data $\mathbf{D}' = \{\boldsymbol{\alpha}_i, \Delta\mathbf{z}_i, \Delta\boldsymbol{\alpha}_i\}|_{i=1}^{N}$ collected within a calibration environment, in which the robot base is fixed. An EM tracking coil (NDI Aurora®) was attached on the tip of the soft robot to record the 3D position. A uniformly distributed actuation input set $\boldsymbol{\alpha}_i$ is used for exploration of the robot configuration space in this study. Camera image is captured at every new actuation input after the equilibrium is reached.

The incremental change of $\Delta\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_j - \boldsymbol{\alpha}_i$ is obtained from differencing of two robot configurations $\boldsymbol{\alpha}_j$ and $\boldsymbol{\alpha}_i$. To obtain $\Delta\mathbf{z}_i$, a visual feature appeared at the image center $\mathbf{z}_c$ corresponding to $\boldsymbol{\alpha}_i$ is first detected, $\Delta\mathbf{z}_i = \mathbf{z}_{new} - \mathbf{z}_c$ is then obtained by detecting the new feature position $\mathbf{z}_{new}$ in the image corresponding to $\mathbf{z}_j$.

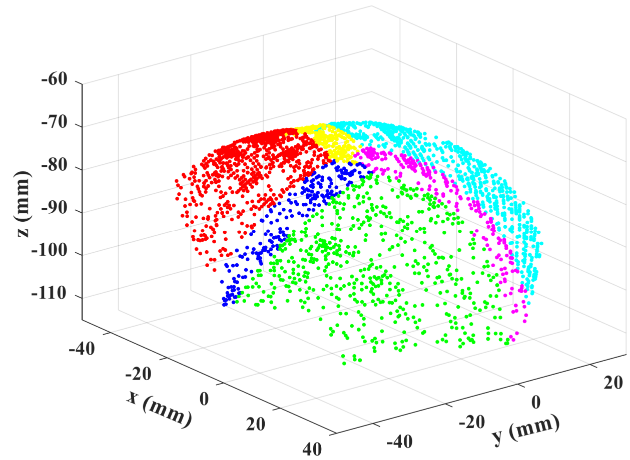


Fig. 4. Three thousand positions of robot tip sampled for pre-training of the inverse model. Using the k-means algorithm, all these training points were divided into six (colored) clusters based on their actuation inputs.

It is noteworthy that the collected data $\mathbf{D}'$ is generally reflecting the nonconvexity of the inverse mapping. This is because the actuation space ($\Delta\boldsymbol{\alpha}_i$) has a higher dimension than the task space ($\Delta\mathbf{z}_i$). There may exist more than one $\Delta\boldsymbol{\alpha}_i$ that corresponds to a data pair ($\boldsymbol{\alpha}_i, \Delta\mathbf{z}_i$), resulting in multi-valued mapping of $\Delta\boldsymbol{\alpha}_i = \Phi(\boldsymbol{\alpha}_i, \Delta\mathbf{z}_i^*)$. Learning such mapping directly from nonconvex $\mathbf{D}'$ is an ill-posed problem. Here we construct a nonredundant set of training data from $\mathbf{D}'$ by filtering out data pairs with all-zero elements in $\boldsymbol{\alpha}_i = [\alpha_{1i}, \alpha_{2i}, \alpha_{3i}]^T$:

$$\mathbf{D} = \{\boldsymbol{\alpha}_i, \Delta\mathbf{z}_i, \Delta\boldsymbol{\alpha}_i \mid \alpha_{1i} \cdot \alpha_{2i} \cdot \alpha_{3i} = 0, \forall i = 1, \ldots, N\} \quad (15)$$

The constraints in **(15)** ensures that at least one chamber among the three has zero pressure, thus every actuation command in **(15)** corresponds to a unique tip position in the workspace. It indicates the actuation space is non-redundant, resulting in a single-valued mapping. In practice, due to approximation error, the controller initialized from the convex set $\mathbf{D}$ would output $\boldsymbol{\alpha}_i$ that violates the convexity. The constraint **(15)** has also to be applied in order to maintain the convexity during the incremental update of the controller.

K-means clustering based on a Gaussian kernel is performed to partition the training data set $\mathbf{D}$ satisfying **(15)** into 6 clusters, as shown in **Fig. 4**. A data set of 1000 samples are randomly selected for pre-training of local GPR models. The data size of 6 clusters are [202, 257, 82, 294, 106, 59] respectively, with the upper limit of data size set to 300 for each local model.

C. Experiments, Results and Discussion

Four manipulation tasks are conducted to evaluate the performance of proposed controller under various conditions, such as target tracking under external interactions and path following with changing load. The tracking error is defined as the shortest distance between the current position of target and the desired trajectory.

1) Point-to-point tracking

In this task, the robot has to aim the camera center at a series of target way-points in the image view (Fig. 5). Five target points (labels 1 to 5) are manually selected in series from the 400×400 image plane. Around the target points, a

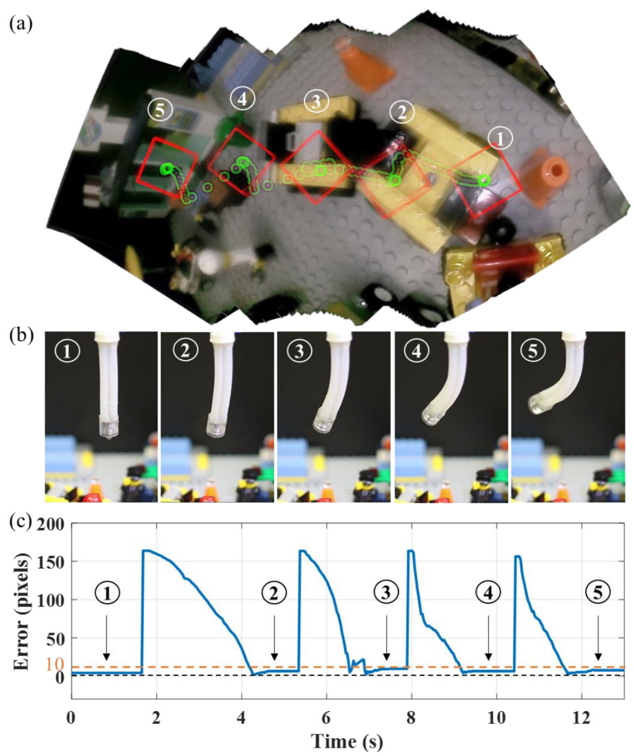


Fig. 5. Five targets manually selected on the image plane for robot tracking. **(a)** Mosaic image obtained during multiple target tracking. A template pattern (in red block) was form and centered at the target point selected. A series of green bubbles represent the instantaneous camera centers, showing the trajectory travelled along the matched template patterns. **(b)** Corresponding robot configurations while tracing targets successively at its image plane center; **(c)** Absolute tracking errors throughout the journey.

100×100 template pattern centered is created and denoted by a red box in image frames. **Fig. 5a** shows the panorama image obtained by mosaicking the camera frames during the tracking task. The image center at each step was marked by a green circle. **Fig. 5b** presents the robot configurations when aiming at each target way-point. We set a 10-pixel tolerance for the accuracy of target matching. **Fig. 5c** provides the tracking error in unit of pixel, showing that the proposed controller can achieve precise targeting at each target way-point with an error less than 10 pixels. It demonstrates that the inverse mapping approximated by the local GPR model can accurately compensate the disorientation between input space and task space.

2) *Target tracking under external force*

This section presents a tracking experiment with an aim to evaluate the feedback control performance in response to external disturbance. The robot is commanded to align its image center to a fixed target, while an unknown force pushes it away from its original configuration. The controller only utilizes image as feedback. The robot configurations and target tracking errors are depicted in **Fig. 6**. It shows that our robot could fixate the desired target within an error <20 pixels even under an unknown external force. Feedback control achieves this enhanced accuracy by reducing the effective compliance of the robot, which may raise concerns about loss of adaptiveness to the environment. However, the force output of the system is still inherently limited by the low stiffness of the robot body.

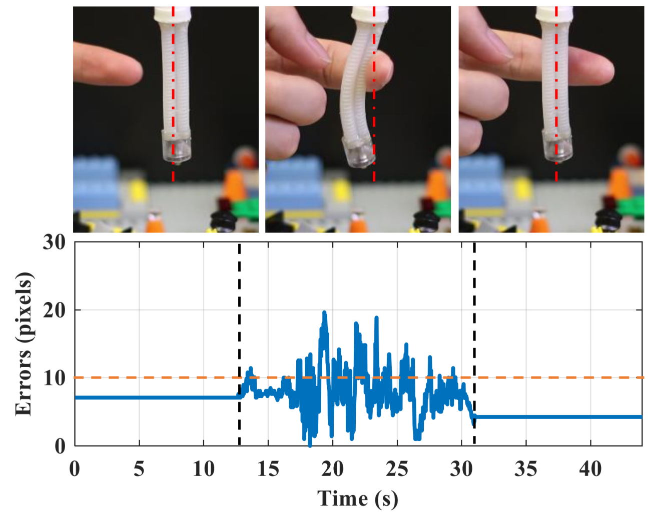


Fig. 6. Target tracking experiment involving external disturbance. The upper pictures (from left to right) show three robot configurations before, during and after the external force application. The correspoding tracking errors are shown below. The 10-pixel tolerance level is marked (dash lines in orange).

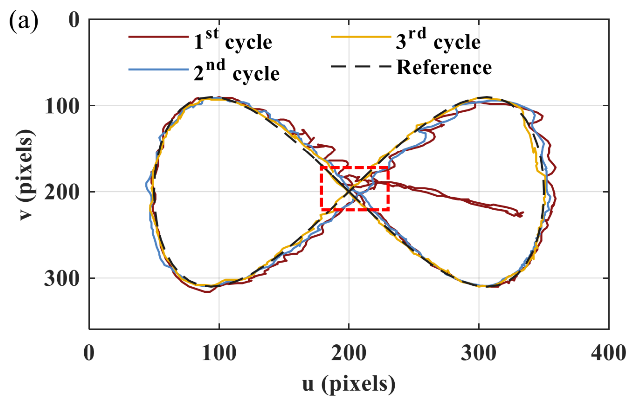
3) *Path following with a scarce pre-trained model*

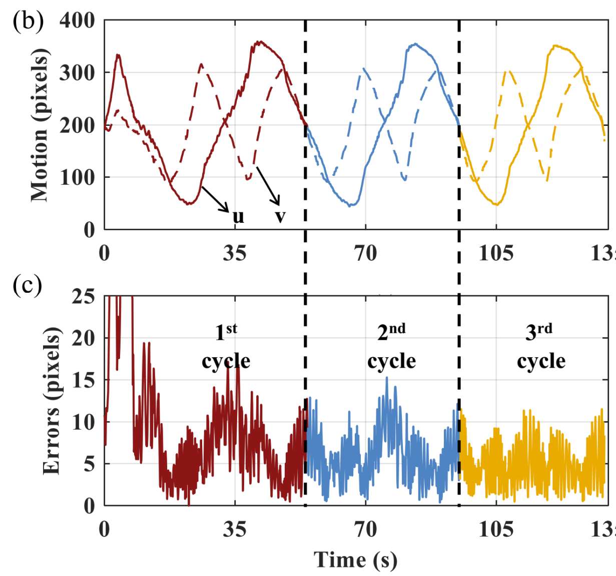
The effect of online learning control is investigated via a path following task. The reference path was defined by a series of desired template center positions inside the 400×400px camera frame. A purposely badly trained local GPR model is used at the start. Only a scarce dataset of 300 samples are presented to initialize the model, thus the controller must rely on the online collected data to refine the inverse mapping. The tracked trajectory and tracking errors are depicted in **Fig. 7**. In the first cycle, the robot can only roughly follow the reference path, with a root-mean-square error (RMSE) of 16.8 pixels and maximum of 64.5 pixels (**Fig. 7d**). Then the controller begins to converge, keeping track of the reference path in the second and third cycles (**Fig. 7a**). The tracking error finally improved to an RMSE of 5.4 pixels and maximum of 11.5 pixels. **Fig. 7b** and **c** illustrate the u and v coordinates of image plane and the tracking error in pixels. The robot could follow the trajectory with a maximum error of 12 pixels after two cycles. This experiment demonstrates that our learning-based controller can achieve high accuracy path following through efficient online refinement of the inverse mapping, despite being initialized with a poorly trained model.

4) *Path following under varying load*

The final experiment aims to study the online learning performance under varying tip load. In this regard, the local GPR model was first pre-trained at a vertical robot pose without any attachment. Then, an inflatable water balloon is wrapped around the robot tip, acting as a variable payload (**Fig. 8d**). Up to 15g of water can be inflated via a silicone tube. In addition to the 6g of balloon setup, it altogether corresponds to 105% of the robot original weight (20g). The robot is also realigned to a horizontal pose after pre-training, meaning that the online learning controller is required to handle new tip load and gravity condition.

The tracking trajectory and errors in the three successive cycles are plotted in **Fig. 8a** and **Fig. 8b** respectively. In the first cycle, the controller quickly adapts to the additional 6g of balloon setup and the change in gravity direction. Despite the





(d)

Cycle No.	1	2	3
RMSE (pixels)	16.8	6.6	5.4
Max. error (pixels)	64.5	15.3	11.5

Fig. 7. Performance of tracking on a predefined "∞" trajectory. **(a)** Robot tracing the reference trajectory in its endoscopic view using our pre-trained local GPR model. The red-dashed block indicates the initial view point centered at the intersection of the "∞" trajectory**.** A large deviation from the reference trajectory is observed at the beginning of the 1st cycle (red). It converges to the reference (dashed) rapidly long before the further two cycles in blue and yellow, once having the sufficient model update. **(b)** Motions depicted in two separated coordinates, *u* and *v*. **(c)** Corresponding tracking errors throughout the 135-second journey. **(d)** Tracking performances in general.

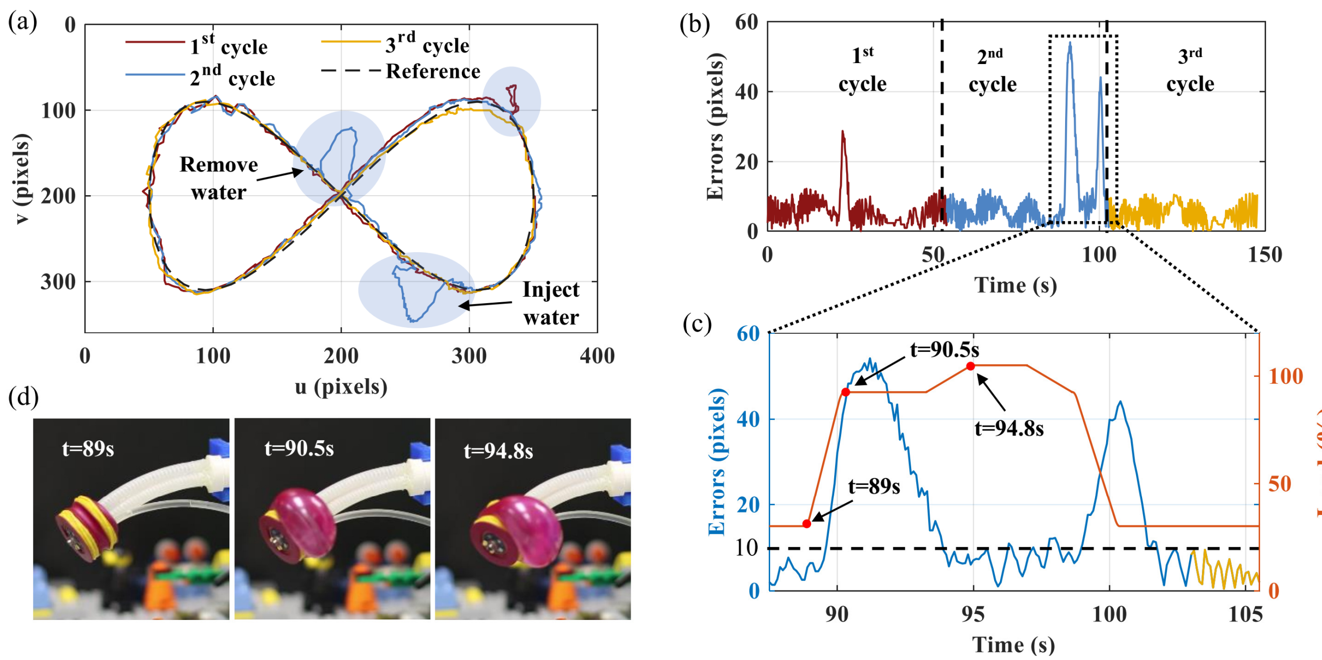


Fig. 8. Tracking performance under variable tip loading. A (6-gram) balloon cap pumped with water in-and-out was mounted at the robot tip, introducing a variable load ranging from 6 to 21g. **(a)** Tracked trajectory in three cycles. Three substantial deviations are observed, corresponding to the errors of pre-training (1st cycle in red), injecting and removing water (2nd cycle in blue); **(b)** Corresponding tracking error in time domain; **(c)** Tracking error *vs* variable load varied below 21g. The load is represented as % relative to the mass of robot itself, 20g; **(d)** Robot configuration and its balloon shape at three-time steps.

error peak at a t=22s, it is effectively compensated and eliminated in the following cycles.

In the second cycle, the water balloon is inflated at t=89s to impose a fast additional payload of 12.5g or 62.5% of the robot mass (dynamic loading), resulting in an error peak in 1 second (**Fig. 8c**). The configurations of robot with the inflated balloon at t=90.5s is shown in **Fig. 8d** (middle). The error is quickly diminished by the online learning controller at t=90.5~93.5s. The water balloon is further injected, increasing the tip load to a maximum of 15g at t=94.8s (**Fig. 8d**, right). During this period where loading slowly changed (quasi-static loading), a small tracking error is maintained (<12 pixels), which implies that the online learning controller can adapt to the additional tip load. Finally, the water balloon is deflated to empty at t=98s. The tracking error rises again due to the sudden change in payload, but then quickly converges to a small level within 1 second. In the third cycle, the tracking error maintained in a range less than 12 pixels. The results demonstrate that our controller could handle the varying load, keeping track of a path with an error smaller than 12 pixels. This implies that the image motion feedback and actuation data can effectively update the kinematic models online.

IV. CONCLUSION

This paper presents a nonparametric online learning control framework, which enables eye-in-hand visual servo of a fluid-driven soft robot with very low stiffness. By estimating the inverse mapping solely from measured data, our controller alleviates the need for a kinematics model or camera calibration, that may be challenging to acquire for soft manipulators. The local weighted learning scheme supports efficient online update of the inverse mapping, thus enabling precise robot manipulation even under external interactions such as changing payload. Integrating image feedback with nonparametric learning can bring new opportunities to minimally invasive surgical applications, such as soft robotic endoscopy and laparoscopy [31, 32], as they can take advantage of existing camera feedback.

Our work first demonstrates vision-based path following for a hyper-elastic robot with heavy variable loading (up to 105% of the robot weight). In the cycle with addition or removal of the payload, it maintained an acceptable accuracy (RMSE 14.4 pixels and maximum 54.1 pixels). Even under a fast and heavy dynamic load (~62.5% of robot mass over 1.5s), the controller could maintain stability. When the water load slowly changed in the period between 93.5 to 98 s (quasi-static loading), the controller could effectively reduce the error, in contrast to the initial dynamic loading. After removing the load, the tracking error converged to an even lower value (RMSE 5.6 pixels and maximum 10.8 pixels). In future work, we will investigate the use of known robot or camera information in our algorithm to improve control performance. For example, Lee et. al [20] utilized FEM of their soft manipulator to initialize a learning-based model, reducing the need for time consuming random exploration. We also intend to extend the online learning controller to more dynamic tasks. Visual servo control of highly redundant robots will also be of our interest. This would enable more versatile manipulation in a confined space.

REFERENCES

[1] B. Mosadegh, *et al*., "Pneumatic Networks for Soft Robotics that Actuate Rapidly," *Adv. Funct. Mater.,* vol. 24, no. 15, pp. 2163-2170, 2014.

[2] S. Kim, C. Laschi, and B. Trimmer, "Soft robotics: a bioinspired evolution in robotics," *Trends Biotechnol.,* vol. 31, no. 5, pp. 287-94, May 2013.

[3] C. Laschi and M. Cianchetti, "Soft robotics: new perspectives for robot bodyware and control," *Front. Bioeng. Biotechnol.,* vol. 2, p. 3, 2014.

[4] J. Shintake, V. Cacucciolo, D. Floreano, and H. Shea, "Soft Robotic Grippers," *Adv Mater.,* vol. 30, May 7 2018, Art. no. e1707035.

[5] H. In, B. B. Kang, M. Sin, and K.-J. Cho, "Exo-Glove: A Wearable Robot for the Hand with a Soft Tendon Routing System," *IEEE Robot. Autom. Mag.,* vol. 22, no. 1, pp. 97-105, Mar. 2015.

[6] T. Ranzani, G. Gerboni, M. Cianchetti, and A. Menciassi, "A bioinspired soft manipulator for minimally invasive surgery," *Bioinspiration Biomimetics,* vol. 10, no. 3, May 13, 2015, Art. no. 035008.

[7] T. G. Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control Strategies for Soft Robotic Manipulators: A Survey," *Soft Robot.,* vol. 5, pp. 149-163, Jan. 3 2018.

[8] D. B. Camarillo, C. F. Milne, C. R. Carlson, M. R. Zinn, and J. K. Salisbury, "Mechanics Modeling of Tendon-Driven Continuum Manipulators," *IEEE Trans. Robot.,* vol. 24, no. 6, pp. 1262-1273, Dec. 2008.

[9] R. J. Webster and B. A. Jones, "Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review," *Int. J. Robot. Res.,* vol. 29, no. 13, pp. 1661-1683, 2010.

[10] R. S. Penning, J. Jung, N. J. Ferrier, and M. R. Zinn, "An evaluation of closed-loop control options for continuum manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.,* 2012, pp. 5392-5397.

[11] M. Luo, *et al*., "Slithering towards autonomy: a self-contained soft robotic snake platform with integrated curvature sensing," *Bioinspiration Biomimetics,* vol. 10, no. 5, Sep. 3, 2015, Art. no. 055001.

[12] E. L. White, J. C. Case, and R. Kramer-Bottiglio, "A Soft Parallel Kinematic Mechanism," *Soft Robot.,* vol. 5, no. 1, pp. 36-53, Feb. 2018.

[13] D. C. Rucker and R. J. Webster, "Mechanics of continuum robots with external loading and general tendon routing," in *Experimental Robotics*, New York, NY, USA: Springer, 2014, pp. 645-654.

[14] J. Till, C. E. Bryson, S. Chung, A. Orekhov, and D. C. Rucker, "Efficient computation of multiple coupled Cosserat rod models for real-time simulation and control of parallel continuum manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 5067-5074.

[15] K.-H. Lee, *et al.*, "FEM-based soft robotic control framework for intracavitary navigation," in *Proc. IEEE Int. Conf. Real-Time Comput. Robot*., 2017, pp. 11-16.

[16] F. Largilliere, *et al*., "Real-time control of soft-robots using asynchronous finite element modeling," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, p. 2550-2555.

[17] M. Giorelli, *et al*., "Neural Network and Jacobian Method for Solving the Inverse Statics of a Cable-Driven Soft Arm With Nonconstant Curvature," *IEEE Trans. Robot.,* vol. 31, no. 4, pp. 823-834, Aug. 2015.

[18] T. G. Thuruthel, *et al*., "Learning closed loop kinematic controllers for continuum manipulators in unstructured environments," *Soft robot.,* vol. 4, no. 3, pp. 285-296, 2017.

[19] M. C. Yip and D. B. Camarillo, "Model-Less Feedback Control of Continuum Manipulators in Constrained Environments," *IEEE Trans. Robot.,* vol. 30, no. 4, pp. 880-889, Aug. 2014.

[20] K.-H. Lee, *et al.*, "Nonparametric Online Learning Control for Soft Continuum Robot: An Enabling Technique for Effective Endoscopic Navigation," *Soft Robot.,* vol. 4, pp. 324-337, 2017.

[21] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Computed torque control with nonparametric regression models," in *Proc. IEEE Amer. Control Conf.*, 2008, pp. 212-217.

[22] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robot. Autom. Mag.,* vol. 13, no. 4, pp. 82-90, Dec. 2006.

[23] Y. H. Liu, H. Wang, C.Y. Wang, K. K. Lam, "Uncalibrated visual servoing of robots using a depth-independent interaction matrix," *IEEE Trans. Robot.,* vol. 22, no. 4, pp. 804-817, Aug. 2006.

[24] H. Wang, *et al*, "Visual servo control of cable-driven soft robotic manipulator," in *Proc. IEEE Int. Conf. Intell. Robots Syst*., 2013, pp. 57-62.

[25] Y. Lu, C. Zhang, S. Song, and M. Q.-H. Meng, "Precise motion control of concentric-tube robot based on visual servoing," in *Proc. IEEE Int. Conf. Inf. Autom*., 2017, pp. 299-304.

[26] J. D. Greer, T. K. Morimoto, A. M. Okamura, and E. W. Hawkes, "Series pneumatic artificial muscles (sPAMs) and application to a soft continuum robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 5503-5510.

[27] H. Wang, *et al*., "Visual Servoing of Soft Robot Manipulator in Constrained Environments With an Adaptive Controller," *IEEE/ASME Trans. Mechatronics,* vol. 22, no. 1, pp. 41-50, Feb. 2017.

[28] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2008.

[29] C. E. Rasmussen and C. K. Williams, Gaussian processes for machine learning, vol. 38. Cambrige, MA, USA: MIT Press, 2006, pp. 715-719.

[30] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local gaussian process regression," *Adv. Robot.,* vol. 23, no. 15, pp. 2015-2034, 2009.

[31] H. Abidi, *et al.*, "Highly dexterous 2-module soft robot for intra-organ navigation in minimally invasive surgery," *Int. J. Med. Robot.,* vol. 14, no. 1, Feb. 2018.

[32] F. Cosentino, E. Tumino, G. R. Passoni, E. Morandi, and A. Capria, "Functional evaluation of the endotics system, a new disposable self-propelled robotic colonoscope: in vitro tests and clinical trial," *Int. J. Artif. Organs,* vol. 32, no. 8, pp. 517-527, 2009.