Fem-based trajectory tracking control of a soft trunk robot

Ke Wu, Gang Zheng, Junfeng Zhang

FEM-based Trajectory Tracking Control of a Soft Trunk Robot*

Ke Wu[a], Gang Zheng[a,*] and Junfeng Zhang[b]

[a]Université de Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL, Lille, F-59000, France
[b]School of Automation, Hangzhou Dianzi University, Hangzhou 310018, China

ARTICLE INFO

Keywords:
Soft Robotics
Finite Element Method
Trajectory Planning
Trajectory Tracking Control

ABSTRACT

As a novel class of robots, soft robots have demonstrated many desirable mechanical properties than traditional rigid robots due to their nature of being compliant, flexible and hyper-redundant, such as great adaptability to unknown environments, safe human robot interaction (HRI), energy-saving actuation and the maneuverability to display diverse mechanical properties. However, its inherent high-DoF nature would result in some complex nonlinear behaviors, and their kinematic or dynamic models are therefore harder to deduce than the ones of conventional rigid robots. In this paper, we propose a trajectory tracking control strategy for a soft trunk robot based on Finite Element Method (FEM). We first plan a feasible trajectory for the studied robot in SOFA (a FEM-based simulator) by solving a model-prediction-control (MPC)-based optimization problem. The second step is to conduct linearization around the pre-designed trajectory, based on which an associated controller can be then developed. The detailed derivation of the mentioned work is explained accordingly. In the end, the results of experimental validation is presented to prove the feasibility of the proposed method.

1. Introduction

As a novel class of robots, soft robots have demonstrated many desirable mechanical properties than traditional rigid robots [1] due to their nature of being compliant, flexible and hyper-redundant. For instance, these robots provide great adaptability to complex environments [1], safe human robot interaction (HRI) [2], energy-saving actuation [3], the maneuverability to display diverse mechanical properties [1][2], a large power-to-weight ratio [4][5] and increased fault tolerance [6][7]. Traditional rigid-body robots which consist of rigid links connected at kinematic joints, serve as mechanical devices to transfer motion, force and energy by the synthesized motion of rigid members whereas soft robots are designed to function similarly just through the elastic deformation of their flexible bodies. Due to the redundant degrees of freedom (DoF), these soft robots are more maneuverable than conventional rigid robots in terms of achieving some complex tasks that need unique and more compliant configurations. For example, this type of robots can be well-used in rescue missions [8], pipeline flaw detection [9][10][11], object grasping [12], medical surgery [13], space tasks [14] and so on.

On the other hand, the advantageous characteristics of these robots also lead to difficulties in deriving the kinematic and dynamic models of these manipulators [15]. Obviously, the control theory developed for rigid robots would hardly be feasible for soft robots, and the research on controlling those robots therefore remains huge possibility for exploration.

In the state-of-art literature, many research efforts for modeling and controlling continuum and soft robots (manipulators) have been reported. The contributors of [15] took the lead to propose an effective geometric modeling method called Constant-Curvature-assumed model (CC) for continuum manipulators. Essentially speaking, this method masterly utilizes the nature of Denavit-Hartenberg method and modifies it into an effective method that's able to map the relationship between these robots' configuration space and operational space. Due to this merit, CC model once became a popular choice of designing static controllers for continuum robots [15][16]. However, in some cases CC assumption becomes less accurate for the whole slender body of a continuum robot. Then, Piecewise Constant Curvature model (see details in [17]) was proposed to take over the modeling problem. PCC basically assumes the continuum robot can be divided into several independent segments where the bending of each segment is characterized by a circular arc of constant length (CC model applied in each segment). This simplification has made PCC one of the most popular and effective methods for modeling continuum manipulators [17]. For example, it is not only used in controlling soft robots that is main scope of this paper: kinematic control [16], feedforward dynamic control

ORCID(s):

[18], feedback dynamic control [19][20], configuration control [21], trajectory tracking or path following [22], but also applied to the cases of mechanical design [23], sensing [24] and workspace study [25]. Cosserat theory was also employed as a feasible option to deduce the kinematics of the trunk-like soft robots [26][27][28], and it's also extended to capture the dynamics of continuum robots [29][30]. Some representative cases of Cosserat-model-based controllers have been reported: robust sliding mode dynamic control [30], dynamic configuration tracking [31], synthesized control for parallel continuum manipulators [32] and so on. For some slender continuum compliant robots, Euler-Bernoulli beam theory can be used to develop the kinematics [33] and dynamics [34] of the robots, and their static and dynamic controllers can be therefore developed accordingly. Recently, Finite Element Method (FEM) has gradually become a promising option of modeling soft or continuum robots[35] but it's rather computationally expensive due to its high-dimensional nature. Because of this fact, some order reduction methods have been developed to deduce its computational cost [36]. However, compared to the mentioned methods, FEM does not need the studied deformable object to have a certain regular geometry (such as rod-like or trunk-like ones) since irregular geometries can be discretized into a group of fine mesh elements for further dynamic analysis using Euler-Lagrange method [37]. This nature has made FEM quite a universal strategy to model deformable robots of any arbitrary shape [35].

Likewise, FE model has been proved to be option to develop controllers for soft robots: [38] introduced a robust control strategy using neural networks; a static observer-based control method was proposed in [39]; [40] proposed a way of modeling the deformation of soft robots and also checking its controllability; a feedback quasi-static control strategy for soft robots has been realized by applying FEM and the visual tracking technique in [41]. In this paper, we also choose FEM as the modeling tool, and then promote our previous work [42] where a FEM-based gain-scheduling control strategy for a soft trunk robot. This former contribution focuses on global path following by smoothly switching the gains in each sub-workspace without considering time constraint. However, in this paper, we propose an effective dynamic controller that can realize time-varying and global trajectory-tracking for the same robot. Precisely, we first plan a feasible trajectory for the studied robot in SOFA (a FEM-based simulator) by solving a MPC-based optimization problem. Then, we conduct linearization of this robotic system around the pre-designed trajectory, and then an associated controller can be developed. We validate its feasibility and effectiveness by doing a comparison test between a common PID controller and this proposed one, and afterwards its robustness against temporary and permanent disturbances is proved by experimental testing as well.

2. Problem statement

2.1. Description of the studied soft trunk-like robot

In this paper, we still focus on the same robot studied in [42] (see Fig. 1) to demonstrate the promotion on the control strategy. We fabricated the prototype with a long uniform hole space from its based to its tip where the position sensor can be properly settled at its tip. Four inextensible plastic cables, which are functioned by 4 steppers and mounted through the robot's body, are designed to drive the robot moving but the soft rubber can easily wear if cables repeatedly travels through the rubber. To reduce this undesired friction, we 3D-printed 15 rigid rings to be mounted along the robot's body (see Fig. 1) where the sliding friction happens between the rigid rings and cables instead. The mass of trunk robot $m = 0.045$ kg, Young's modulus $E = 6.5 \times 10^6$ Pa and Poisson ratio $\nu = 0.45$.

2.2. Control objective

The objective of this paper is to control the end-effector of the trunk-like robot to track a planned trajectory with or without the interference of external disturbances. As stated above, this work is regarded as an extension of our previous work [42] to realize global trajectory tracking tasks in its feasible workspace.

3. Modeling and trajectory generation

3.1. Modeling via Finite Element Method

Soft robots are strenuous to model owing to its high-DoF nature and complex deformation of materials [35]. In this paper, FEM is employed to obtain the dynamic model of the soft trunk manipulator on the basis of SOFA Framework, a FEM-based simulator. Therefore, the associated controller for trajectory tracking can be logically developed via synthesizing the information of the FE model. Concerning the derivation of FE dynamical model, the details can be found in [37] and [43] where the basic idea is similar to the classical one: we first choose the type and the number of fine mesh elements (displayed in Fig. 2) to discretize the geometry of the modeled deformable object (which in this

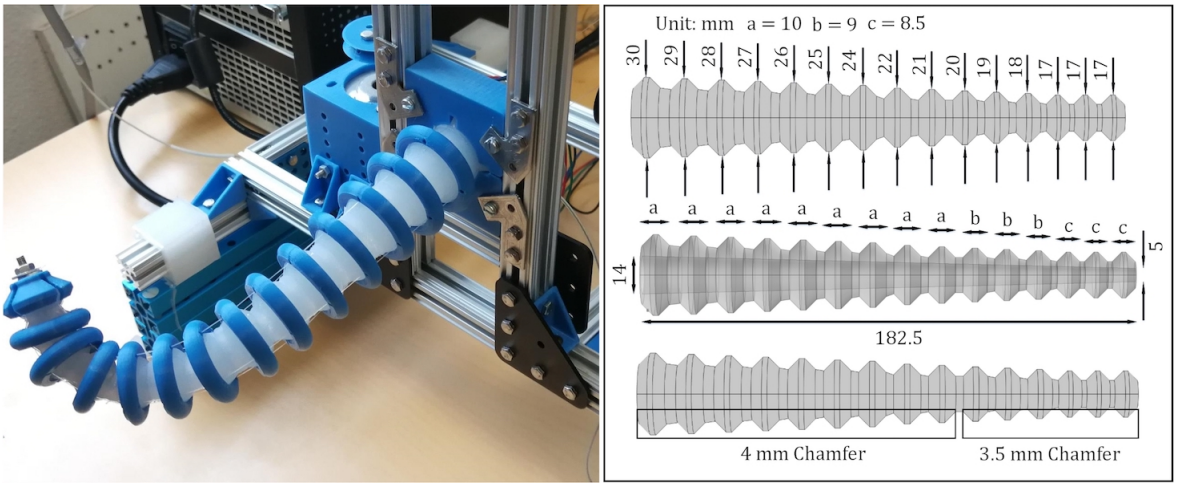


Figure 1: The studied trunk robot [42]

case is the studied robot); secondly, we calculate the displacement of any point inside the element via the displacement of its nodes through shape functions, and the corresponding strain and stress (might be linear or nonlinear according to the applied constitutive law); thirdly, we use either Euler-Lagrange equation or virtual work principal to derive the FE dynamical model for the studied manipulator. In this paper, we focus on the displacement control of the robot's end-effector ($q_e \in \mathbb{R}^{3\times1}$), and q_e is defined with respect to the inertial frame $F_I(q_e)$ as shown in Fig. 3. Therefore, this control problem is essentially an output regulation problem of the following nonlinear system based on the derived FE model:

$$\begin{aligned} &M(q)\ddot{q} + D(\dot{q}, q)\dot{q} + K(q)q = H(q)^T u \\ &q_e = Cq \end{aligned} \tag{1}$$

where $M(q) \in \mathbb{R}^{3n\times3n}$ denotes mass matrix; $D(\dot{q}, q) \in \mathbb{R}^{3n\times3n}$ denotes damping matrix; $K(q) \in \mathbb{R}^{3n\times3n}$ denotes stiffness matrix; $H(q) \in \mathbb{R}^{4\times3n}$ denotes the input control matrix; $u \in \mathbb{R}^{4\times1}$ denotes the tension forces exerted at the four cables; the friction between the cables and the rubber is ignored in our FE model so $H(q)$ only distributes u to all the elements containing the cable nodes where the cables travel through within the rubber (red nodes displayed in Fig. 2); $q \in \mathbb{R}^{3n\times1}$ denotes the displacement of all the n nodes (supposing q_i is the ith element of q, q_i refers to the displacement of the ith node with respect to its inertial frame $F_I(q_i)$ as shown in Fig. 3); logically, $\dot{q} \in \mathbb{R}^{3n\times1}$ and $\ddot{q} \in \mathbb{R}^{3n\times1}$ denote the corresponding velocity and acceleration respectively; the function of matrix $C \in \mathbb{R}^{3\times3n}$ is to pick out the end-effector displacement from all n ($n = 1484$) nodes defined in our derived FE model; the units of q and u are centimeter (cm) and Newton (N) respectively, and the rest units can be deduced accordingly.

In terms of the mesh type, tetrahedron elements (see Fig. 2) are chosen as it is compatible with complex geometry, and the robot's body is spatially discretized into $n = 1484$ nodes. The size of the mesh elements has been carefully chosen in order to deal with the trade-off between accuracy and computation expense.

3.2. Trajectory generation via SOFA

With the deduced dynamical model (1) of the investigated soft trunk robot, this subsection discusses how to generate a feasible end-effector trajectory via the FEM-based simulator (SOFA). Note that FE simulations are conducted in an off-line manner in this paper to complete two tasks (see details in Step 1 and Step 2 respectively in this chapter): numerically defining the workspace of the robot's end effector; numerically generating planned trajectory based on a MPC-based optimization structure. Because these two steps are both done in an off-line manner, the computational expense would not be a critical issue for this methodology. To treat this problem, let us firstly recall the definition of the end-effector workspace [43].

Definition 1. *Given a configuration of a soft manipulator dynamically modeled by (1) with a bounded actuator* $u \in$

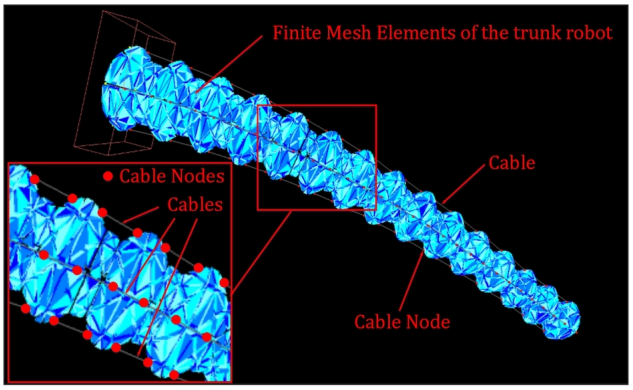


Figure 2: Mesh elements of the modeled trunk robot.

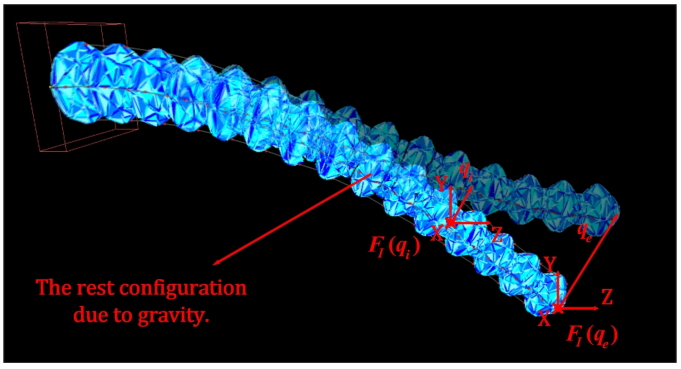


Figure 3: The inertial frames defined for the studied robot.

$\mathcal{U}$, the workspace $\mathcal{W}_E$ of the end-effector q_e, is a subspace of $\mathbb{R}^3$, defined below:

$$\mathcal{W}_E = \{\ q_e \in \mathbb{R}^3 \mid q_e = Cq, K(q)q - H(q)^T u = 0,\ \forall u \in \mathcal{U}\}$$

Remark 1. *The above definition of the workspace, which is similar to the one defined for a rigid manipulator, implies that $\mathcal{W}_E$ contains all possible equilibrium positions for which the end-effector q_e can reach and stay there.*

According to the above definition, given two positions in the end-effector's workspace, named as the initial position $q_e^{init} \in \mathcal{W}_E$ and the final position $q_e^{fin} \in \mathcal{W}_E$, the trajectory generation problem for such a soft trunk robot is to seek a feasible path starting from q_e^{init} and ending with q_e^{fin} over a time interval $[0, T]$, i.e., $q_e(0) = q_e^{init}$ and $q_e(T) = q_e^{fin}$, and at the same time satisfy the dynamical constraint (1).

In the literature, differential flatness [44] has been widely used to plan a feasible trajectory for different types of robots. Roughly speaking, the output of a nonlinear dynamical system is called to be flat if its state and input can be written as functions of this output and its derivatives. However, concerning the studied soft robot with dynamics (1), it is clear that the state q and the input u cannot be function of the output q_e and its derivatives. Otherwise, it means that the displacement of each node of any mesh can be uniquely determined by the position of the end-effector and its derivatives, which is obviously not true in practice. In other words, the chosen output (the position of end-effector), is not flat, and therefore we cannot apply the existing differential-flatness-based method to generate feasible trajectories for the investigated soft robot.

This paper adopts numerical simulation method to solve this problem for the soft manipulator (as shown in Fig. 1) described in (1). Precisely, such a method contains the following two steps:

Step 1 : Numerically evaluate $\mathcal{W}_e$, in which we determine the initial and final positions of the desired trajectory , i.e., $q_e^{init} \in \mathcal{W}_E$ and $q_e^{fin} \in \mathcal{W}_E$;

Step 2 : Freely choose a differentiable path in $\mathcal{W}_e$ linking q_e^{init} and q_e^{fin}, and then generate the desired trajectory $q_r(t)$ for $t \in [0, T]$ where T is the prescribed time interval.

It is worth noting that the above method is to generate the desired trajectory directly in the workspace of the end-effector (i.e., directly planning the end-effector's trajectory q_e^r in **the output space** by solving the inverse dynamics to get q_r), and it is possible as well to generate the desired trajectory in the configuration space of q (i.e., planning a trajectory of q_r in the **configuration space** and then map it to get q_e^r), like the trajectory generation for rigid manipulator. This paper adopts the direct manner since it is more visible to evaluate the controller performance which will be detailed in Section 5.

Step 1: In this step, in order to evaluate the end-effector's workspace, for all $u \in \mathcal{U}$, we solve the static model $K(q)q - H(q)^T u = 0$ to compute the associated configuration q which then yields the associated end-effector position $q_e = Cq$.

Precisely, denote the control input as $u = [u_1, \cdots, u_4]^T$ for $u_i \in \mathcal{U}_i$ with $\mathcal{U}_i = [\underline{u}_i, \overline{u}_i]$ where $\underline{u}_i$ and $\overline{u}_i$ imply respectively the lower and upper bound of the ith input. Then, we discretize the set $\mathcal{U}_i$ with a chosen fixed sampling positive constant $h = \frac{\overline{u}_i - \underline{u}_i}{k}$ where k represents the number of sampled intervals. Following such a discretization, $\mathcal{U}_i$ has been sampled as

$$\mathcal{U}_i^s = \left\{\underline{u}_i, \underline{u}_i + h, \cdots, \overline{u}_i\right\} \subset \mathcal{U}_i, i = 1, 2, 3, 4$$

It is clear that, for any given $u \in \mathcal{U}^s = \mathcal{U}_1^s \times \mathcal{U}_2^s \times \mathcal{U}_3^s \times \mathcal{U}_4^s$, we can solve the algebraic equation $K(q)q - H(q)^T u = 0$ by applying the classical gradient method, which yields

$$q^{(j)} = q^{(j-1)} - \left(\nabla_q \left[K(q)q - H(q)^T u\right] \left[K(q)q - H(q)^T u\right]\right) |_{q=q^{(j-1)}}$$

where $q^{(j)}$ means the jth iteration of the value q and ∇_q represents the gradient operator with respect to q. After the convergence (in practice the convergence criteria is set as $||q^{(j)} - q^{(j-1)}|| \leq \epsilon$ where ϵ is a pre-defined threshold), then we can obtain the corresponding end-effector position via $q_e = Cq^{(j)}$ for the given input u.

It is worth noting that, for a given u, the solutions of $K(q)q - H(q)^T u = 0$ might not be globally unique, and they heavily depend on the initial value $q^{(0)}$ of the applied gradient method. To overcome this problem, we start the evaluation procedure from the rest configuration where $u_i = 0$, for which the corresponding value q is known. Then, for each evaluation we modify only one input. By using the final value obtained in the former evaluation as the initial guess of the new evaluation, we can guarantee that the applied gradient method can quickly converge to its local unique solution.

By exhaustively evaluate all inputs in $\mathcal{U}^s$, we can obtain the corresponding sampled end-effector workspace $\mathcal{W}_E^s \subset \mathcal{W}_E$ (see Fig. 4b), whose precision can be tuned by changing the sampling step h. Note that the inertial frame of the end effector $F_I(q_e)$ is chosen as the global coordinate system of FEM simulations where O is the origin of this coordinate system (see Fig. 4a).

In order to generate a feasible trajectory for the end-effector, it is required that the chosen initial and final positions should satisfy

$$q_e^{init} \in \mathcal{W}_E^s \text{ and } q_e^{fin} \in \mathcal{W}_E^s$$

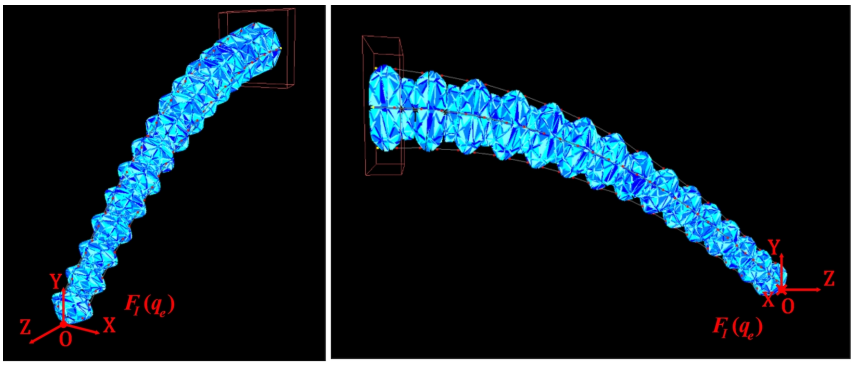
Step 2: After having obtained the sampled workspace $\mathcal{W}_E^s$, we can obtain the boundary of the real workspace, and this enables us to choose a differentiable path, noted as q_e^r, belonging to $\mathcal{W}_E$, i.e.,

$$q_e^r \subset \mathcal{W}_E$$

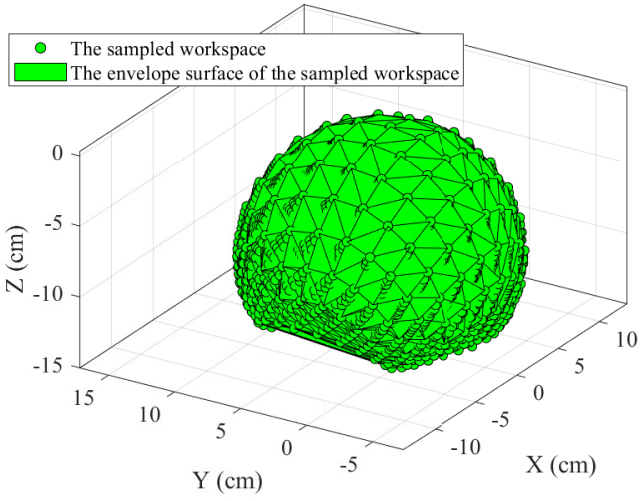
In the experimental part, we use SOFA to evaluate the workspace $\mathcal{W}_E$, and choose a path from the sampled workspace $\mathcal{W}_E^s$ obtained from SOFA between the q_e^{init} and q_e^{fin}:

$$q_e^r = (q_{e_1}^r, q_{e_2}^r, q_{e_3}^r, \cdots, q_{e_n}^r)$$

Then, we assign the time interval $[0, T]$ for this path, which logically generates a user-designed trajectory $q_e^r(t)$ with $t = 0, \tau, 2\tau ... T$. However, to achieve this planned $q_e^r(t)$, we need to find the possible input and output of the studied



(a) The coordinate system in SOFA (FEM simulations)



(b) The sampled workspace $\mathcal{W}_E^s$

Figure 4: Workspace $\mathcal{W}_{_E}$ and path planning.

robotic system that is able to approximate $q_e^r(t)$ as close as possible, which yields the following optimization problem [45]:

$$\min_{u} \quad J = \int_0^T \left[||q_e^r(s) - Cq(s)||^2 + u^T(s)u(s)\right] \mathrm{d}s \tag{2}$$

$$\begin{aligned} s.t. \quad & M\ddot{q}(t) + D(\dot{q}(t), q(t))\dot{q}(t) + K(q(t))q(t) = H(q(t))^T u(t) \\ & Cq(t_0) = q_e^r(t_0); \\ & C\dot{q}(t_0) = \dot{q}_e^r(t_0); \\ & u(t) \in \mathcal{U} \end{aligned}$$

Essentially speaking, we use the FEM-based simulator SOFA as the numerical black box to simulate the dynamical constraint $M(q)\ddot{q}+D(\dot{q},q)\dot{q}+K(q)q = H(q)^T u$, considering J as the objective function and $u(t)$ as the input variables. This process enables us to calculate the numerical gradient $\nabla_u J = \left[\frac{\partial J}{\partial u}\right]^T$. Precisely, the following procedure is applied:

Step 1: Divide the time interval $[0, T]$ into i segments: $0 = t_0 < t_1 < \cdots < t_{i-1} < t_i = T$ with $\Delta t = \frac{T}{i}$;

Step 2: Define the input $u(t)$ as following:

$$u(t) = u_k, \forall t \in \left[t_k, t_{k+1}\right], k = 0, 1, 2 \cdots i-1.$$

where $u_k = u(t_k) = \left[u_{k,1}, \cdots, u_{k,4}\right]^T = \left[u_1(t_k), \cdots, u_4(t_k)\right]^T$ is treated as a constant for $t \in [t_k, t_{k+1}]$ with $u_{k,s}$ being the value of the sth input ($0 \leq s \leq 4$) at the time t_k, and construct the input sequence as a vector:

$$U_c = \left[u_0^T, \cdots, u_{i-1}^T\right]^T \in \mathbb{R}^{4i\times 1}$$

and concatenate the corresponding configuration as

$$Q_c = \left[q^T(t_0), \cdots, q^T(t_i)\right]^T \in \mathbb{R}^{3n(i+1)\times 1}$$

Step 3: For the jth iteration, with the knowledge of U_c^j, Q_c^j and $\dot{Q}_c^j$ (for the initialization where $j = 0$ the robot is on its initial static configuration q^{init} with the corresponding input u_0, therefore $U_c^j = \left[u_0^T, \cdots, u_0^T\right]^T$, $Q_c^j = \left[[q^{init}]^T, \cdots, [q^{init}]^T\right]^T$ and $\dot{Q}_c^j = 0$), evaluate its cost function J^j;

Step 4: Based on U_c^j, generate a new sequence of control input:

$$\tilde{U}_c^j = U_c^j + [0, \cdots, 0, \delta_l^j, 0, \cdots, 0]^T$$

where δ_l^j represents the variation of the lth component of U^j. Then using SOFA to simulate the dynamics (Eq. (1)) to get the new corresponding configuration $\tilde{Q}_c^j$ and $\dot{\tilde{Q}}_c^j$ which enables us to compute the new value of cost function $\tilde{J}^j$;

Step 5: Calculate the numerical partial derivative of J with respect to the lth component of U_c as $\frac{\tilde{J}^j - J^j}{\delta_l^j}$, and concatenate all numerical partial derivatives for $1 \leq l \leq 4i$ to obtain the numerical gradient as $\nabla_{U_c^j} J^j = \left[\frac{\partial J^j}{\partial U_c^j}\right]^T$.

Step 6: Applying projected gradient descent or projected Newton method to calculate U_c^{j+1}, and use SOFA to calculate Q_c^{j+1} and $\dot{Q}_c^{j+1}$ which enables us to compute the cost function J^{j+1};

Step 7: If $|J^{j+1} - J^j| > \epsilon$ with ϵ being a pre-defined threshold, then go to Step 3 with $j = j + 1$ for another round of iteration. Otherwise, the procedure is terminated and the desired feasible trajectory is $Q_r = Q_c^{j+1}$ with $\dot{Q}_r = \dot{Q}_c^{j+1}$ and the corresponding open-loop controller is $U_r = U_c^{j+1}$.

Note that the obtained Q_r, $\dot{Q}_r$ and U_r are discrete, based on which a continuous open-loop controller $u_r(t) \in C^0$ and a differentiable trajectory $q_r(t) \in C^1$ with $\dot{q}_r(t) \in C^0$ need to be generated. For this, interpolation technique can be applied.

Precisely, to construct a continuous $u_r(t)$ from $U_r = \left[u_0^T, \cdots, u_{i-1}^T\right]^T$, the following linear interpolation is used:

$$u_r(t) = \frac{\hat{u}(t_{k+1}) - \hat{u}(t_k)}{\Delta t}(t - t_k) + \hat{u}(t_k),\ k = 0, 1, 2 \cdots i - 1; \tag{3}$$

with $\hat{u}(t_0) = u_0$, $\hat{u}(t_i) = u_{i-1}$ and $\hat{u}(t_k) = [u_k + u_{k-1}]/2$ for $1 \leq k \leq i - 1$.

Similarly, to compute a differentiable trajectory $q_r(t)$ from $Q_r = \left[q^T(t_0), \cdots, q^T(t_i)\right]^T$ and $\dot{Q}_r = \left[\dot{q}^T(t_0), \cdots, \dot{q}^T(t_i)\right]^T$, a cubic polynomial is adopted to calculate $q_r(t)$ for each interval $t \in \left[t_k, t_{k+1}\right]$ with $0 \leq k \leq i-1$ since we have position and velocity constraints at t_k and t_{k+1}. Such an interpolation yields:

$$q_r(t) = \begin{cases} b_{k0} + b_{k1}t + b_{k2}t^2 + b_{k3}t^3, & \forall t \in \left[t_k, t_{k+1}\right] \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

with b_{km} for $0 \leq m \leq 3$ being the 4 unknown polynomial coefficients on the kth interval. In total, we have $4i$ unknown parameters to be identified. Due to the fact that $q_r(t)$ should satisfy the boundary conditions of each interval $t \in \left[t_k, t_{k+1}\right]$, thus this yields the following $4i$ linear algebraic equations to be fulfilled:

$$\begin{array}{lll} \text{For } t = t_k: & b_{k0} + b_{k1}t_k + b_{k2}{t_k}^2 + b_{k3}{t_k}^3 = q_r(t_k) & \text{and} \quad b_{k1} + 2b_{k2}t_k + 3b_{k3}t_k^2 = \dot{q}_r(t_k) \\ \text{For } t = t_{k+1}: & b_{k0} + b_{k1}t_{k+1} + b_{k2}t_{k+1}^2 + b_{k3}t_{k+1}^3 = q_r(t_{k+1}) & \text{and} \quad b_{k1} + 2b_{k2}t_{k+1} + 3b_{k3}t_{k+1}^2 = \dot{q}_r(t_{k+1}) \end{array}$$

Solving the above linear equation enables us to determine the values of b_{km}, then the differentiable trajectory $q_r(t)$ and its associated derivative $\dot{q}_r(t)$ can be obtained via (4).

4. Controller design via trajectory linearization

Given the planned end-effector trajectory Cq_r via (4) and its corresponding control u_r defined in (3), the first intuitive manner to design trajectory-tracking controller is the well-known trajectory linearization method. The main idea of this approach is to conduct linearization of the FE model along the pre-obtained trajectory, and then design the controller based on the corresponding u_r.

Precisely, for the soft manipulator described in (1), we can rewrite it into the following projected space:

$$\ddot{q}_e + \alpha(q, \dot{q}) = CM(q)^{-1} H(q)^T u \tag{5}$$

with

$$\alpha(q, \dot{q}) = CM(q)^{-1} D(\dot{q}, q)\dot{q} + CM(q)^{-1} K(q)q \tag{6}$$

Then, we introduce new variables:

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} q_e \\ \dot{q}_e \end{bmatrix} \tag{7}$$

so system (5) can be transformed into the state-space form:

$$\begin{aligned} \dot{X} &= f(X, u) \\ Y &= PX \end{aligned} \tag{8}$$

where

$$f(X, u) = \begin{bmatrix} x_2 \\ CM(q)^{-1} H(q)^T u - \alpha \end{bmatrix} \tag{9}$$

and

$$P = [I, 0] \tag{10}$$

Based on the desired trajectory $Cq_r(t)$ and its corresponding controller $u_r(t)$ obtained via (2) for a pre-defined time interval $[0, T]$, we can then define

$$X_r(t) = \begin{bmatrix} Cq_r(t) \\ C\dot{q}_r(t) \end{bmatrix} \text{ and } Y_r(t) = Cq_r(t), \quad t \in [0, T] \tag{11}$$

Then, by noting

$$X(t) = X_r(t) + \Delta X(t) \text{ and } u(t) = u_r(t) + \Delta u(t) \tag{12}$$

and

$$Y(t) = Y_r(t) + \Delta Y(t) \tag{13}$$

system (8) can be then linearized around the designed trajectory as follows [46]:

$$\begin{aligned} \Delta\dot{X}(t) &= A\Delta X(t) + B\Delta u(t) \\ \Delta Y(t) &= P\Delta X(t) \end{aligned} \tag{14}$$

with

$$A = \frac{\partial f(X, u)}{\partial X}|_{X=X_r, u=u_r} \tag{15}$$

and

$$B = \frac{\partial f(X, u)}{\partial u}|_{X=X_r, u=u_r} = \begin{bmatrix} 0 \\ CM(q_r)^{-1} H(q_r)^T \end{bmatrix} \tag{16}$$

where q_r is obtained via (2), representing the corresponding displacement of all nodes along the pre-designed end-effector trajectory.

Since $P = [I, 0]$, then system (14) can be written as:

$$\begin{aligned}\Delta\ddot{Y}(t) &= PA^2\Delta X(t) + PAB\Delta u(t)\\ &= \bar{\alpha}(t) + CM(q_r)^{-1}H(q_r)^T\Delta u(t)\end{aligned} \tag{17}$$

with

$$\bar{\alpha}(t) = PA^2\Delta X(t) \tag{18}$$

In the above equation (17), we can regard $\bar{\alpha}(t)$ as a vanishing external bounded disturbance, since $\bar{\alpha}(t) = 0$ when $\Delta Y = 0$. As a result, we can propose the following robust controller:

$$u(t) = u_r(t) + \Delta u(t) \tag{19}$$

with

$$\Delta u(t) = (CM(q_r)^{-1}H(q_r)^T)^{-1}u_e(e_{q_e}, \dot{e}_{q_e}) \tag{20}$$

where the observation error e_{q_e} is defined as follows:

$$e_{q_e} = Y_r(t) - Y(t) = -\Delta Y(t) \tag{21}$$

and the term $u_e(e_{q_e}, \dot{e}_{q_e})$ has a classic PID form:

$$u_e(e_{q_e}, \dot{e}_{q_e}) = k_p e_{q_e} + k_i \int_0^t e_{q_e}\mathrm{d}s + k_d\dot{e}_{q_e} \tag{22}$$

where the diagonal gain matrices k_p, k_i, k_d must have the appropriate dimension.

Considering (17) to (22) together, we can finally reach:

$$\ddot{e}_{q_e} + k_d\dot{e}_{q_e} + k_p e_{q_e} + k_i \int_0^t e_{q_e}\mathrm{d}s = \bar{\alpha}(t) \tag{23}$$

which is a classical 3-order linear system with a bounded disturbance $\bar{\alpha}(t)$ which will be vanished when $e_{q_e} = 0$. Therefore, we can always find matrices k_p, k_i, k_d to guarantee that e_{q_e} converges to 0.

When the robot is perpetuated by disturbance, then the derived closed-loop system can be written as

$$\ddot{e}_{q_e} + k_d\dot{e}_{q_e} + k_p e_{q_e} + k_i \int_0^t e_{q_e}\mathrm{d}s = \bar{\alpha}(t) + d(t) \tag{24}$$

where $d(t)$ represents external disturbance. According to (24), the proposed controller provides robustness with respect to vanishing or piece-wise constant (or slowly time-varying) disturbance $d(t)$ due to the integral term in the proposed controller and $\bar{\alpha}(t)$ is vanishing. For other types of disturbance (such as quickly time-varying), if they are bounded, e_{q_e} will be bounded as well considering the input-state-stable (ISS) property provided by (24).

5. Experimental testing

The experimental testing regarding control feasibility and robustness was conducted to validate the proposed control strategy. The overall setup of the testing platform (see Fig. 5) mainly has three parts: the mechanical base, the Polhemus magnetic position sensor, the micro communication electronic board and the actuation system. In our work, 4 different tests were implemented: 2 feasibility tests for comparison between a common PID controller and the proposed trajectory tracking controller; 2 robustness analysis tests regarding rejecting temporary and permanent disturbances respectively. Similar to the FEM simulations shown in Fig. 4a, the experimental coordinate system of the position sensor F_s is set as $F_s = F_I(q_e)$. As shown in Fig. 5, O represents the origin of the coordinate system.

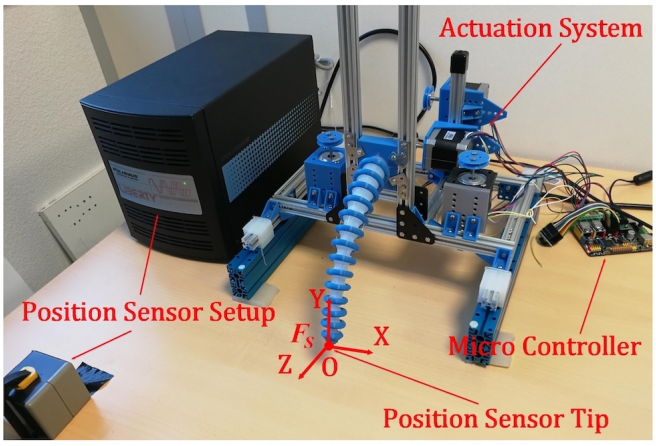
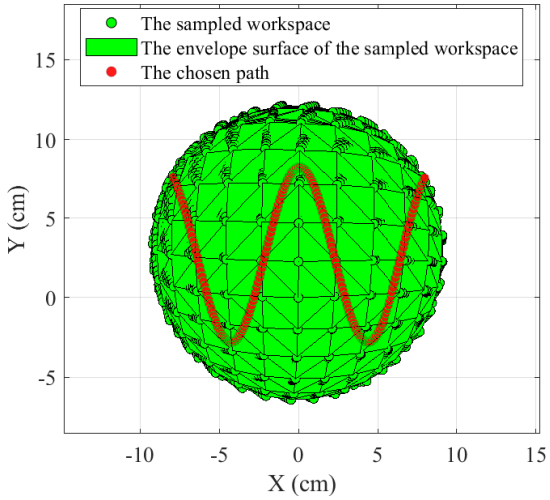
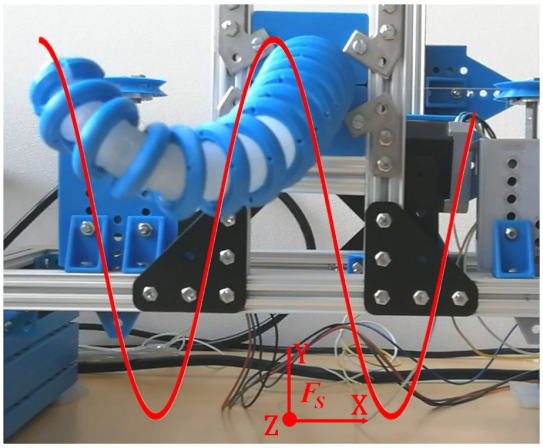


Figure 5: Experimental settings



(a) The planned trajectory obtained from $\mathcal{W}_E^s$



(b) The practical tracked trajectory

Figure 6: The designed trajectory

5.1. Trajectory generation

Following the two steps described in Section 3.2, we chose a sinusoidal-shape path located on the top of the external envelop surface of sampled workspace W_E^s and added a pre-designed time constraint into the chosen path, which yields the planned trajectory shown in Fig. 6a. Then, the practical tracked trajectory can be found accordingly (see Fig. 6b). This trajectory was used as the targeted trajectory both for the feasibility tests and robustness analysis. The reasons mainly lie in two parts: firstly, it is a typically 3D trajectory; secondly, it contains acceleration and deceleration processes, which provides good indices to evaluate the performance of the proposed controller.

5.2. Feasibility tests

5.2.1. Trajectory tracking using a common PID controller

In this test, the trunk robot was expected to track the planned sinusoidal shape trajectory as shown in Fig. 6b using a common PID controller. To develop the PID controller, we linearize the studied system around the initial position to determine the gains of the PID controller. Precisely, for (5), we linearize $CM(q)^{-1}H(q)^T$ around its initial configuration $q_r(t_0)$, noted as $\Gamma \in \mathbb{R}^{3\times4}$, then the designed PID takes the following form:

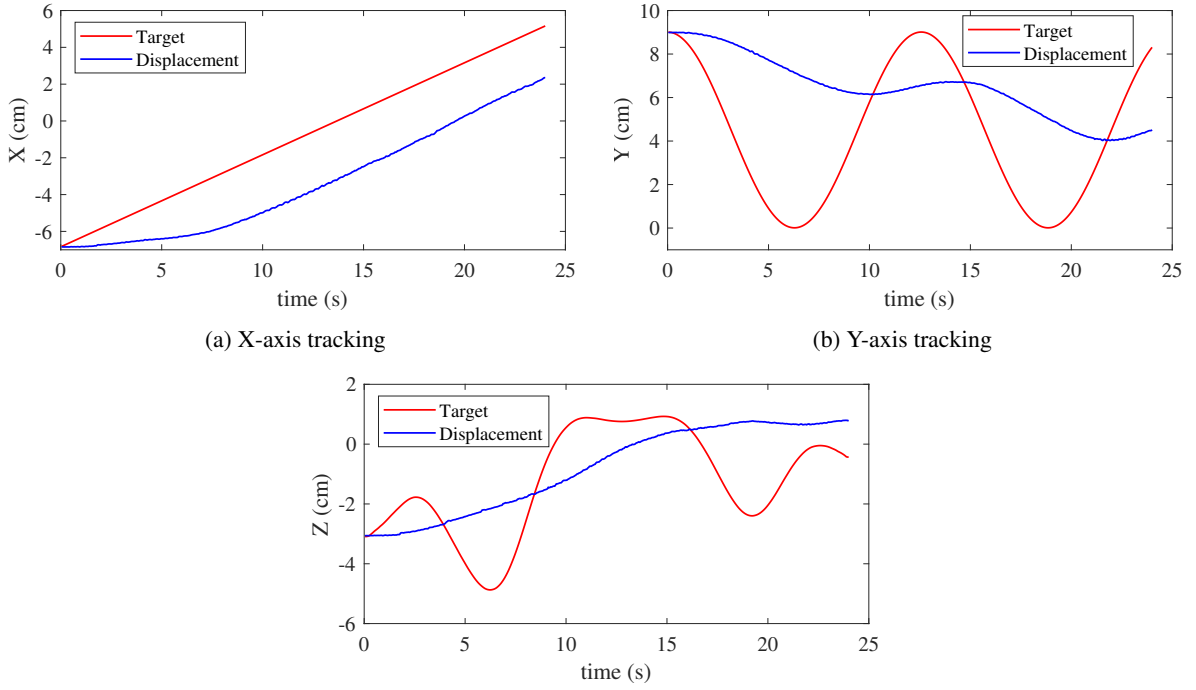
$$u = \Gamma^{-1}(K_d\dot{e}_{q_e} + K_p e_{q_e} + K_i \int_0^t e_{q_e}(s)ds) + u_r(t_0)$$

where $e_{q_e} = q_e^r - q_e$, $K_d, K_p, K_i \in \mathbb{R}^{3\times3}$. Therefore the closed-loop system is

$$\ddot{e}_{q_e} + K_d\ddot{e}_{q_e} + K_p\dot{e}_{q_e} + K_i e_{q_e} = \dot{\bar{\alpha}}(t)$$

where $\bar{\alpha}(t) = \ddot{q}_e^r + \alpha(t) - \left[CM^{-1}H(q)^T - \Gamma\right]u$. Finally the gains of PID (i.e., K_d, K_p and K_i) are determined by guaranteeing that $\ddot{e}_{q_e} + K_d\ddot{e}_{q_e} + K_p\dot{e}_{q_e} + K_i e_{q_e} = 0$ is stable.

According to Fig. 7, it can be easily noticed that the robot end-effector was not able to catch up with the trajectory and ended up with a huge amount of displacement delay. The results imply that a common PID controller is not valid in terms of tracking a relatively quick and complex trajectory in this case. Theoretically speaking, the PID controller only works locally around $q_r(t_0)$ if the linearization matrix Γ is close to $CM^{-1}H(q)^T$, and will fail if their values are far enough. Similarly, the PID controller will also fail if the value of $[\ddot{q}_e^r + \alpha(t)]$ varies fast. In our case, the mentioned two reasons may both partially contribute to the failure of this test.



(a) X-axis tracking

(b) Y-axis tracking

(c) Z-axis tracking

Figure 7: Trajectory tracking using a common PID controller

5.2.2. Trajectory tracking using the proposed controller

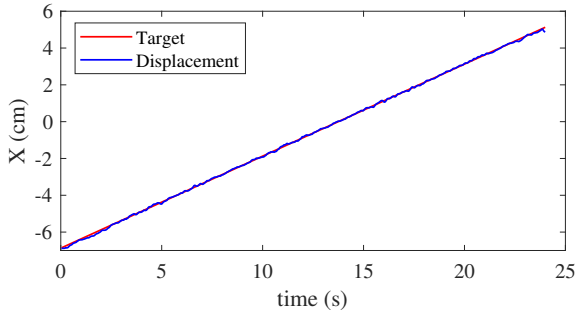
In this test, we used the proposed controller instead to track the same sinusoidal shape trajectory (Fig. 6b). The results are displayed in Fig. 8 which demonstrates that the robot end effector effectively tracked the trajectory with very little hysteresis. In other words, the proposed controller has been proved to be feasible to track a relatively quick and complex trajectory. Likewise, the results have also proved that the unknown term $\bar{\alpha}(t)$ defined in Eq. 23 can be regarded as a bounded disturbance, which agrees with our theoretical assumption.

5.3. Robustness Analysis

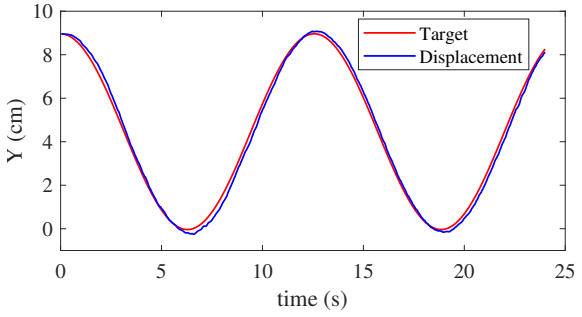
To validate the robustness of the proposed controller, the process of rejecting temporary disturbance and permanent disturbance was observed in 2 different tests (see Fig. 9).

5.3.1. Temporary disturbance

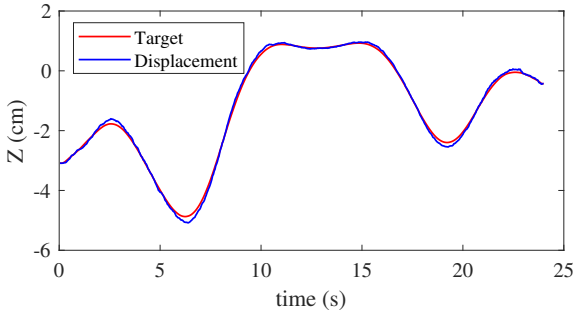
In this test, the trunk robot was expected to track the same sinusoidal shape trajectory (Fig. 6b) under four temporary disturbances exerted by a pen as shown in Fig. 9a. The corresponding results are displayed in Fig. 10



(a) X-axis tracking

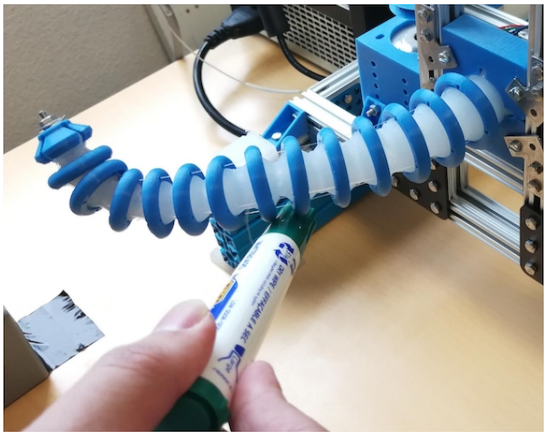


(b) Y-axis tracking

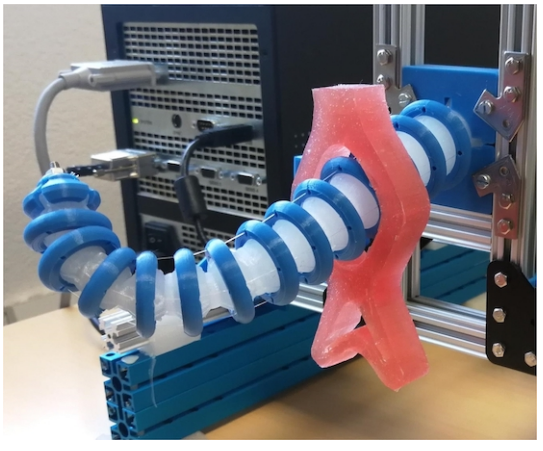


(c) Z-axis tracking

Figure 8: Trajectory tracking using the proposed controller



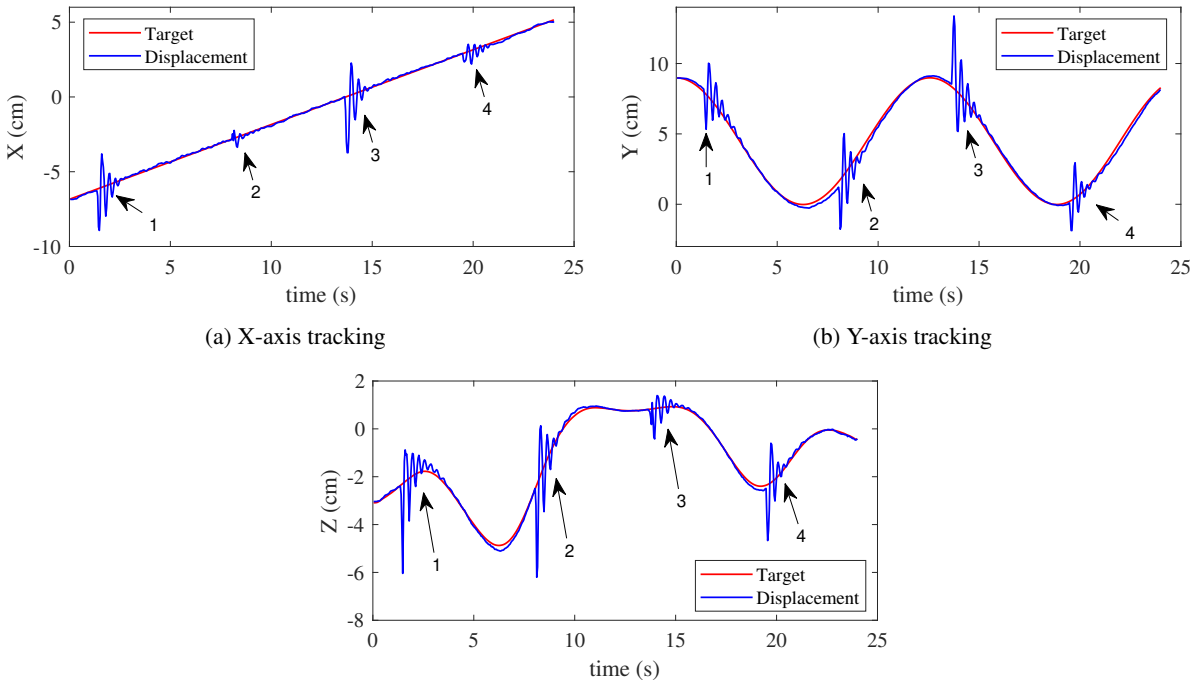
(a) Temporary-disturbance testing



(b) Permanent-disturbance testing

Figure 9: Experimental settings

where the four disturbances are marked as 1 - 4: the robot end-effector was able to quickly return to and stably catch up with the trajectory even after being affected by four relatively large temporary disturbances (see Fig. 10a, Fig. 10b and Fig. 10c), which implies that the proposed controller is valid for rejecting vanishing bounded disturbances. This experimental phenomenon explains why we can consider $d(t)$ defined in Eq. 24 as a vanishing disturbance, and this term can be logically eliminated due to the existence of the integral term in Eq. 24.



(a) X-axis tracking

(b) Y-axis tracking

(c) Y-axis tracking

Figure 10: Temporary disturbance test using the proposed controller

5.3.2. Permanent disturbance

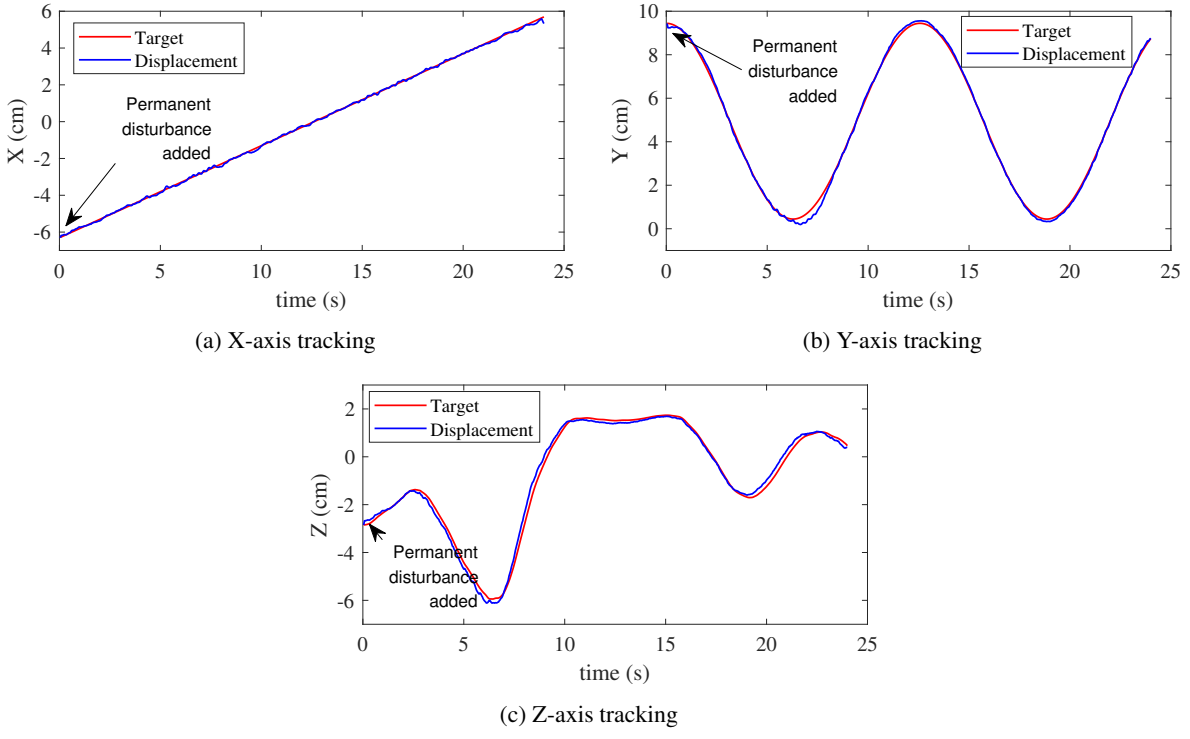
In this test, the robot was planned to track the sinusoidal shape trajectory (Fig. 6b) while carrying a heavy piece of red rubber from the beginning of the trajectory-tracking process. In particular, the extra piece of material on the robot caused the displacement delay at the beginning but the controller rapidly dealt with this permanent disturbance quickly made up for the hysteresis in a very short time (see Fig. 11a, Fig. 11b and Fig. 11c). Therefore, the proposed controller has demonstrated reliable robustness with respect to bounded permanent disturbances. In this case, the piece of rubber can be considered as a permanent constant disturbance, which essentially agrees with the definition of $d(t)$ defined in Eq. 24. Logically, this term can be eliminated by the integral term defined in Eq. 24 if $d(t)$ is constant or piecewise constant.

6. CONCLUSIONS

Soft robots have been attracting more and more research attention due to their unique properties that are different from conventional rigid robots. Their compliance and flexibility are considered as promising features for the current robotic systems. However, many scientific challenges regarding dynamically controlling these novel robots still exist mainly because of their inherently complex nature. In this paper, we proposed a dynamic controller for a soft trunk robot to track trajectories. The feasibility has been validated through experimental testing, and the robustness of the proposed method has been illustrated by rejecting both the temporary and permanent external disturbances. Note that the axial elongation of the robot body can not be achieved by the current set up since the needed actuators (such as pneumatic actuators) to stretch the robot have not been integrated in this robotic system. However, this will be further discussed and investigated in our future work. Besides, we will also promote this control strategy on other soft robots as well as optimizing it for better control performance.

References

[1] Deepak Trivedi, Christopher D Rahn, William M Kier, and Ian D Walker. Soft robotics: Biological inspiration, state of the art, and future research. Applied bionics and biomechanics, 5(3):99–117, 2008.



(a) X-axis tracking

(b) Y-axis tracking

(c) Z-axis tracking

Figure 11: Permanent disturbance test using the proposed controller

[2] Daniela Rus and Michael T Tolley. Design, fabrication and control of soft robots. Nature, 521(7553):467–475, 2015.

[3] Sangbae Kim, Cecilia Laschi, and Barry Trimmer. Soft robotics: a bioinspired evolution in robotics. Trends in biotechnology, 31(5):287–294, 2013.

[4] Andrew D Marchese, Russ Tedrake, and Daniela Rus. Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator. The International Journal of Robotics Research, 35(8):1000–1019, 2016.

[5] Thor Morales Bieze, Frederick Largilliere, Alexandre Kruszewski, Zhongkai Zhang, Rochdi Merzouki, and Christian Duriez. Finite element method-based kinematics and closed-loop control of soft, continuum manipulators. Soft robotics, 5(3):348–364, 2018.

[6] William S Rone and Pinhas Ben-Tzvi. Continuum robot dynamics utilizing the principle of virtual power. IEEE Transactions on Robotics, 30(1):275–287, 2013.

[7] Isuru S Godage, Gustavo A Medrano-Cerda, David T Branson, Emanuele Guglielmino, and Darwin G Caldwell. Dynamics for variable length multisection continuum arms. The International Journal of Robotics Research, 35(6):695–722, 2016.

[8] Hongbin Fang, Yetong Zhang, and KW Wang. Origami-based earthworm-like locomotion robots. Bioinspiration & biomimetics, 12(6):065003, 2017.

[9] Stanislao Grazioso, Giuseppe Di Gironimo, and Bruno Siciliano. A geometrically exact model for soft continuum robots: The finite element deformation space formulation. Soft robotics, 6(6):790–811, 2019.

[10] D Caleb Rucker and Robert J Webster III. Statics and dynamics of continuum robots with general tendon routing and external loading. IEEE Transactions on Robotics, 27(6):1033–1044, 2011.

[11] Frédéric Boyer, Shaukat Ali, and Mathieu Porez. Macrocontinuous dynamics for hyperredundant robots: application to kinematic locomotion bioinspired by elongated body animals. IEEE Transactions on Robotics, 28(2):303–317, 2011.

[12] Long Li, Tao Jin, Yingzhong Tian, Fei Yang, and Fengfeng Xi. Design and analysis of a square-shaped continuum robot with better grasping ability. IEEE Access, 7:57151–57162, 2019.

[13] Jessica Burgner-Kahrs, D Caleb Rucker, and Howie Choset. Continuum robots for medical applications: A survey. IEEE Transactions on Robotics, 31(6):1261–1280, 2015.

[14] Manas M Tonapi, Isuru S Godage, AM Vijaykumar, and Ian D Walker. A novel continuum robotic cable aimed at applications in space. Advanced Robotics, 29(13):861–875, 2015.

[15] Michael W Hannan and Ian D Walker. Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots. Journal of robotic systems, 20(2):45–63, 2003.

[16] Bryan A Jones and Ian D Walker. Kinematics for multisection continuum robots. IEEE Transactions on Robotics, 22(1):43–55, 2006.

[17] Robert J Webster III and Bryan A Jones. Design and kinematic modeling of constant curvature continuum robots: A review. The International Journal of Robotics Research, 29(13):1661–1683, 2010.

[18] Valentin Falkenhahn, Alexander Hildebrandt, Rüdiger Neumann, and Oliver Sawodny. Model-based feedforward position control of constant

curvature continuum robots using feedback linearization. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 762–767. IEEE, 2015.

[19] Cosimo Della Santina, Robert K Katzschmann, Antonio Biechi, and Daniela Rus. Dynamic control of soft robots interacting with the environment. In 2018 IEEE International Conference on Soft Robotics (RoboSoft), pages 46–53. IEEE, 2018.

[20] Robert K Katzschmann, Cosimo Della Santina, Yasunori Toshimitsu, Antonio Bicchi, and Daniela Rus. Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model. In 2019 2nd IEEE International Conference on Soft Robotics (RoboSoft), pages 454–461. IEEE, 2019.

[21] Isuru S Godage, Gustavo A Medrano-Cerda, David T Branson, Emanuele Guglielmino, and Darwin G Caldwell. Modal kinematics for multisection continuum arms. Bioinspiration & biomimetics, 10(3):035002, 2015.

[22] Tobias Mahl, Alexander Hildebrandt, and Oliver Sawodny. A variable curvature continuum kinematics for kinematic control of the bionic handling assistant. IEEE transactions on robotics, 30(4):935–949, 2014.

[23] Gundula Runge and Annika Raatz. A framework for the automated design and modelling of soft robotic systems. CIRP Annals, 66(1):9–12, 2017.

[24] Beobkyoon Kim, Junhyoung Ha, Frank C Park, and Pierre E Dupont. Optimizing curvature sensor placement for fast, accurate shape sensing of continuum robots. In 2014 IEEE international conference on robotics and automation (ICRA), pages 5374–5379. IEEE, 2014.

[25] Mohamed Taha Chikhaoui, Kanty Rabenorosoa, and Nicolas Andreff. Kinematics and performance analysis of a novel concentric tube robotic structure with embedded soft micro-actuation. mechanism and Machine Theory, 104:234–254, 2016.

[26] B. A. Jones, R. L. Gray, and K. Turlapati. Three dimensional statics for continuum robotics. In 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2659–2664, Oct 2009.

[27] Federico Renda, Matteo Cianchetti, Michele Giorelli, Andrea Arienti, and Cecilia Laschi. A 3d steady-state model of a tendon-driven continuum soft manipulator inspired by the octopus arm. Bioinspiration & biomimetics, 7(2):025006, 2012.

[28] Bin Zhao, Lingyun Zeng, Zhonghao Wu, and Kai Xu. A continuum manipulator for continuously variable stiffness and its stiffness control formulation. Mechanism and Machine Theory, 149:103746, 2020.

[29] Federico Renda, Michele Giorelli, Marcello Calisti, Matteo Cianchetti, and Cecilia Laschi. Dynamic model of a multibending soft robot arm driven by cables. IEEE Transactions on Robotics, 30(5):1109–1122, 2014.

[30] Ahmad Abu Alqumsan, Suiyang Khoo, and Michael Norton. Robust control of continuum robots using cosserat rod theory. Mechanism and Machine Theory, 131:48–61, 2019.

[31] Azadeh Doroudchi and Spring Berman. Configuration tracking for soft continuum robotic arms using inverse dynamic control of a cosserat rod model. In 2021 IEEE International Conference on Soft Robotics, RoboSoft, 2021.

[32] John Till, Caroline E Bryson, Scotty Chung, Andrew Orekhov, and D Caleb Rucker. Efficient computation of multiple coupled cosserat rod models for real-time simulation and control of parallel continuum manipulators. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 5067–5074. IEEE, 2015.

[33] Kai Xu and Nabil Simaan. Analytic formulation for kinematics, statics, and shape restoration of multibackbone continuum robots via elliptic integrals. Journal of Mechanisms and Robotics, 2(1), 2010.

[34] Jinzhao Yang, Haijun Peng, Wenya Zhou, Jie Zhang, and Zhigang Wu. A modular approach for dynamic modeling of multisegment continuum robots. Mechanism and Machine Theory, 165:104429, 2021.

[35] Christian Duriez. Control of elastic soft robots based on real-time finite element method. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 3982–3987. IEEE, 2013.

[36] Olivier Goury and Christian Duriez. Fast, generic, and reliable control and simulation of soft robots using model order reduction. IEEE Transactions on Robotics, 34(6):1565–1576, 2018.

[37] Singiresu S Rao. The finite element method in engineering. Butterworth-heinemann, 2017.

[38] Gang Zheng, Yuan Zhou, and Mingda Ju. Robust control of a silicone soft robot using neural networks. ISA transactions, 100:38–45, 2020.

[39] Zhongkai Zhang, Jeremie Dequidt, Alexandre Kruszewski, Frederick Largilliere, and Christian Duriez. Kinematic modeling and observer based control of soft robot using real-time finite element method. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5509–5514. IEEE, 2016.

[40] G. Zheng, O. Goury, M. Thieffry, A. Kruszewski, and C. Duriez. Controllability pre-verification of silicon soft robots based on finite-element method. In Robotics and Automation (ICRA), 2019 IEEE International Conference on.

[41] Zhongkai Zhang, Jérémie Dequidt, and Christian Duriez. Vision-Based Sensing of External Forces Acting on Soft Robots Using Finite Element Method. IEEE Robotics and Automation Letters, 3(3):1529 – 1536, February 2018.

[42] Ke Wu and Gang Zheng. Fem-based gain-scheduling control of a soft trunk robot. IEEE Robotics and Automation Letters, 6(2):3081–3088, 2021.

[43] Walid Amehri, Gang Zheng, and Alexandre Kruszewski. Fem based workspace estimation for soft robots: a forward-backward interval analysis approach. In 2020 3rd IEEE International Conference on Soft Robotics (RoboSoft), pages 170–175. IEEE, 2020.

[44] Alessandro De Luca and Giuseppe Oriolo. Trajectory planning and control for planar robots with passive last joint. The International Journal of Robotics Research, 21(5-6):575–590, 2002.

[45] Franz Gritschneder, Knut Graichen, and Klaus Dietmayer. Fast trajectory planning for automated vehicles using gradient-based nonlinear model predictive control. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7369–7374. IEEE, 2018.

[46] Joao P Hespanha. Linear systems theory. Princeton university press, 2018.