2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)
April 4-8, 2022, Edinburgh, UK

Task-oriented robot control user interface designer for cable-driven soft robots without kinematic models

Haoyan Li, Tomoka Nishino, Binod Dhakal, Hironori Mitake, Shoichi Hasegawa

2022 IEEE 5th International Conference on Soft Robotics (RoboSoft) | 978-1-6654-0828-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ROBOSOFT54090.2022.9762102

***Abstract*—This paper introduces a method to allow casual users to create task-oriented control user interfaces (UIs) for cable-driven soft robots without prior knowledge, and the extra time and labor costs in measurements or calibration. The task space created by the proposed method is that it no longer must be physics-based, which provides more freedom, especially for casual users. Any control UI created previously could be used to create new UIs. We also develop a graphical user interface (GUI) implementing the proposed method for a feasibility evaluation. Practical cases that use the proposed GUI to create control UIs for particular tasks of a cable-driven soft robot arm system have been investigated. Finally, we conduct a concise user evaluation to test the usability of the proposed method for casual users.**

***Index Terms*—Soft robotics, Robot kinematics, Robot control, task-oriented control**

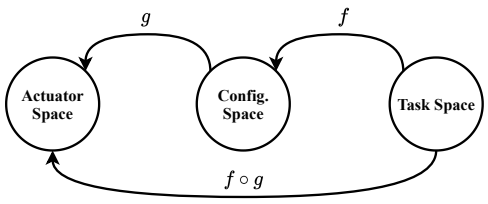
I. Introduction

As an emerging and rapidly developing field, soft robotics has received much attention from researchers in recent years, inspired by natural biology and benefiting from the development of new materials and mechanisms [1]–[5]. Compared to traditional rigid-body robots, a soft robot has intrinsic safety with its environments and operators due to its compliant nature. Deformable materials have the potential to exhibit adaption for unpredictable environments. Owing to soft and lightweight materials, they also ensure agility and efficiency for robot systems. Researchers expect that these capabilities that are different from conventional rigid body robots could bring more possibilities for the future of robotics.

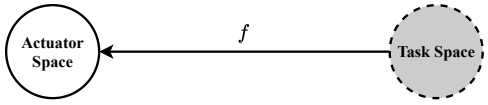
However, beyond the advantages, the soft nature also presents a whole new set of challenges [6] that may not arise in conventional robotics. Conventional approaches to robot control assume an inherent linkage structure consisting of movable joints and unchanged links for a robot. The kinematics for a rigid robot could be modeled as a set of coordinate transformations of joint rotations. Meanwhile, deformations of the materials occur on the whole body of the soft robot, and the distribution of the deformations depends on the soft robot's design. Any change in the deformable structure of a soft robot, even if it might only partially affect the structure, would cause a significant change in the robot's kinematics. Additionally, a deterministic mathematical model for the kinematics, composed of actuator's angles or displacements, is not evident to a soft robot.

In order to compose such a kinematic model, researchers often consider an appropriate modeling method, such as Piece-wise Constant Curvature (PCC) or the Finite Elements Model (FEM). The PCC [1], [7] understands a soft robot under a simplified assumption, whose performance depends on the robot's morphology and the characteristics of the actuation system. The FEM [8], on the other hand, models the complex deformations of soft robots, but its computation results would be significantly impacted by slight changes in some material parameters.

Using a data-driven calibration method, like goal babbling learning [9] or neural networks [10], to develop a kinematic model allows researchers to avoid parameters fitting for various deformable materials. However, to develop a well-defined model, it is critical to create sufficient, high-quality datasets that cover the motion range of actuators of soft robots as much as possible with correct labels. It requires a significant investment of time and labor to prepare such a dataset.



(a) Conventional Methods



(b) Proposed Method

Fig. 1. The mappings of inverse kinematic models. The task space of a conventional method usually consists of end-effectors' coordinates, whereas our method is flexible in that users can design it depending on their interpretations. A configuration space usually depends on the selected model of robot, which is ignored by our method. An actuator space directly consists of control parameters that are related to actuators.

Conventional approaches, in essence, try to find the maps between three robotics spaces, the actuator space, the configuration space, and the task space (see Figure 1). For an inverse kinematic model in particular, the task space is often defined by the end-effectors' coordinates of a soft robot, which makes covering all the aspects difficult. Moreover, conventional approaches also require substantial prerequisites of professional knowledge. The challenge of building an appropriate kinematic model for soft robots restricts the

development of soft robots.

In this paper, we introduce a novel method to allow casual users to create task-oriented control user interfaces (UIs) for cable-driven soft robots without prior knowledge and extra time and labor costs such as in measurements or calibration. The task space created by the proposed method is that it no longer has to be physics-based, which provides more freedom, especially for casual users. Any control UI created previously could be used for other UIs. We also developed a graphical user interface (GUI) implementing the proposed method for the feasibility evaluation. Several practical cases that use the proposed GUI to create control UIs for particular tasks of a cable-driven soft robot arm system have been considered. Finally, we conduct a user evaluation to test the usability of the proposed method for casual users.

The paper is organized as follows: Section II introduces several works in the fields of soft robotics and computer graphics (CG). In Section III, we formulate the concept of our method in detail, and the implementation is presented in Section IV. Section V introduces the GUI and the soft robot arm system used in the evaluation and illustrates some typical usages of our method. Section VI describes the user evaluation and provides the results and discussion. Finally, we provide some conclusions and identify future works in Section VII.

II. Related Works

Various conventional approaches have been proposed to find a kinematic model to control soft robots. Takase et al. [11] proposed an inverse kinematic (IK) model and calibration method to generate motions for a cable-driven soft toy robot. Duriez [8] proposed a real-time FEM method to control elastic soft robots, while Bern et al. [12] proposed a Soft IK to find a set of activations for each contractile element that deforms the model mesh based on the FEM method. A series of works by Giorelli et al. [10] addressed the challenges of finding a kinematic model for a cable-driven soft robot. They have implemented a Jacobian-based computational approach and a neural network approach to learning an inverse kinematic model. Schelagenhauf et al. [13] presented a series of control strategies for soft manipulators. Their method was used to control multi-fingered tendon-driven hands by a glove-type control interface. They also implemented several methods, including FEM-based and learning-based, to solve inverse kinematics and compared the performance of the methods.

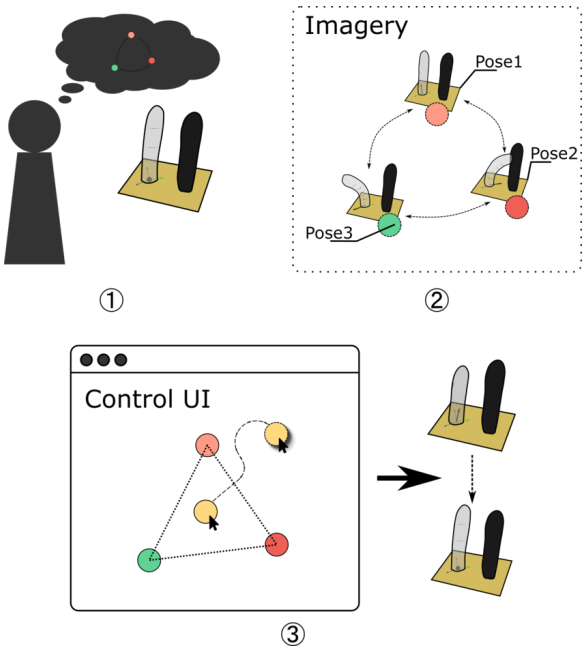
In the computer-assisted animation or computer graphics (CG) field, several methods have been proposed as the "reparameterization," which has a similar procedure that maps an artificial data space, sometimes called "state space," to the configuration space of CG characters or graphics. However, the purpose of using mapping in our method is different from the works in CG. For CG, the control points in the configuration space are made for some specific purpose. For example, the purpose of setting two control points for the Cubic Bezier algorithm is to create a smooth curve. The calculations behind algorithms are generally apparent to users. On the other hand, the relationship between actuator space and the robot's posture depends on the configuration of the actuators, according to the design of a soft robot, which is not apparent to users. Moreover, the CG object typically has a higher degree of freedom than a robot. Most of its possible configuration sets only show meaningless and useless visuals. Hence, using the "reparameterization" method in CG reduces the high dimensionality caused by a complex configuration of its control algorithm. In contrast, our method lets users define an uncertain relationship between the robot's posture and displacements of the actuators.

Rademacher et al. [14] have proposed a view-dependent model to adapt artistic distortions of 2D cel animation onto 3D character models for 3D animation. The distortions were preserved in a geometry that was a minimal composition of a simplicial map. The triangle consisted of three camera view-related positions. Ngo et al. [15] showed the simplicial configuration modeling could allow users to edit a graphic object freely while keeping the visual integrity of computer graphics by embedding free-form constraints. Both used a simplicial complex model to preserve data structure and interpolated a composite image or a posture of a 3D model from new input, which is also used in our methods. Igarashi et al. [16] introduced a system-implemented spatial keyframing technique for performance-driven character animation. Their results showed the usability and efficiency of their system for casual users to create an expressive animation of imaginary characters. However, since the spatial coordinates in a 3D space are the key component of this technique, their method is not applicable for some spatial-unrelated motions.

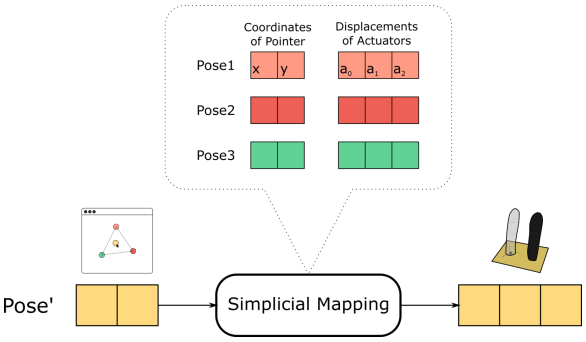
III. Proposed Method

Our aim is to create an intuitive user interface (UI) for soft robot control regarding users' interpretation of intended tasks. In the case of a robot with a kinematic model, the end-effector's space is often used as the task space for the robot controller. Contrarily, here we do not have a kinematic model for a soft robot, and neither its end-effector space nor end-effectors are known. Instead of preparing a kinematic model, users could use the imagery of the robots and tasks in their minds.

Our proposed method supposes that a user has task imagery in mind as a graph structure, whose nodes correspond to the states of the robot, which are also in the user's mind. The method then translates the graph into a graphical user interface on a two-dimensional display. This process consists of four procedures, as shown in Figure 2. First, we ask the user to copy the graph of the task imagery into a graph on the GUI, which represents the task space. Next, to make the graph usable as a robot control UI, we transform the graph into a simplicial complex, where points inside could be interpolated. After defining the simplicial complex, users assign actuator's displacements that direct the state of the robot corresponding to each node in mind. Finally, this continuously defined simplicial complex represents the robot



(a) Procedures



(b) Data structure inside simplicial mapping

Fig. 2. The concept underlying the proposed method.

task space. Therefore, the user could use it as the robot control UI by manipulating a control point on the simplicial complex to make the soft robot adopt desired postures.

A. Imaging Task Graph

When users plan to enact a soft robot task, such as adopting a particular posture or taking an object, they usually have unique interpretations in their minds for the target task, according to their knowledge backgrounds and observations of the soft robot and the environment. As one of the possible embodiments for these interpretations, the graph structure is chosen here. A node represents one particular task state corresponding to a specific posture of a soft robot in users' minds. A transition between two different states would be represented as an edge. Hence we have a state graph with a set of task state nodes and a set of state transition relationships.

B. Transforming Graph to Simplicial Complex

In order to use the graph as a robot control GUI, the transition between two different state nodes has to be continuous. This requires the interpolation of nodes, which are connected by edges. Here we choose the simplicial complex to achieve this goal. Any arbitrary point inside the simplicial complex could be interpolated by barycentric coordinates. Consider that for a simplicial complex, the interior region is only defined by the barycentric coordinates. In order to create a useful robot controller, the exterior of the simplicial complex should be defined as well. In Section IV, we will introduce the implementation of the extrapolation method in detail.

The reason we choose a simplicial complex to interpolate the task space is its robust topological characteristic. The states and their generated space will always be preserved by the simplicial complex itself, regardless of the geometric realization. Another interpolation method, the radial basis function (RBF), is widely used in different fields [16], [17]. The RBF interpolation depends on the choice of a distance function. In our case, distances, or the geometric shape of the task graph, are not essential for our purposes, as we focus on the nodes and their transitions. Using a topological space to represent our task state space is a more appropriate choice.

Most people are unfamiliar with spaces of more than three dimensions. However, for the either original graph in their minds or the transformed simplicial complex, the dimension of the structure could be more than three. For example, the shape of a complete graph with four nodes is a tetrahedron, namely a non-planar graph. The high dimensionality may cause many difficulties in visualization in a 2D GUI. Meanwhile, people are used to using pen and paper to illustrate their ideas. Therefore, we limit the maximum dimension of a simplicial complex to two (up to triangles), and the simplicial complex in task space will be drawn on a 2D canvas of the proposed GUI.

To assist in creating a simplicial complex more effectively, the transformation from a graph to a simplicial complex can also be achieved via the Delaunay triangulation [18] algorithm, in addition to specifying triangles directly. Users select and group target state nodes that have transitions with each other, and the algorithm will help construct the target simplicial complex on the 2D GUI.

C. Assigning Actuator's Displacements

Since we have the simplicial complex representing the task space drawing on the UI, the next step is to assign each state node of the simplicial complex with a specific set of displacements of the soft robot's actuators. The displacements represent a corresponding soft robot's posture that matches the user's impressions of the task node. Users could also adjust the task space to be more precise by adding more state nodes into the existing simplicial complex.

D. Control Robot in Task Space

Once assigning all the nodes with their corresponding actuator displacements, a piece-wise linear, continuous simplicial map is generated between two homeomorphic topological spaces: the task and actuator spaces. Any input at an arbitrary point in the task space will be mapped to a set of robot actuators' displacements that represent the robot's expected posture in users' minds.

E. Multi-maps

Notice that, sometimes, the user's interpretation of the task might include several different graphs. For example, when manipulating a soft robot to catch and release an object, one control UI controls catch and release, and the other controls the position where the catch-and-release happens. In that case, users could use two simplicial complexes to represent two different task state graphs and then manipulate these two simplicial complexes simultaneously to achieve the task. This combination of two or more simplicial maps could be considered as bi-linear or multi-linear simplicial maps. Details about multi-linear simplicial maps will be described in Section IV.

F. Chain of Simplicial Maps

In order to maximize the reusability of task spaces that we created, instead of actuator space, the output of the higher-level task space could be a point in a lower-level task space which represents a fundamental manipulation task of the soft robot. This feature of reusability of task spaces is called in this paper the chain of simplicial maps. Using this feature, users can create different control UIs and then combine them as a chain. For example, one might create two low-level controllers for each soft robot arm and then combine them with a high-level controller that achieves the posture performing the task of two robot arms.

IV. Implementation

In our method, we used simplicial complexes to represent users' interpretations of tasks and construct task spaces, and any arbitrary point in a simplicial complex hence could be interpolated. In order to maximize the usability of our method, we also extend an extrapolation method for the exterior of a simplicial complex, which is not defined in previous works.

A. Interpolation

A simplicial complex [15], [19] is commonly used for constructing a continuous data space from discrete data points. Its topologically invariant characteristics hold the connectivity of points with flexible geometry realizations. A k-simplex, written as Δ^k, is the simplest geometric form that consists of $k+1$ points in space. For any arbitrary point $\mathbf{v}$ in a k-simplex, there is

$$\mathbf{v} = \sum_{i=0}^{k} \lambda_i \mathbf{v}_i, \tag{1}$$

where $0 \leq \lambda_i \leq 1$, $\sum_{i=0}^{k} \lambda_i = 1$, and $\mathbf{v}_i$ are the coordinates of the vertex of the k-simplex. The set of λ_i for the point $\mathbf{v}$ is called the barycentric coordinate of point $\mathbf{v}$.

A simplicial complex $\mathcal{K}$ is a finite collection composed of simplices, which satisfies the conditions that every face of a simplex from $\mathcal{K}$ is in $\mathcal{K}$ as well, and the non-empty intersection of any two simplices $\delta_1, \delta_2 \in \mathcal{K}$ is a face of both δ_1 and δ_2. The dimension of a simplicial complex $\mathcal{K}$ is the largest dimension of any simplex in $\mathcal{K}$. Therefore, we have a set of barycentric coordinates for any arbitrary point $\mathbf{v}$ inside the simplicial complex $\mathcal{K}$, which could be written in the same form as Equation 1, where $\mathbf{v}_i$ is a vertex of the simplicial complex $\mathcal{K}$, with the rule that if $\mathbf{v}$ is inside a simplex of $\mathcal{K}$, the subset of λ_i for this simplex is the set of λ_i corresponding to its vertices of the simplex, and if $\mathbf{v}$ is outside a simplex of $\mathcal{K}$, each λ_i in the subset is zero.

B. Extrapolation

In order to allow users to exploit the task space as much as they can, the extrapolation method has to be defined since the original simplicial complex only has the interpolation inside its convex hull. The exterior of the simplicial complex, or Voronoi Region, basically has two different types, rectangle, and sector as Figure 3 shows.

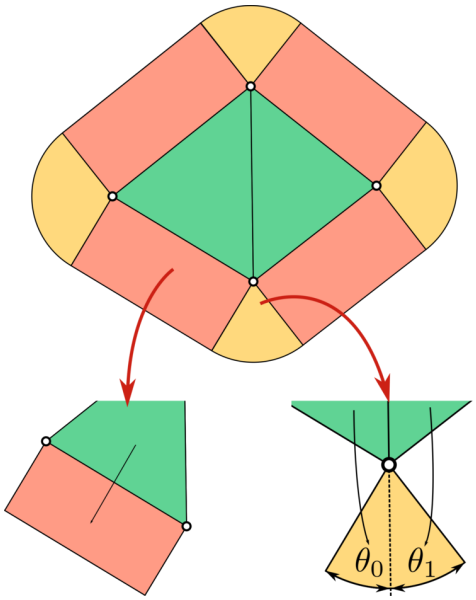


Fig. 3. Two different types of Voronoi Region. The orange zone is defined as the rectangular Voronoi region, and the yellow zone is defined as the sectoral Voronoi region.

For the rectangle type, the Voronoi Region consists of one exterior edge of the simplicial complex and two extension rays from each edge vertex, where the extension rays are perpendicular to the edge. Since this exterior edge always belongs to only one particular simplex in the simplicial complex, we then define the extrapolation method as the barycentric coordinates of this "governor" simplex, allowing the λ_i of the simplex to be any real number.

For the sector type, the Voronoi Region consists of two extension rays from the same vertex of the exterior edge, where the extension rays are perpendicular to each edge that includes the vertex. In this case, the vertex belongs to two exterior edges of the simplicial complex. The governor simplices, in other words, are two simplices that include one of the exterior edges, respectively. The point inside this Voronoi Region could be defined as a blending of two barycentric coordinates of the governor simplices according to the angles of θ_0, θ_1,

$$\mathbf{v} = \sum_{i=0}^{k} (\frac{\theta_1}{\theta}\lambda_i^0 + \frac{\theta_0}{\theta}\lambda_i^1)\mathbf{v}_i, \tag{2}$$

where $\theta = \theta_0 + \theta_1$.

C. Simplicial Map

In the third step of our method, users will assign nodes of the simplicial complex with a set of displacements of actuators or points in lower-level simplicial complexes.

Assume that we have one simplicial complex in the task space that consists of four task nodes, $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4$, and three actuators to achieve the task. Hence, a point v in this task space could be represented as the vector form, $\mathbf{v} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$, and its basis is $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$. For each node v_i, $i \in \{1, 2, 3, 4\}$, we have a particular set of displacements of three actuators, $\mathbf{w}_i = \{w_{i1}, w_{i2}, w_{i3}\}$ whose basis is the standard basis. Therefore, the simplicial map f will be

$$f(\mathbf{v}) = \sum_{i=1}^{4} \lambda_i g(\mathbf{v}_i), \tag{3}$$

where $g(\mathbf{v}_i) = \mathbf{w}_i$.

We now generalize the previous situation and define the vertex map from the task space to the actuator space or the lower level task space to be written as a function $g(\mathbf{v}_i) = \mathbf{w}$, where $\mathbf{v}_i = \{\lambda_{i,0}, \ldots, \lambda_{i,k}\}$ and $\mathbf{w}$ is a set of values corresponding to actuators or points in low-level task space. Then g can be extended to a continuous map $f : |\mathbf{V}| \to |\mathbf{W}|$ defined by

$$f(\mathbf{v}) = \sum_{i=0}^{k} \lambda_i g(\mathbf{v}_i), \tag{4}$$

where $\mathbf{v} = \{\lambda_0, \ldots, \lambda_k\}$ is the new input of the simplicial complex.

D. Multi-linear Simplicial Maps

Multi-linear simplicial mapping is used for the case the user's imagery has multiple graphs such as the catch and move example written in Section III-E. Hence, we could write $\mathcal{K}_0 = \{v_{0,1}, v_{0,2}\}, i_0 \in \{1, 2\}$ that represents the states of catch and release, and $\mathcal{K}_1 = \{v_{1,1}, v_{1,2}, v_{1,3}\}, i_1 \in \{1, 2, 3\}$ that represents a top-view of the soft robot. Since this combination can be regarded as a multi-linear mapping, we therefore have

$$f(\mathbf{v}_1, \mathbf{v}_2) = \sum_{i_0=1}^{2} \sum_{i_1=1}^{3} \lambda_{0,i_0}\lambda_{1,i_1} g(i_0, i_1), \tag{5}$$

where $g(i_0, i_1)$ is a set of displacements of actuators in which the vertex combination of two simplicial complexes are i_0, i_1, respectively.

In general, assuming we have a multi-linear function $f : \mathbf{V}_1 \times ... \times \mathbf{V}_n \to \mathbf{W}$, where $\mathbf{V}_i \in \mathbb{R}^{d_i}$, $\mathbf{W} \in \mathbb{R}^d$, and d_i, d are the dimensions of $\mathbf{V}_i and \mathbf{W}$, respectively. The basis of $\mathbf{V}_i$ is $\{\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_j}\}$, and the basis of $\mathbf{W}$ is $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$.

$$\begin{aligned} f(\mathbf{v}_1, \ldots, \mathbf{v}_n) &= f(\mathbf{v}_1 \otimes \cdots \otimes \mathbf{v}_n) \\ &= \sum_{i_1=0}^{j_1} \cdots \sum_{i_j=0}^{j_n} v_{1,i_1} \ldots v_{n,i_j} f(\mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_j}) \\ &= (\mathbf{v}_1 \otimes \cdots \otimes \mathbf{v}_n) \circ \mathbf{T}, \end{aligned} \tag{6}$$

where the sign of $\circ$ is the Hadamard product, and the $\otimes$ is the tensor product.

Considering that $f(\mathbf{b}_{i_1} \otimes \cdots \otimes \mathbf{b}_{i_j}) = \sum_{k=0}^{d} \mathbf{A}_{i_1 \ldots i_j}^k \mathbf{b}_k$ is our definition, $f(\mathbf{b}_{i_1} \otimes \cdots \otimes \mathbf{b}_{i_j})$ could be also written as a tensor $\mathbf{T}$,

$$\mathbf{T} = \mathbf{A}_{i_1,\ldots,i_j} \mathbf{e}_{i_1,\ldots,i_j},$$

where $\mathbf{A}_{i_1 \ldots i_j} = \{A_{i_1 \ldots,i_j}^1, \ldots, A_{i_1 \ldots i_j}^d\}$, and $\mathbf{e}_{i_1,\ldots,i_j} = \mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_j})$.

Notice that the increasing dimensions when adding more independent parameters might cause extra labor on matching different postures to the vertex combinations. Users might be more cautious when composing their task graphs.

V. Typical Usages

In this section, we will introduce two typical usages of the proposed method by using an experimental GUI to create control UIs for cable-driven soft robot arms.

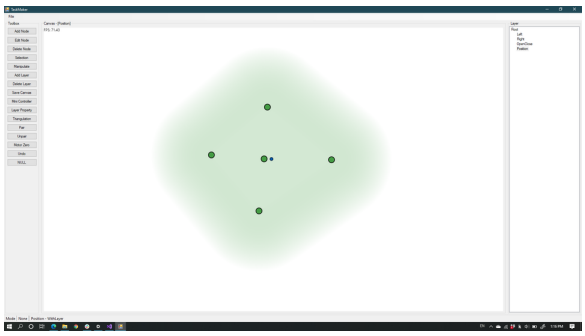
A. Graphical User Interface for Evaluations

We developed a graphical user interface (GUI) to allow users to design their desired robot control UI. The GUI shown in Figure 4a is coded in a C# programming language. Qhull [20],an open-source library implementing the Quickhull algorithm, is used for computing the convex hull and Delaunay triangulation. Notice that when we say "canvas" of the UI, it means the same as the simplicial complex introduced in the previous section.

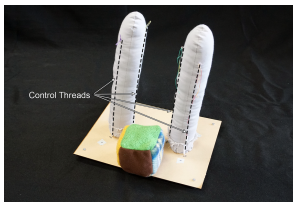
As the primary function of this system to implement our approach, we design a canvas-based UI to allow users to add/edit canvas, add/edit nodes, triangulate a manipulable task space, and assign displacements of actuators or points in low-level control UIs to nodes.

B. Soft Robot Arms

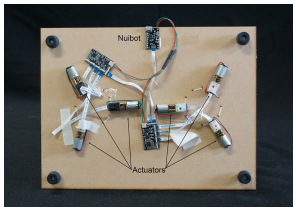
The soft robot arm system used for the evaluations is shown in Figure 4b. This system consists of two standalone soft arms with each arm having three driving cables sewed on it as Figure 4c shows. The control mechanism was proposed in Yamashita's works [2]. The cables are distributed evenly around the arm at an angle of 120 degrees and driven by three DC motor modules belonging to Nuibot [21], a development



(a) GUI



(b) Soft Robot Platform



(c) Soft Robot Platform: Bottom View

Fig. 4. Software and Hardware Platform for Evaluations.

kit for cable-driven soft robots. The cable is sewed from the bottom of the soft arm to the tip. The length of the tendons made by cable sections is not precisely defined, because we expect to evolve such manufacturing errors in standard fabrication procedures and show that our method is well adapted for this problem.

C. Manipulating Single Robot Arm

In this example, users first designed a simple simplicial complex that only includes three nodes as the control UI. Each vertex exactly matches a particular posture of the left arm of the soft robot. Considered the ignorance of the kinematic model for our soft robot, we directly adjust the displacement values of actuators to perform a desired posture. Then, users can directly manipulate the control UI's point by dragging the mouse to move around the task space created by the proposed UI. The point in the control UI will be interpolated and mapped to a set of displacements of three motors. Finally, this control UI could easily manipulate the soft robot arm, and the interpolated postures were matched to the users' expectations. The control UI, the defined postures, and interpolated or extrapolated postures are shown in Figure 5.

D. Taking Soft Dice by Two Robot Arms

In this example, users created two separated control UIs to enact a relatively complex robot task: taking a soft dice by two robot arms. One control UI consisted of two nodes is used to represent the taking and releasing status for the robot arms. The other control UI have five nodes that represents the coordinates of the dice from its top-view. The matching targets of the map of two control UIs are both inputs of the control UIs from two robot arms created previously. Since two control UIs are composited as a bi-linear simplicial map, users matched ten compositions of two sets of nodes in both control UIs with the inputs of low-level control UIs which are corresponding to the displacements of soft arms representing particular postures. Finally, users could manipulate both soft robot arms to take and release a soft dice on target positions. The control UI, the defined postures, and interpolated or extrapolated postures are shown in Figure 6.

VI. User Evaluation

To evaluate the usability of the proposed method, we conducted a user evaluation. At the end of this evaluation, we sought answers to three questions as follows:

- Is the proposed method convenient for casual users?
- Is the proposed method usable to create control UI for soft robots?
- How long will it take to create desired control UI by this proposed method in practice?

In order to let different casual users create control UIs for the same robot tasks, this evaluation is conducted in the tutorial style. The experimenter gave minimal supports to the participants through all steps of the evaluation.

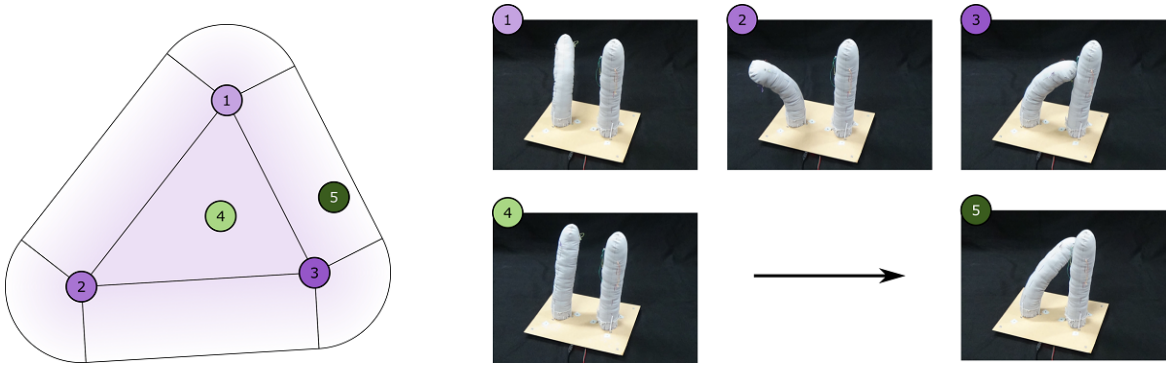
First, the experimenter created a control UI for the left arm of the soft robot as an example. Then the participants would be asked to create a control UI for the right arm of the soft robot in the same procedure as the experimenter showed before. For both soft arms, the control UI had three state nodes that represented the postures of the arm when a single actuator had been pulling its corresponding driving cable. The procedures to create a control UI followed the one introduced in Section III.

After finishing the creation of both soft arms, the participants were asked to create a high-level control UI for the manipulation task to generate a synchronized motion of both soft arms. The mapping targets of this high-level control UI were then the UIs of both two soft arms created previously.

A. Results and Discussion

In this evaluation, we invited 5 participants to use our GUI to accomplish a series of soft robot manipulation tasks. Notice that we only evaluate the participants with an interpolation-only GUI that partially implemented our method in this evaluation due to the schedule. The participants who joined this experiment are categorized into "Experienced with Soft Robots" and "Non-experienced with Soft Robots."

Most participants achieved the synchronized manipulating task in their way and enjoyed the UI they created to control soft arms for other simple manipulation tasks freely. Some participants also discussed the possibilities of achieving complex tasks and actively asked how to realize that in detail, which indicated they already had a clear awareness of the proposed concept and desired an advanced usage of the system. Also, these discussions showed that this method could inspire more participants' imagination of expressive applications of cable-driven soft robots.



(a) Control UI (b) Assigned poses and interpolated results

Fig. 5. Manipulating a single robot arm(Left only) Manipulating soft robots by three set initial vertices (node 1-3) of the simplicial complex with three pose-related actuator displacements. The green points in the control UI decide the interpolated and extrapolated results of actual postures of soft robots.

(a) Control UI (b) (Partial)Assigned poses (c) Interpolated results

Fig. 6. Manipulating soft robots by five set initial vertices (node 1-3 and white nodes) of the simplicial complex with five pose-related actuator displacements. The combinations of green and brown points in each control UIs decide the interpolated and extrapolated results of actual postures of soft robots.

The purpose of the synchronized manipulating task is to evaluate the chain mapping in a practical environment. The level structure of control UIs lets the participants focus on the current task and do not need to care about the lower-level UIs' details. Compared to manipulating actuators embedded in the soft robot directly, the chain of maps promises more possibilities for applications of soft robots. Additionally, the level structure could be used for the collaborative working of multiple soft robots.

Interestingly, the top-level control UIs created by different participants showed some similarities in their layouts. Almost all participants in this evaluation preferred to create their interpretations of the performing task into spatial-related layouts. This suggested that the spatial configuration of nodes is reasonable and understandable for most people. Despite these spatial similarities, the non-experienced group's layouts showed that they were more flexible in the geometric form than the experienced group, implying that experienced users might prefer to use a precise numerical representation of information due to their prior knowledge. However, the completeness of the performing task was acceptable for all participants, which proved that the task-oriented robustness to unspecified geometric representations is benefited by the topological aspect of the simplicial complex.

Although the current user experience of our GUI significantly impacted the proficiency when creating control UIs, the feasibility of our method has been confirmed in this evaluation. All the participants finished their experiments within 1 hour. In summary, we believe that the results of user evaluations have answered the questions we raised at the beginning of this section.

However, there were difficulties in the use of this interpolation-only GUI of our method during the experiments. When users transited from one task state to another, the trajectory of the controller approximated to be a series of line segments with imperfect geometric shapes. Moreover, users were used to finding the desired pose following their hand movement trends of manipulating the control point

with their observations. The lack of extrapolation violated the intuitions of the participants while the visual-tactile linkage was working. This result prompted us to implement the extrapolation method in further evaluations due to its importance.

VII. Conclusion and Future Work

This paper adopted a simplicial mapping method to allow casual users to design a task-oriented control UI for a cable-driven soft robot. Compared to conventional kinematic model methods, our method does not need prior knowledge and reduces the time and labor costs. The task space created by the proposed method provides more freedom, especially for casual users. Several practical cases that use the proposed GUI to create control UIs for particular tasks of a cable-driven soft robot arm system have been investigated. A qualitative user evaluation showed the proposed method's usability for casual users and suggested further quantitative user evaluations. This paper offers a generic solution well-adapted to the previously-mentioned question, lacking flexibility and reusability of soft robot control interfaces.

The soft robot we tested here was the cable-driven type; however, we believe this method can solve more generic problems located in the whole of soft robotics. We look forward to having more researchers join us to explore the possibilities of this method. In this paper, the user's observation is used as a baseline for how a soft robot performs tasks that do not need additional devices. User's personal experiences, therefore, influence the results.

For future work, we would like to combine this method in different configuration spaces that might be defined by a physics-based model or a statistic data-driven model. For example, we could map our user-defined task space to the configuration space of the FEM model to accomplish a simulation for the soft robot under an agile design and development. Moreover, a study of the relationship or combination of the proposed method with the end-effector's coordinates could be considered as a next step. Since the proposed method is to solve the kinematics problem without physics-based models, the combinations of force control or dynamics control will open more possibilities for soft robotics. This improvement of our GUI is also important and would be helpful to practice.

References

[1] M. W. Hannan and I. D. Walker, "Kinematics and the Implementation of an Elephant's Trunk Manipulator and Other Continuum Style Robots," *Journal of Robotic Systems*, vol. 20, no. 2, pp. 45–63, 2003.

[2] Y. Yamashita, T. Ishikawa, H. Mitake, Y. Takase, F. Kato, I. Susa, S. Hasegawa, and M. Sato, "Stuffed toys alive! cuddly robots from fantasy world," in *ACM SIGGRAPH 2012 Emerging Technologies*, ser. SIGGRAPH '12. New York, NY, USA: Association for Computing Machinery, 2012. [Online]. Available: https://doi.org/10.1145/2343456.2343476

[3] S. Seok, C. D. Onal, K.-J. Cho, R. J. Wood, D. Rus, and S. Kim, "Meshworm: A Peristaltic Soft Robot with Antagonistic Nickel Titanium Coil Actuators," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 5, pp. 1485–1497, 2013.

[4] A. D. Marchese, K. Komorowski, C. D. Onal, and D. Rus, "Design and Control of a Soft and Continuously Deformable 2D Robotic Manipulation System," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2189–2196, 2014.

[5] J. P. King, D. Bauer, C. Schlagenhauf, K.-H. Chang, D. Moro, N. Pollard, and S. Coros, "Design. Fabrication, and Evaluation of Tendon-Driven Multi-Fingered Foam Hands," *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, vol. 00, pp. 1–9, 2018.

[6] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, 2015.

[7] H. Wang, W. Chen, X. Yu, T. Deng, X. Wang, and R. Pfeifer, "Visual Servo Control of Cable-Driven Soft Robotic Manipulator," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 57–62, 2013.

[8] C. Duriez, "Control of Elastic Soft Robots based on Real-Time Finite Element Method," *2013 IEEE International Conference on Robotics and Automation*, pp. 3982–3987, 2013.

[9] M. Rolf and J. J. Steil, "Efficient Exploratory Learning of Inverse Kinematics on a Bionic Elephant Trunk," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1147–1160, 2014.

[10] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, "Neural Network and Jacobian Method for Solving the Inverse Statics of a Cable-Driven Soft Arm With Nonconstant Curvature," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 823–834, 2015.

[11] Y. Takase, H. Mitake, Y. Yamashita, and S. Hasegawa, "Motion generation for the stuffed-toy robot," in *The SICE Annual Conference 2013*, Sep. 2013, pp. 213–217.

[12] J. M. Bern, K.-H. Chang, and S. Coros, "Interactive design of animated plushies," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.

[13] C. Schlagenhauf, D. Bauer, K.-H. Chang, J. P. King, D. Moro, S. Coros, and N. Pollard, "Control of Tendon-Driven Soft Foam Robot Hands," *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, vol. 00, pp. 1–7, 2018.

[14] P. Rademacher, "View-dependent geometry," *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99*, pp. 439–446, 1999.

[15] T. Ngo, D. Cutrell, J. Dana, B. Donald, L. Loeb, and S. Zhu, "Accessible animation and customizable graphics via simplicial configuration modeling," *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pp. 403–410, 2000.

[16] T. Igarashi, T. Moscovich, and J. F. Hughes, "Spatial keyframing for performance-driven animation," *ACM SIGGRAPH 2007 courses on - SIGGRAPH '07*, p. 25, 2007.

[17] C. F. R. III, P. J. Sloan, and M. F. Cohen, "Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation," *Computer Graphics Forum*, vol. 20, no. 3, pp. 239–250, 2001.

[18] M. d. Berg, M. v. Kreveld, M. Overmars, and O. C. Schwarzkopf, *Computational Geometry, Algorithms and Applications*. Springer, 2000.

[19] M. Herlihy, D. Kozlov, and S. Rajsbaum, *Distributed computing through combinatorial topology*. Newnes, 2013.

[20] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, vol. 22, no. 4, pp. 469–483, 1996.

[21] G. Zile, D. Binod, L. Haoyan, N. Tomoka, M. Hironori, and H. Shoichi, "Nuibot: Motion control system and visual programming environment for string driven soft mechanism," *The Proceedings of JSME annual Conference on Robotics and Mechatronics (Robomec)*, vol. 2019, pp. 1P2–G10, 2019.