Differentiable rendering as a way to program cable-driven soft robots

Kasra Arnavaz,[1] Kenny Erleben,[1]

[1]Department of Computer Science, University of Copenhagen,

Universitetsparken 1, 2100 Copenhagen, Denmark

E-mails: kasra@di.ku.dk, kenny@di.ku.dk

December 1, 2023

Keywords: Soft Robots, Simulation, and Rendering

Abstract: Soft robots have gained increased popularity in recent years due to their adaptability and compliance. In this paper, we use a digital twin model of cable-driven soft robots to learn control parameters in simulation. In doing so, we take advantage of differentiable rendering as a way to instruct robots to complete tasks such as point reach, gripping an object, and obstacle avoidance. This approach simplifies the mathematical description of such complicated tasks, and removes the need for landmark points and their tracking. Our experiments demonstrate the applicability of our method.

1 Introduction

While rigid robots are suitable for industrial fabrication due to their precision, they are unfitting for home applications where contact safety is required [1]. Inspired by nature, soft robots are more adaptable and flexible, making them suitable for use-cases with little prior knowledge about the environment [2]. As a result, they have been used in mechanical gripping [3], exploration [4], and healthcare [5, 6]. This very flexibility, however, makes modeling and control of soft robots difficult, due to their high degrees of freedom [7]. Soft robots are underactuated, and it is costly to predict the effect of changing a control parameter through trial and error.

Previous work has addressed this challenge by applying optimization of model-based control, i.e. having a physical simulation model to help prevent the consequences of changing the control signals [8]. This is promising as a computer algorithm now can numerically explore the space of possible control solutions. Having said that, there are two open questions: 1) how to reduce the reality gap and 2) how to program the robot, i.e. how one would mathematically set up goals to be achieved. In [9, 10], rendering and computer vision combined with differentiability have shown great strength in addressing the first challenge through optimization of material parameters to ensure the simulated deformations are as close as possible to the ones observed in the real world.

In this work, we are inspired by how differentiability and rendering have created an elegant solution to simulator calibration, and we explore this paradigm as a means to more easily program complex tasks for soft robots. Such tasks are often expressed as point-constraints or point-tracking objectives [11, 12]. One needs to describe these functions to achieve desired behaviors. It is then up to manual trial and error to add more constraints to the setup. Take, for example, a gripping task, where a soft robot needs to conform to the surface of a geomet-

ric shape to maximize contact area resulting in a firm grasp. One would have to place multiple point samples manually on these surfaces to get an objective which yields to the desired outcome. Or the case where the robot needs to avoid dangerous areas in space, where other objects or fabrication processes (e.g. a water jet cutter) could physically damage it. One solution is to put dummy shapes in a simulation to force a soft robot away from those no-touch zones, but that would bias the solution since one relies on contact forces between a robot and a dummy shape to stay away.

We recast such scenarios into rendering interpretations such that these tasks are redefined through depth images and thus eliminating the need to define point correspondences to solve tasks. In summary, our contributions in this paper are

- formulating grasping and avoidance tasks for curved surfaces with differentiable rendering,
- proposing a simple mathematical description for maximal contact area for a gripping task,
- and instructing the soft robot to avoid certain zones without contact forces.

2 Related Works

Modeling Soft robots are infinite-dimensional in their degrees of freedom, making modeling them a trade-off between tractability and accuracy [13]. Analytical models tend to impose restrictive assumptions on the dynamics of the robot [14, 15, 16], limiting the generality of their applications. Numerical methods have proven to be more precise in practice [14] and can tackle shapes of higher complexity at the higher cost of computation. Among them, the Finite

Element Method (FEM) has shown success in describing nonlinear dynamics as well as contact interactions with objects [17, 18].

Simulation Several engines have been developed based on various modeling techniques. Since differentiability is a core part of our pipeline, we focus our attention on differentiable simulators suited for soft deformable bodies. To name a few, DiffAqua [19], ChainQueen [20], and Warp [21] have been used for soft deformable bodies. While we use Warp in this work, our method is not limited to this choice and can be substituted with any differentiable simulator. Our work is closest to [9, 10], where differentiability has been utilized to calibrate physical properties of soft bodies. We utilize differentiable rendering as a language to express soft robotics tasks and tackle control problems

Control Actuation of soft robots can be divided into pneumatic- and control-based methods. The latter relies on air pressure change and the former on cable length change to control the behavior of soft robots. In this paper, we focus on cable-driven robots. Model-based control has been extensively surveyed in [13]. In particular, we use the FEM to iteratively simulate the behavior of the soft robot and use physical modeling of the cables by damped springs to obtain cable forces. While model-free approaches are getting more popular [22], they require iterative time-consuming training to find control strategies [23].

3 Method

We argue that one could solve tasks such as gripping and obstacle avoidance by modeling the contact surface between the robot and the objects in the scene. We achieve this goal by rendering this surface by viewing the world from the object's perspective, i.e. we place cameras inside the object and render the

object's surface and the robot. With those depth images, we propose a fully differentiable pipeline (Figure 1) to learn the control parameters of cable-driven soft robots.

Our approach has three main functions, namely simulation, rendering, and objective functions. The simulation function takes the control parameters $\mathbf{p}$ as input and outputs the state of the scene after a duration of t seconds, i.e. $\mathbf{s}_t = \mathcal{S}(\mathbf{p})$. The state contains the position and velocity of all mesh nodes in the scene. In order to get $\mathbf{s}_t$, one needs to compute the cable forces as well as elastic forces and gravitational force iteratively—which we cover in Section 3.1. The rendering function takes the state of the three-dimensional scene and converts it to two-dimensional depth images for every mesh in the scene, i.e. $\mathbf{I}_t = \mathcal{R}(\mathbf{s}_t)$. The intensity of these grayscale images conveys how far the mesh is from the camera, with darker pixels denoting closeness to the camera and brighter pixels denoting further distance to the camera. Finally, the objective function takes these depth images and outputs a scalar loss, i.e. $\ell = \mathcal{O}(\mathbf{I}_t)$.

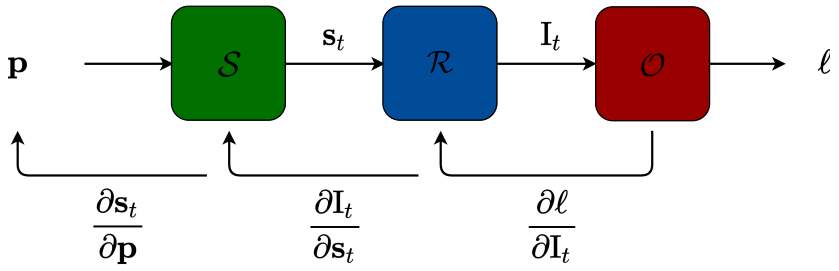

Figure 1: The data flow of our differentiable pipeline. The simulation function $\mathcal{S}$ produces a state $\mathbf{s}_t$ using control parameters $\mathbf{p}$. The state is then projected to an image $\mathbf{I}_t$, which is depending on the objective $\mathcal{O}$ is converted to a scalar loss ℓ. The chain rule can then be applied to compute the derivative of ℓ w.r.t. control parameters $\mathbf{p}$.

3.1 Cable Force

In order to simulate the behavior of the cable driven robot, we need to model how cable forces are computed. Once the forces are obtained, the simulation engine does numerical integration to calculate the resulting change in position and velocity.

Consider the robotic finger shown in Figure 2, which has one cable running through it. We are interested in computing the forces at each point where the cable enters/leaves the robot, also known as via points. We will model the cable forces by multiple identical springs at each interval. For every force at each via point, there is an equal but opposite force in the next via point where the spring ends—shown in Figure 2 by arrows of the same color. Given that we have H via points for a cable, the forces corresponding via point i are computed as

$$\mathbf{f}_i = \begin{cases} pk(\mathbf{h}_2 - \mathbf{h}_1) - b\dot{\mathbf{h}}_1 & \text{if } i = 1, \\ -\mathbf{f}_{H-1} - b\dot{\mathbf{h}}_H & \text{if } i = H, \\ -\mathbf{f}_{i-1} + \mathrm{p}k(\mathbf{h}_{i+1} - \mathbf{h}_i) - b\dot{\mathbf{h}}_i & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathbf{h}_i, \dot{\mathbf{h}}_i, \mathbf{f}_i \in \mathbb{R}^3$ are the position, velocity, and force of via point i respectively, k is the spring stiffness, b is the damping coefficient, and $p \in \mathbb{R}$ is the control parameter we would like to determine to make our robot achieve a particular task. If we denote the position of the tip of the cable by $\mathbf{h}_0$, then p, also referred to as the pull ratio, is given by

$$p = \frac{||\mathbf{h}_1 - \mathbf{h}_0||_2}{||\mathbf{h}_H - \mathbf{h}_1||_2} \ . \tag{2}$$

If the cable is not pulled at all, then $\mathbf{h}_0 = \mathbf{h}_1$ or equivalently $p = 0$, and since $\dot{\mathbf{h}}_i = \vec{0}$, then all cable forces at via points would be zero ($\mathbf{f}_i = \vec{0}$). If a robot has C cables, all pull ratios are stored in a vector denoted by $\mathbf{p} \in \mathbb{R}^C$.

Thus far, we have computed the cable forces at the position of via points, but our simulation requires having the cable forces at each node. To perform this mapping, we will be using the barycentric coordinates of $\mathbf{h}_i$ inside the tetrahedron that contains it. If our mesh has N nodes, then we will be getting a matrix of barycentric coordinates $\mathbf{W} \in \mathbb{R}^{H \times N}$, which sums to one in each row by definition of barycentric coordinates. Let us collect forces at every via point in a column vector defined as $\mathbf{f} = (\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_H)^T$. Similarly, if we denote the force at node j with $\mathbf{F}_j$, we have $\mathbf{F} = (\mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_N)^T$. We would then have the relationship $\mathbf{f} = \mathbf{W}\mathbf{F}$. Given that $N \ll H$, $\mathbf{W}^T\mathbf{W}$ is invertible and we have

$$\mathbf{F} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{f}. \tag{3}$$

These cable forces are going to be added to the elastic forces as an external force, i.e.

$$\mathbf{F}_{\text{total}} = \sum_{c \in \mathcal{C}} \mathbf{F}^c + \mathbf{F}_{\text{elastic}} + \mathbf{F}_{\text{gravity}},$$

where $\mathbf{F}^c$ denotes the nodal force matrix of cable c and $\mathcal{C}$ is the set of cables in a given soft robot. Note that $\mathbf{F}_{\text{elastic}}$ depends on the type of material the soft robot is made of, which can be specified with Young's modulus, Poisson's ratio, density, and damping factor.

3.2 Robot Tasks

Robots in isolation cannot achieve many meaningful tasks. In our experiments in Section 4, we are interested in the interaction of the soft robot with other objects. In particular, we are interested in programming the robot to a) grip an object with maximal contact and b) avoid obstacles while reaching a point in space. The key ingredient to solving both these tasks is having a measure of distance between the robot and the object of interest. Then, the obstacle

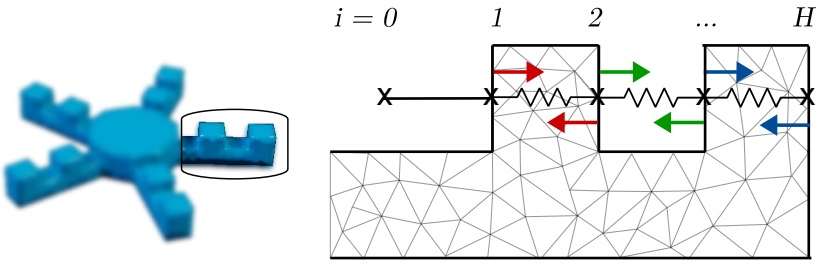

Figure 2: **Left:** A soft robot design whose fingers are curled by a cable running through them. **Right:** A depiction of how cable forces are computed for one finger. Cables in each segment are modeled with identical springs with damping to obtain forces at via points. Barycentric coordinates are then used to map these forces to nodal points.

avoidance task would translate to learning control parameters to maximize this distance while gripping corresponds to minimizing it.

Let us denote the robot by r, the object we want to grip or the point to reach by s, and the set of obstacle objects in the scene by $\mathcal{Q}$. To obtain such a distance measure, we will place the camera at various angles inside the object of interest, i.e. the obstacle we want to avoid or the object we would like to grip. The set of camera views for each object is defined by $\mathcal{V}$, and the set of time frames where an image has been recorded is defined by $\mathcal{T}$. Finally, the depth image of the robot r and the object s at time $t \in \mathcal{T}$ with camera view $v \in \mathcal{V}$ with $\mathbf{I}_t^r(v)$ and $\mathbf{I}_t^s(v)$, respectively. Then the distance measure is defined by

$$\Delta \mathbf{I}_v(r, s, t) \equiv \max(\mathbf{I}_t^r(v) - \mathbf{I}_t^s(v), 0), \tag{4}$$

where $\max(\cdot, 0)$ sets the distance to zero when the robot penetrates the object.

We can then write the loss of the gripping task by robot r on object s as

$$\ell_{\text{grip}}(r, s) \equiv \frac{1}{|\mathcal{T}||\mathcal{V}|} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} \Delta \mathbf{I}_v(r, s, t). \tag{5}$$

Similarly, the loss corresponding to the avoidance of obstacle q is defined as

$$\ell_{\text{avoid}}(r, q) \equiv -\frac{1}{|\mathcal{T}||\mathcal{V}|} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} \Delta \mathbf{I}_v(r, q, t). \tag{6}$$

Both task can be achieved simultaneously with a multi-object loss as

$$\ell(r, s) \equiv \alpha \ell_{\text{grip}}(r, s) + \frac{(1-\alpha)}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \ell_{\text{avoid}}(r, q), \tag{7}$$

where the first part encourages reaching the point s, the second part penalizes getting close to the obstacles, and $\alpha \in [0, 1]$ is the weighting parameter between the two parts. Figure 3 illustrates the structure of gripping and avoidance losses with an example.

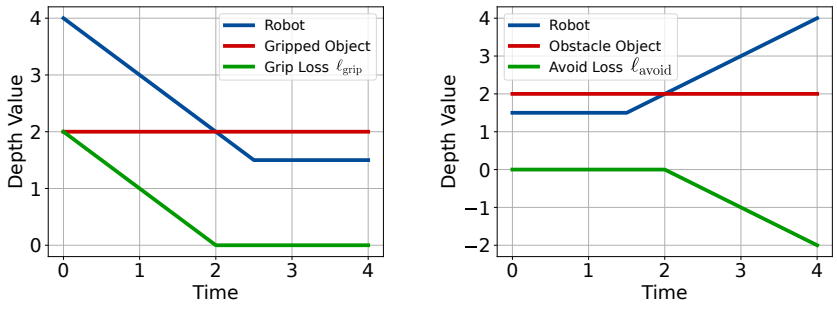

Figure 3: Didactic plots for depicting loss values given a robot depth map. **Left:** When gripping, the robot gets closer to the object over time and might penetrate it depending on the collision resolution. **Right:** The robot is assumed to be penetrating the obstacle initially and gets further away from it with time. Notice, how lower loss values would translate to gripping and avoiding in each case. In both cases, the object does not move, and as a result its depth value remains constant.

4 Experiments

We have conducted four experiments to demonstrate the effectiveness and simplicity of our approach. We have viewed the robot-object interactions as attraction (e.g. point reach and gripping) and repulsion (e.g. obstacle avoidance). Differentiable rendering has proven to be a potent tool in modeling both scenarios and learning the control parameters with gradient-based methods. We have used Pytorch3D [24] as our differentiable renderer and Gradient Descent as our optimizer. Initially, the pull ratio introduced in Equation (2) is set to zero for all cables in all experiments and then updated according to the gradients. Furthermore, pull ratios are clipped to zero, if their value is negative. In the following, we describe each experiment. Table 1 contains the parameters which were common in all experiments, and Table 2 shows the custom parameters for each experiment.

Table 1: Common parameters in all experiments

Parameter	Value
Young's modulus	149 kPa
Poisson's ratio	0.40
Damping factor	0.40
Density	1080 kg/m^3
Cable stiffness	100 N/m
Cable damping coefficient	0.01 kg/s
Simulation time step	5×10^{-5} s

Table 2: Experiment-specific parameters

Experiment	Cables	Vertices	Tetrahedra	Duration	Learning rate
Reach	3	2830	10748	2 s	0.1
Avoidance	3	2830	10748	2 s	1.0
Cylinder	5	2559	9407	3 s	10^{-5}
Egg	5	764	1753	2 s	10^{-3}

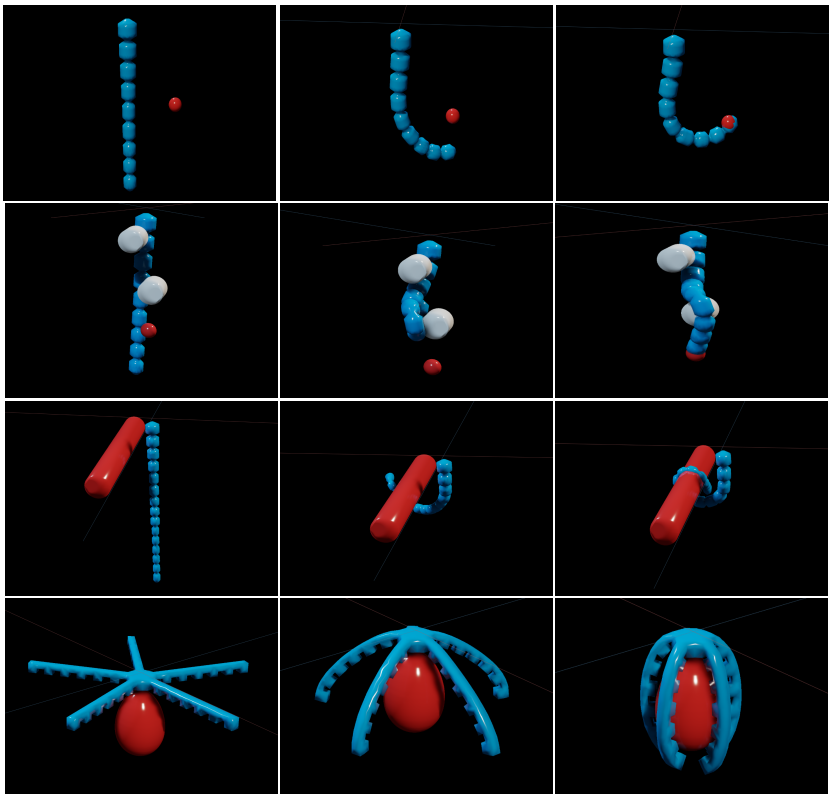

Figure 4: Snapshots of the scene showing progression of tasks. **First row:** Reach Experiment at 0, 0.2, 2 seconds. **Second row:** Avoidance Experiment at 0, 0.2, 2 seconds. **Third row:** Cylinder Experiment at 0, 0.5, 3 seconds. **Forth row:** Egg Experiment at 0, 0.25, 2 seconds.

Reach Experiment The goal in this experiment is to get the robot to reach a point in 3D space (Figure 4 first row). The red ball denotes the point to be reached. There are no contact forces between the robot and the ball and as a result it does not exist from robot's standpoint. Six cameras have been placed inside the ball to compute the distance measure in Equation (4). The robot has three cables and hence three pull ratio parameters to be optimized. The loss, pull ratio values, and gradients are shown in Figure 5 first row.

Avoidance Experiment Building on top of Reach Experiment, cylindrical obstacles are now inserted on robot's path to reach the point (Figures 4 and 5 second row). Six cameras are placed inside all three objects looking at every direction. None of the objects in this scene produce contact forces (not the ball nor the obstacles). The obstacles are therefore only there as areas where the robot is not supposed to go, and there is no collision forces which prevent it from touching the obstacles. The robot is discouraged from touching the obstacles by the increase in loss and is encouraged to reach the ball by the reduction in loss as described in Equation (7). The weight parameter α is set to 0.7 after some trial and error.

Cylinder Experiment The trunk robot in this task is designed to be longer compared to previous tasks to showcase the maximal contact behavior in gripping around the cylinder shaft. Figure 6 shows the distance measure (4) before and after optimization for all six cameras placed inside the cylinder. Notice how the distance is smaller after optimization when the cylinder has been gripped. By lowering the loss in Equation (5) and enabling the collision detection in the simulation engine, the robot curls around the cylinder shaft (Figures 4 and 5 third row).

Egg Experiment To demonstrate that this approach is applicable to other designs of soft robots, we have made a starfish gripper with five fingers with one cable running through each finger. The task is to have maximal contact area with an egg shell object (Figures 4 and 5 fourth row). The experimental setup are similar to that of Cylinder Experiment.

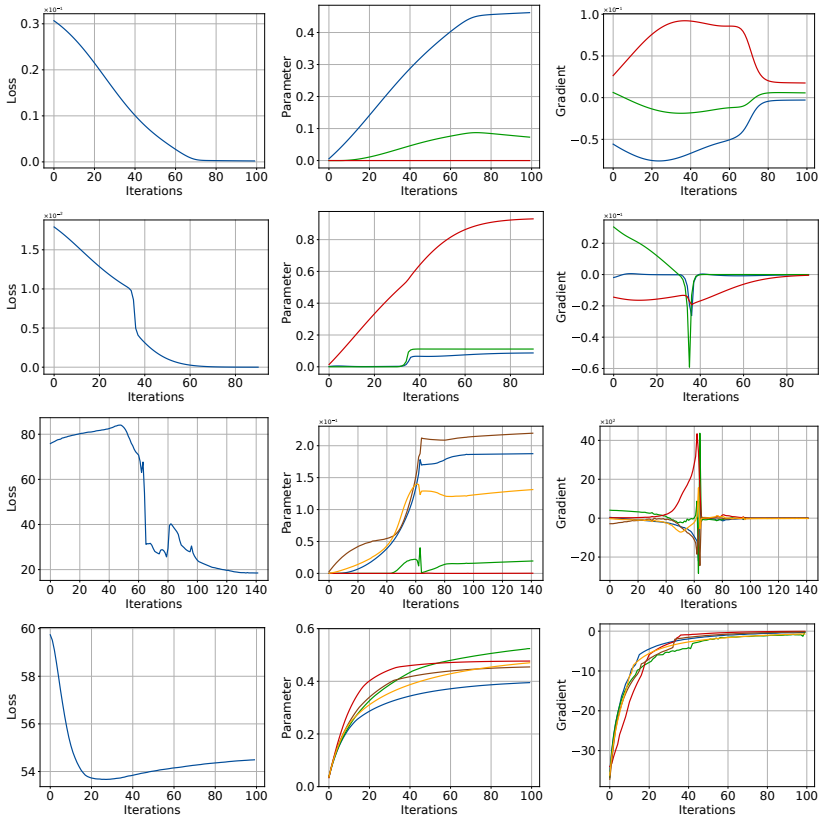

Figure 5: Convergence plots for all experiments. **First row:** Reach Experiment. **Second row:** Avoidance Experiment. **Third row:** Cylinder Experiment. **Fourth row:** Egg Experiment. The columns show the loss, parameter values for each cable, and gradients for each cable.

5 Conclusion

This paper presents a novel approach to programming robots using differentiable rendering. The strength of our method lies in the simplicity of formulating complex robotics tasks such as mechanical gripping of objects and obstacle avoidance, both of which can be expressed with depth images obtained from interior view these objects. This eliminates the need to come up with point correspondences and tracking these landmarks, a complex computer vision task

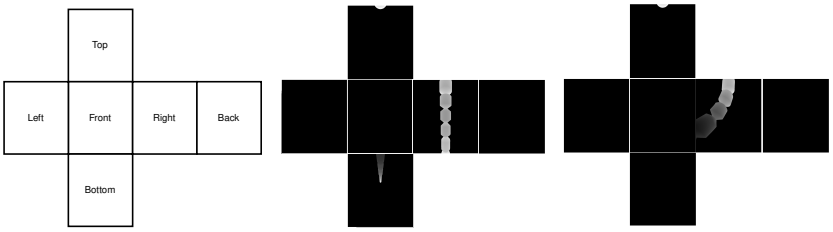

Figure 6: The distance measure introduced in Equation (4) for Cylinder Experiment. Brighter values denote further distance between the robot and object and vice versa. **Left:** The cube map for all six camera views. **Middle:** The distance measure before optimization of control parameters ($\mathbf{p} = \vec{0}$). **Right:** The distance measure as a result of optimized control parameters.

with real world data. Our experiments demonstrate the effectiveness of this programming paradigm for soft robots. A limitation of our work is that learned control parameters apply only to those specific predefined environments. It is a future work to combine this framework with reward functions and reinforcement learning to learn control *policies*, which would be more robust to changes in the environment.

References

[1] Hod Lipson. Challenges and opportunities for design, simulation, and fabrication of soft robots. *Soft Robotics*, 1(1):21–27, 2014.

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

[3] Jun Shintake, Vito Cacucciolo, Dario Floreano, and Herbert Shea. Soft robotic grippers. *Advanced materials*, 30(29):1707035, 2018.

[4] Simona Aracri, Francesco Giorgio-Serchi, Giuseppe Suaria, Mohammed E Sayed, Markus P Nemitz, Stephen Mahon, and Adam A Stokes. Soft robots

for ocean exploration and offshore operations: A perspective. *Soft Robotics*, 8(6):625–639, 2021.

[5] Yong Zhong, Luohua Hu, and Yinsheng Xu. Recent advances in design and actuation of continuum robots for medical applications. In *Actuators*, volume 9, page 142. MDPI, 2020.

[6] Matteo Cianchetti, Cecilia Laschi, Arianna Menciassi, and Paolo Dario. Biomedical applications of soft robotics. *Nature Reviews Materials*, 3(6):143–153, 2018.

[7] Mathieu Dubied, Mike Yan Michelis, Andrew Spielberg, and Robert Kevin Katzschmann. Sim-to-real for soft robots using differentiable fem: Recipes for meshing, damping, and actuation. *IEEE Robotics and Automation Letters*, 7(2):5015–5022, 2022.

[8] Eulalie Coevoet, Thor Morales-Bieze, Frederick Largilliere, Zhongkai Zhang, Maxime Thieffry, Mario Sanz-Lopez, Bruno Carrez, Damien Marchal, Olivier Goury, Jeremie Dequidt, et al. Software toolkit for modeling, simulation, and control of soft robots. *Advanced Robotics*, 31(22):1208–1224, 2017.

[9] K Arnavaz, M Kragballe Nielsen, PG Kry, M Macklin, and K Erleben. Differentiable depth for real2sim calibration of soft body simulations. In *Computer Graphics Forum*, volume 42, pages 277–289. Wiley Online Library, 2023.

[10] J Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. In *International conference on learning representations*, 2020.

[11] Frederick Largilliere, Valerian Verona, Eulalie Coevoet, Mario Sanz-Lopez, Jeremie Dequidt, and Christian Duriez. Real-time control of soft-robots using asynchronous finite element modeling. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2550–2555. IEEE, 2015.

[12] Christian Duriez. Control of elastic soft robots based on real-time finite element method. In *2013 IEEE international conference on robotics and automation*, pages 3982–3987. IEEE, 2013.

[13] Cosimo Della Santina, Christian Duriez, and Daniela Rus. Model-based control of soft robots: A survey of the state of the art and open challenges. *IEEE Control Systems Magazine*, 43(3):30–65, 2023.

[14] Pierre Schegg and Christian Duriez. Review on generic methods for mechanical modeling, simulation and control of soft robots. *Plos one*, 17(1):e0251059, 2022.

[15] Hesheng Wang, Weidong Chen, Xiaojin Yu, Tao Deng, Xiaozhou Wang, and Rolf Pfeifer. Visual servo control of cable-driven soft robotic manipulator. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 57–62. IEEE, 2013.

[16] Federico Renda, Frédéric Boyer, Jorge Dias, and Lakmal Seneviratne. Discrete cosserat approach for multisection soft manipulator dynamics. *IEEE Transactions on Robotics*, 34(6):1518–1533, 2018.

[17] Nicholas W Bartlett, Michael T Tolley, Johannes TB Overvelde, James C Weaver, Bobak Mosadegh, Katia Bertoldi, George M Whitesides, and Robert J Wood. A 3d-printed, functionally graded soft robot powered by combustion. *Science*, 349(6244):161–165, 2015.

[18] Hadrien Courtecuisse, Jérémie Allard, Pierre Kerfriden, Stéphane PA Bordas, Stéphane Cotin, and Christian Duriez. Real-time simulation of contact and cutting of heterogeneous soft-tissues. *Medical image analysis*, 18(2):394–410, 2014.

[19] Pingchuan Ma, Tao Du, John Z Zhang, Kui Wu, Andrew Spielberg, Robert K Katzschmann, and Wojciech Matusik. Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021.

[20] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*, pages 6265–6271. IEEE, 2019.

[21] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. `https://github.com/nvidia/warp`, March 2022. NVIDIA GPU Technology Conference (GTC).

[22] Gabriele Tiboni, Andrea Protopapa, Tatiana Tommasi, and Giuseppe Averta. Domain randomization for robust, affordable and effective closed-loop control of soft robots. *arXiv preprint arXiv:2303.04136*, 2023.

[23] Mengyu Wu, Ying Zhang, Xuanye Wu, Zhiheng Li, Wenlin Chen, and Lina Hao. Review of modeling and control methods of soft robots based on machine learning. In *2023 42nd Chinese Control Conference (CCC)*, pages 4318–4323. IEEE, 2023.

[24] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.